CONFIGURATION GUIDE

# Extreme NetIron Layer 3 Routing Configuration Guide, 06.3.00

## Supporting NetIron OS 06.3.00

*9035621*
*31 August 2018*

© 2018, Extreme Networks, Inc. All Rights Reserved.

Extreme Networks and the Extreme Networks logo are trademarks or registered trademarks of Extreme Networks, Inc. in the United States and/or other countries. All other names are the property of their respective owners. For additional information on Extreme Networks Trademarks please see www.extremenetworks.com/company/legal/trademarks. Specifications and product availability are subject to change without notice.

# Contents

# Preface



# Document conventions

The document conventions describe text formatting conventions, command syntax conventions, and important notice formats used in Extreme technical documentation.

## Notes, cautions, and warnings

Notes, cautions, and warning statements may be used in this document. They are listed in the order of increasing severity of potential hazards.

NOTE
A Note provides a tip, guidance, or advice, emphasizes important information, or provides a reference to related information.

ATTENTION
An Attention statement indicates a stronger note, for example, to alert you when traffic might be interrupted or the device might reboot.

CAUTION
**A Caution statement alerts you to situations that can be potentially hazardous to you or cause damage to hardware, firmware, software, or data.**

DANGER
***A Danger statement indicates conditions or situations that can be potentially lethal or extremely hazardous to you. Safety labels are also attached directly to products to warn of these conditions or situations.***

## Text formatting conventions

Text formatting conventions such as boldface, italic, or Courier font may be used to highlight specific words or phrases.

| Format | Description |
|---|---|
| **bold** text | Identifies command names. |
| | Identifies keywords and operands. |
| | Identifies the names of GUI elements. |
| | Identifies text to enter in the GUI. |
| *italic* text | Identifies emphasis. |
| | Identifies variables. |
| | Identifies document titles. |
| `Courier font` | Identifies CLI output. |

| Format | Description |
| --- | --- |
| | Identifies command syntax examples. |

## Command syntax conventions

Bold and italic text identify command syntax components. Delimiters and operators define groupings of parameters and their logical relationships.

| Convention | Description |
| --- | --- |
| **bold** text | Identifies command names, keywords, and command options. |
| *italic* text | Identifies a variable. |
| [ ] | Syntax components displayed within square brackets are optional.<br>Default responses to system prompts are enclosed in square brackets. |
| { **x** \| **y** \| **z** } | A choice of required parameters is enclosed in curly brackets separated by vertical bars. You must select one of the options. |
| **x** \| **y** | A vertical bar separates mutually exclusive elements. |
| < > | Nonprinting characters, for example, passwords, are enclosed in angle brackets. |
| ... | Repeat the previous element, for example, *member*[*member*...]. |
| \ | Indicates a "soft" line break in command examples. If a backslash separates two lines of a command input, enter the entire command at the prompt without the backslash. |

# Extreme resources

Visit the Extreme website to locate related documentation for your product and additional Extreme resources.

White papers, data sheets, and the most recent versions of Extreme software and hardware manuals are available at www.extremenetworks.com. Product documentation for all supported releases is available to registered users at www.extremenetworks.com/support/documentation.

# Document feedback

Quality is our first concern at Extreme, and we have made every effort to ensure the accuracy and completeness of this document. However, if you find an error or an omission, or you think that a topic needs further development, we want to hear from you.

You can provide feedback in two ways:

- Use our short online feedback form at https://www.extremenetworks.com/documentation-feedback/.
- Email us at documentation@extremenetworks.com.

Provide the publication title, part number, and as much detail as possible, including the topic heading and page number if applicable, as well as your suggestions for improvement.

# Contacting Extreme Technical Support

As an Extreme customer, you can contact Extreme Technical Support using one of the following methods: 24x7 online or by telephone. OEM customers should contact their OEM/solution provider.

If you require assistance, contact Extreme Networks using one of the following methods:

- GTAC (Global Technical Assistance Center) for immediate support
  - Phone: 1-800-998-2408 (toll-free in U.S. and Canada) or +1 408-579-2826. For the support phone number in your country, visit: www.extremenetworks.com/support/contact.
  - Email: support@extremenetworks.com. To expedite your message, enter the product name or model number in the subject line.
- GTAC Knowledge - Get on-demand and tested resolutions from the GTAC Knowledgebase, or create a help case if you need more guidance.
- The Hub - A forum for Extreme customers to connect with one another, get questions answered, share ideas and feedback, and get problems solved. This community is monitored by Extreme Networks employees, but is not intended to replace specific guidance from GTAC.
- Support Portal - Manage cases, downloads, service contracts, product licensing, and training and certifications.

Before contacting Extreme Networks for technical support, have the following information ready:

- Your Extreme Networks service contract number and/or serial numbers for all involved Extreme Networks products
- A description of the failure
- A description of any action(s) already taken to resolve the problem
- A description of your network environment (such as layout, cable type, other relevant environmental information)
- Network load at the time of trouble (if known)
- The device history (for example, if you have returned the device before, or if this is a recurring problem)
- Any related RMA (Return Material Authorization) numbers

# About This Document



# Supported hardware and software

**End of Support for ExtremeSwitching CES 2000 Series devices**

Beginning with NetIron OS 06.3.00 and later, the ExtremeSwitching CES 2000 Series devices are not supported. Refer to the End of Sale and End of Support page for additional information.

The hardware platforms in the following table are supported by this release of this guide.

**TABLE 1** Supported devices

| ExtremeRouting XMR Series | ExtremeRouting MLX Series | ExtremeRouting CER 2000 Series |
|---|---|---|
| XMR 4000 | MLX-4 | CER 2024C |
| XMR 8000 | MLX-8 | CER-RT 2024C |
| XMR 16000 | MLX-16 | CER 2024F |
| XMR 32000 | MLX-32 | CER-RT 2024F |
| | MLXe-4 | CER 2048C |
| | MLXe-8 | CER-RT 2048C |
| | MLXe-16 | CER 2048CX |
| | MLXe-32 | CER-RT 2048CX |
| | | CER 2048F |
| | | CER-RT 2048F |
| | | CER 2048FX |
| | | CER-RT 2048FX |

# What's new in this document

On October 30, 2017, Extreme Networks, Inc. acquired the data center networking business from Brocade Communications Systems, Inc. This document has been updated to remove or replace references to Brocade Communications, Inc. with Extreme Networks., Inc., as appropriate.

For the complete list of supported features and the summary of enhancements and configuration notes for this release, refer to the *Extreme NetIron OS Release Notes*.

| Feature | Description | Described in |
|---|---|---|
| BGP Large Communities | The BGP Large Communities attribute is defined as an optional path attribute of variable length (an unordered set of Large Community values. Each Large Community value is encoded as a 12-byte quantity. | BGP Large Communities on page 306 |

# How command information is presented in this guide

Starting with Extreme NetIron 5.6.00, command syntax and parameter descriptions are removed from commands that are referenced in configuration tasks. To find the full description of a specific command, including all required and optional keywords and variables, refer to the *Extreme NetIron Command Reference* for your software release.

# ARP



# Basic ARP configuration

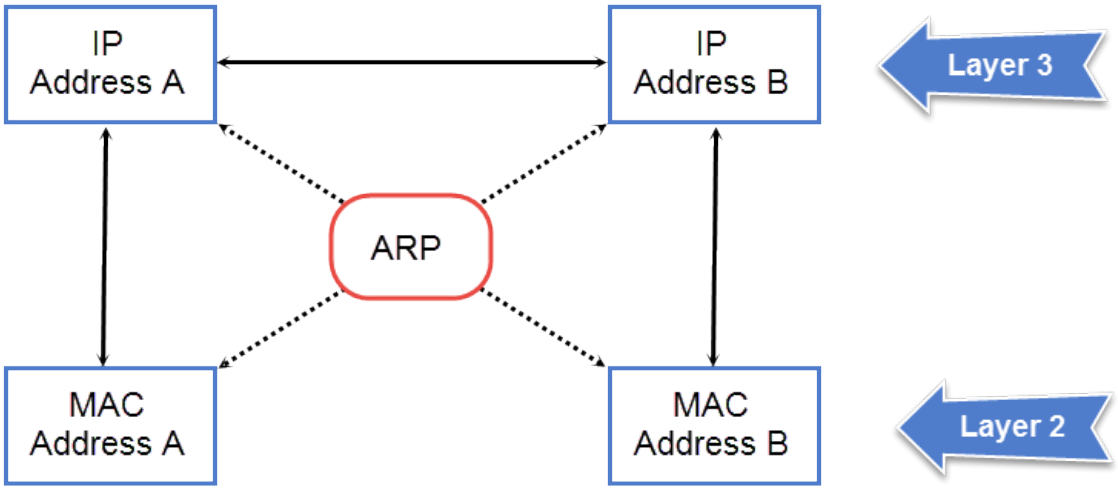
Address Resolution Protocol (ARP) enables a device to obtain the MAC address of another device's interface based on the other's IP address.

ARP is enabled by default and cannot be disabled.

## How ARP works

The following figure illustrates the purpose of ARP. If Device A wants to communicate with Device B, knowing the IP address of Device B is not sufficient. The MAC address is also required. ARP supplies the MAC address.

**FIGURE 1** ARP supplies the MAC address corresponding to an IP address




The NetIron OS device needs to know a destination's MAC address when forwarding traffic because this device encapsulates the IP packet in a Layer 2 packet (MAC layer packet) and sends the Layer 2 packet to a MAC interface on a device directly attached to the NetIron OS device. The device can be the packet's final destination or the next-hop router toward the destination.

The NetIron OS device encapsulates IP packets in Layer 2 packets regardless of whether the ultimate destination is locally attached or is multiple router hops away. Since the NetIron OS device's IP route table and IP forwarding cache contain IP address information but not MAC address information, the NetIron OS device cannot forward IP packets based solely on the information in the route table or forwarding cache. The NetIron OS device needs to know the MAC address that corresponds with the IP address of either the packet's locally attached destination or the next-hop router that leads to the destination.

For example, to forward a packet whose destination is multiple router hops away, the NetIron OS device must send the packet to the next-hop router toward its destination, or to a default route or default network route if the IP route table does not contain a route to the packet's destination. In each case, the NetIron OS device must encapsulate the packet and address it to the MAC address of a locally attached device, the next-hop router toward the IP packet's destination.

To obtain the MAC address required for forwarding a datagram, the NetIron OS device does the following:

1. First, the device looks in the ARP cache (not the static ARP table) for an entry that lists the MAC address for the IP address. The ARP cache maps IP addresses to MAC addresses. The cache also lists the port attached to the device and, if the entry is dynamic, the age of the entry. A dynamic ARP entry enters the cache when the device receives an ARP reply or receives an ARP request (which contains the sender's IP address and MAC address). A static entry enters the ARP cache from the static ARP table (which is a separate table) when the interface for the entry comes up. To ensure the accuracy of the ARP cache, each dynamic entry has its own age timer. The timer is reset to zero each time the NetIron OS device receives an ARP reply or ARP request containing the IP address and MAC address of the entry. If a dynamic entry reaches its maximum allowable age, the entry times out, and the software removes the entry from the table. Static entries do not age out and can be removed only by you.
2. If the ARP cache does not contain an entry for the destination IP address, the NetIron OS device broadcasts an ARP request out all its IP interfaces. The ARP request contains the IP address of the destination. If the device with the IP address is directly attached to the NetIron OS device, the device sends an ARP response containing its MAC address. The response is a unicast packet addressed directly to the NetIron OS device. The NetIron OS device places the information from the ARP response into the ARP cache. ARP requests contain the IP address and MAC address of the sender, so all devices that receive the request learn the MAC address and IP address of the sender and can update their own ARP caches accordingly.

   NOTE
   The ARP request broadcast is a MAC broadcast, which means the broadcast goes only to devices that are directly attached to the NetIron OS device. A MAC broadcast is not routed to other networks. However, some routers, including the NetIron OS device, can be configured to reply to ARP requests from one network on behalf of devices on another network. Refer to "Proxy ARP."

   NOTE
   If the device receives an ARP request packet that it is unable to deliver to the final destination because of the ARP timeout and no ARP response is received (the NetIron OS device knows of no route to the destination address), the device sends an ICMP Host Unreachable message to the source.

# Rate limiting ARP packets

For rate-limiting purposes, ARP traffic destined for the CPU is assigned a separate global QoS ID 0xFFE. You can configure the rate-limit parameters using the following global CONFIG command.

```
device(config)# ip rate-limit arp policy-map policy-map-name
```

By default, the rate-limit parameters for QoS ID 0xFFE will be initialized to allow line-rate traffic. The rate-limit parameters specified using the policy-map are applicable on a per-PPCR basis.

To display ARP accounting statistics, enter the following command.

```
device(config)# show rate-limit arp
```

This command displays the byte counters corresponding to QoS ID 0xFFE.

```
device(config)# clear rate-limit arp
```

This command clears the byte counters corresponding to QoS ID 0xFFE.

When priority-based rate limiting is enabled, QoS IDs 0x3FE, 0x7FE and 0xBFE will be re-mapped to 0xFFE. When priority-based rate limiting is disabled, QoS IDs 0x3FE, 0x7FE and 0xBFE will not be re-mapped to 0xFFE. In either case, only QoS ID 0xFFE will be added to the list of used QoS IDs.

To enable the dynamic addition, deletion, or change in rate-limit values of a policy-map, enter the following command.

```
device(config)# ip rate-limit arp policy-map policy-map-name
```

This command takes effect automatically, without unbinding and rebinding the ARP RL policy. If the ARP Rate Limit policy specifies an undefined policy-map, rate limit values are initialized to line-rate values. Dynamic enabling and disabling of priority based rate limiting on a global basis takes effect automatically for the ARP RL policy.

NOTE

ARP packets destined for the CPU will be not be rate-limited by interface-level Layer 2 RL-ACLs. To rate-limit switched ARP packets using interface-level Layer 2 ACLs, you must define an explicit ACL filter with an "etype arp" option, as shown in the following example:

To define an explicit ACL filter, enter commands similar to the following.

```
device(config)# access-list 410 permit any any any etype arp
device(config)# int eth 4/1
device(config-if-e10000-4/1)# rate-limit in access-gr 410 policy-map view
```

NOTE

Since ARP packets are broadcast packets, ARP packets are switched by default within a VLAN by the CPU. Thus to rate-limit switched ARP packets using interface-level Layer 2 ACLs, you must also configure vlan-cpu-protection.

# Changing the ARP aging period

When the Extreme device places an entry in the ARP cache, the Extreme device also starts an aging timer for the entry. The aging timer ensures that the ARP cache does not retain learned entries that are no longer valid. An entry can become invalid when the device with the MAC address of the entry is no longer on the network. The underlying MAC aging out causes deletion of the corresponding ARP entries.

The ARP age affects dynamic (learned) entries only, not static entries. The default ARP age is ten minutes. On the Extreme device, you can change the ARP age to a value from 0 through 240 minutes. If you set the ARP age to zero, aging is disabled and entries do not age out.

To globally change the ARP aging parameter to 20 minutes, enter the following command.

```
device(config)# ip arp-age 20
```

**Syntax: [no] ip arp-age num**

The *num* parameter specifies the number of minutes and can be from 0 through 240. The default is 10. If you specify 0, aging is disabled.

To override the globally configured IP ARP age on an individual interface, enter a command such as the following at the interface configuration level.

```
device(config-if-e1000-1/1)# ip arp-age 30
```

# Enabling proxy ARP

Proxy ARP allows a device to answer ARP requests from devices on one network on behalf of devices in another network.

Since ARP requests are MAC-layer broadcasts, they reach only the devices that are directly connected to the sender of the ARP request. Thus, ARP requests do not cross routers.

For example, if Proxy ARP is enabled on a device connected to two subnets, 10.10.10.0/24 and 10.20.20.0/24, the device can respond to an ARP request from 10.10.10.69 for the MAC address of the device with IP address 10.20.20.69. In standard ARP, a request from a device in the 10.10.10.0/24 subnet cannot reach a device in the 10.20.20.0 subnet if the subnets are on different network cables, and thus is not answered.

NOTE
An ARP request from one subnet can reach another subnet when both subnets are on the same physical segment (Ethernet cable), since MAC-layer broadcasts reach all the devices on the segment.

Proxy ARP is disabled by default.

To enable IP proxy ARP, enter the following command.

```
device(config)# ip proxy-arp
```

To again disable IP proxy ARP, enter the following command.

```
device(config)# no ip proxy-arp
```

# Enabling local proxy ARP

Under some Layer-2 configurations such as uplink-switch or private VLAN, broadcast packets are not flooded to every port in a VLAN. In these configurations, an ARP request from one host may not reach another host. Enabling the local-proxy ARP feature on a port directs the device to reply on behalf of a target host if it exists. The ARP reply returned contains the device's mac address instead of the mac address of the target host. In this transaction, the traffic sent to the target host is Layer-3 forwarded rather than Layer-2 switched.

To enable local-proxy ARP, the global-level command **ip proxy-arp** must first be enabled as described in Enabling proxy ARP on page 29. After **ip proxy-arp** is enabled globally, local-proxy ARP can be enabled on a specified interface using the following command.

```
device# configure terminal
device(config)# interface ethernet 1/1
device(config-if-e1000-1/1)# ip local-proxy-arp
```

# Disabling gratuitous ARP requests for local proxy ARP

When local proxy ARP is configured under the IP interface, a device replies to ARP requests on behalf of the hosts inside the subnet using its own MAC address. In this configuration, when a host comes up, the host tries to ping its own IP address to make sure there is no duplicated IP address by issuing a gratuitous ARP request (sender address equals target address) to its own IP address. The device will reply to this request because it is required under the local proxy ARP configuration. When the host receives the ARP reply, the host incorrectly assumes that there is another host using the same IP address.

By default, the device drops all ARP packets sent from its own interface. When the **ignore-gratuitous-arp** parameter is turned on, the device will not reply to a gratuitous ARP request even if the target protocol address matches the configured interface IP address.

To enable the **ignore-gratuitous-arp** parameter when the **ip local-proxy-arp** command is turned on, enter the following command.

```
device(config-if-e1000-1/6)# ip local-proxy-arp ignore-gratuitous-arp
```

To disable only the **ignore-gratuitous-arp** parameter when local proxy ARP is configured, enter the following command.

```
device(config-if-e1000-1/6)# no ip local-proxy-arp ignore-gratuitous-arp
```

To disable both **local-proxy-arp** and the **ignore-gratuitous-arp**, enter the following command.

```
device(config-if-e1000-1/6)# no ip-local-proxy-arp
```

# Static ARP entries

Static ARP entries are added to the ARP cache when you configure the entries.

Static ARP entries are useful when you want to pre-configure an entry for a device that is not connected to the NetIron OS device or when you want to prevent a particular entry from aging out. The software removes a dynamic entry from the ARP cache if the ARP aging interval expires before the entry is refreshed. Static entries do not age out, regardless of whether the device receives an ARP request from the device that has the entry's address.

NOTE
For task details, refer to Configuring static ARP on a VLAN and port on page 35.

# Changing the ARP timer

When an entry is initially added to the ARP table, it is listed as "Pending." When it is in this state, a series of ARP requests are made to determine if it is a valid entry. If the first attempt succeeds, the status of the entry is changed to "dynamic." It is then subject to the normal rules for dynamic entries. If three attempts fail, the entry is removed from the table.

The ARP timer determines the amount of time that elapses after the ARP request is sent before determining that the request has failed.The **arp-timer** command allows you change the length of the ARP timer as shown in the following.

```
device(config)# ip arp-timer 12
```

**Syntax: [no] ip arp-timer timer-value**

The *timer-value* variable has now been changed so that you are able to enter a value between 1 and 500. Each increment represents 100 ms. Consequently, the minimum value of 1 equals 100 ms.

The default value is 10 which equals 1 sec.

This value can be used to adjust how frequently an ARP request is sent out for a pending ARP entry.

# Changing the ARP pending retry timer

The ARP Pending Retry Timer for this device will send out three ARP request packets for the configured period until ARP is resolved to prevent large amounts of ARP requests from flooding the network during network host scanning activity. The ARP Pending Retry Timer is configurable depending upon the requirements of your system configurations.

The **arp-pending-retry-timer** command allows you to change the length of the ARP pending retry timer as shown in the following.

```
device(config)# ip arp-pending-retry-timer 120
```

**Syntax: [no] ip arp-pending-retry-timer timer-value**

The *timer-value* variable is a value between 10 to 3600 seconds. The default value is 60 seconds.

# Generating syslog notification for differing Ethernet source MAC and ARP sender MAC addresses

The NetIron OS devices generate a syslog notification whenever there is a mismatch between the Layer 2 header source MAC address and the ARP sender MAC address.

This syslog notification is supported on the XMR Series, MLX Series, CER 2000 Series, and CES 2000 Series platforms.

**Configuration step**

Enter the **logging enable mac-mismatch-detection** command for syslog notification due to MAC address mismatch.

The syslog message helps you identify the root cause for the traffic outage scenario and you can proceed with the static MAC address workaround in MCT by configuring the static MAC address in the CCEP port. The following syslog message is displayed when there is a MAC address mismatch.

```
SYSLOG: <14>Dec 16 05:53:23 MLX_1 MAC_MISMATCH_DETECTION: ARP pkt received with diff eth source MAC
and diff ARP sender MAC. Eth src MAC: 0024.3892.4c02 ARP sender MAC: 0034.2867.2c01.
```

# Displaying ARP entries

You can display the ARP cache and static ARP tables.

The ARP cache contains entries for devices attached to the NetIron OS device. The static ARP table contains user-configured ARP entries. An entry in the static ARP table enters the ARP cache when the entry's interface comes up.

## *Displaying the ARP cache*

To display the contents of the ARP cache, enter the following command at any CLI level.

```
device# show arp
Total number of ARP entries: 5
     IP Address          MAC Address         Type        Age     Port
1     10.95.6.102        0800.5afc.ea21      Dynamic     0         6
2     10.95.6.18         00a0.24d2.04ed      Dynamic     3         6
3     10.95.6.54         00a0.24ab.cd2b      Dynamic     0         6
4     10.95.6.101        0800.207c.a7fa      Dynamic     0         6
5     10.95.6.211        00c0.2638.ac9c      Dynamic     0         6
6     10.30.30.15        none                Pending     0         v1
```

## *Displaying the static ARP table*

To display the static ARP table, enter the following command at any CLI level.

```
device# show ip static-arp
Total no. of entries: 4
  Index   IP Address          MAC Address         Port      VLAN  ESI
  1       10.1.1.1             0001.0001.0001      1/1
  2       10.6.6.2             0002.0002.0002      1/2
  3       10.6.6.7             1111.1111.1111      2/1...
          Ports  : ethe 2/1 to 2/7 ethe 3/1 to 3/2
  4       10.7.7.7             0100.5e42.7f40      3/3
```

This example shows four static entries, one of which is multi-port. Multi-port static ARP entries are supported only on the XMR Series and MLX Series devices. Note that for multi-port entries the Port column shows a single port number followed by an ellipsis; the full list of ports associated with that ARP entry is displayed on the following line.

# Dynamic ARP inspection

NOTE
This feature is supported on Layer 2 and Layer 3 code.

Dynamic ARP Inspection (DAI) enables the device to intercept and examine all ARP request and response packets in a subnet and discard those packets with invalid IP to MAC address bindings. DAI can prevent common man-in-the-middle (MiM) attacks such as ARP cache poisoning, and disallow mis-configuration of client IP addresses.

## ARP poisoning

ARP provides IP communication within a Layer 2 broadcast domain by mapping an IP address to a MAC address. Before a host can talk to another host, it must map the IP address to a MAC address first. If the host does not have the mapping in its DAI table, it creates an ARP request to resolve the mapping. All computers on the subnet will receive and process the ARP requests, and the host whose IP address matches the IP address in the request will send an ARP reply.

An ARP poisoning attack can target hosts, switches, and routers connected to the Layer 2 network by poisoning the ARP caches of systems connected to the subnet and by intercepting traffic intended for other hosts on the subnet. For instance, a malicious host can reply to an ARP request with its own MAC address, thereby causing other hosts on the same subnet to store this information in their DAI tables or replace the existing ARP entry. Furthermore, a host can send gratuitous replies without having received any ARP requests. A malicious host can also send out ARP packets claiming to have an IP address that actually belongs to another host (e.g. the default router). After the attack, all traffic from the device under attack flows through the attacker's computer and then to the router, switch, or host.

## How DAI works

DAI allows only valid ARP requests and responses to be forwarded.

A device on which ARP Inspection is configured does the following:

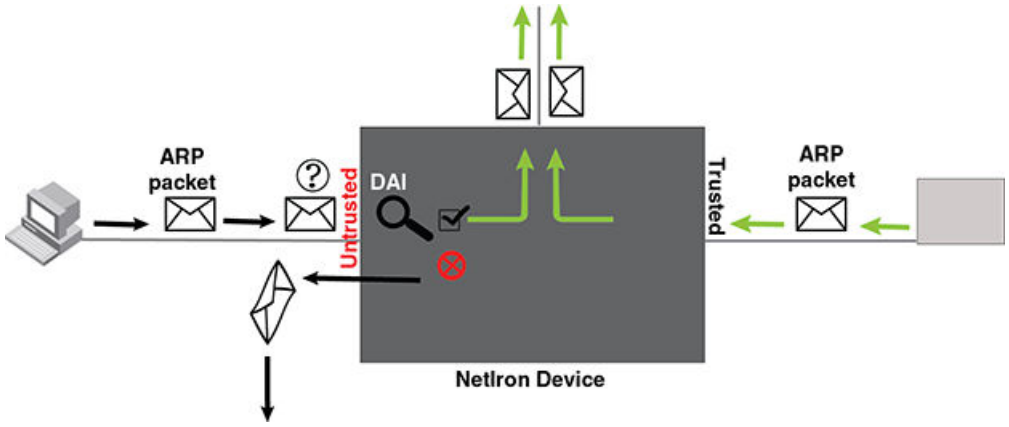
- Intercepts ARP packets received by the system CPU
- Inspects all ARP requests and responses received on untrusted ports
- Verifies that each of the intercepted packets has a valid IP-to-MAC address binding before updating the ARP table, or before forwarding the packet to the appropriate destination
- Drops invalid ARP packets

When you enable ARP Inspection on a VLAN, by default, all member ports are untrusted. You must manually configure trusted ports. In a typical network configuration, ports connected to host ports are untrusted. You configure ports connected to other switches or routers as trusted.

DAI inspects ARP packets received on untrusted ports, as shown in Figure 2. DAI carries out the inspection based on IP-to-MAC address bindings stored in a trusted binding database. For the Extreme device, the binding database is the ARP table, which supports DAI, DHCP snooping, and IP Source Guard. To inspect an ARP request packet, DAI checks the source IP and source MAC address against the ARP table. For an ARP reply packet, DAI checks the source IP, source MAC, destination IP, and destination MAC addresses. DAI forwards the valid packets and discards those with invalid IP-to-MAC address bindings.

When ARP packets reach a trusted port, DAI lets them through, as shown in Figure 2.

FIGURE 2 Dynamic ARP inspection at work




## *ARP entries*

DAI uses the IP/MAC mappings in the ARP table to validate ARP packets received on untrusted ports. ARP entries in the ARP table derive from the following:

- **ARP Inspection** - statically configured VRF+VLAN +IP/MAC mapping.
- **ARP** - statically configured VRF+IP/MAC/port mapping.
- **DHCP-Snooping ARP** - information collected from snooping DHCP packets when DHCP snooping is enabled on VLANs.

# Configuring DAI

NOTE
An index number is no longer needed to configure static ARP entries.

Follow the steps listed below to configure DAI.

1. Configure inspection of ARP entries for hosts on untrusted ports. Enable ARP Inspection on a VLAN to inspect ARP packets.
2. Configure the trust settings of the VLAN members. ARP packets received on *trusted* ports bypass the DAI validation process. ARP packets received on untrusted ports go through the DAI validation process.
3. Enable DHCP snooping to populate the DHCP snooping IP-to-MAC binding database. **Refer to** DHCP binding database on page 382 **for more information.**

   The following shows the default settings of ARP Inspection.

| Feature | Default |
|---|---|
| Dynamic ARP Inspection | Disabled |
| Trust setting for ports | Untrusted |

## *Enabling dynamic ARP inspection on a VLAN*

ARP and Dynamic inspection ARP entries need to be configured for hosts on untrusted ports. Otherwise, when Dynamic ARP Inspection checks ARP packets from these hosts against entries in the ARP table, it will not find any entries for them, and the device will not allow and learn ARP from an untrusted host.

Dynamic ARP Inspection is disabled by default. To enable Dynamic ARP Inspection on an existing VLAN or a range of VLANs, enter the following command.

```
device(config)# ip arp-inspection vlan 18 to 20
```

The command enables Dynamic ARP Inspection on VLAN 18 through VLAN 20. ARP packets from untrusted ports in VLAN 18 through VLAN 20 will undergo Dynamic ARP Inspection.

**Syntax: [no] ip-arp inspection vlan vlan_id to vlan_id**

The *vlan_id* variable specifies the ID of a configured VLAN or VLAN range. Valid VLAN ranges are 1-4090.

## *Configuring static ARP on a VLAN and port*

In the Extreme device configuration, the DHCP binding database is integrated with the ARP Inspection table. The ARP inspection table stores the DAI IP/MAC binding information, which is used to build the IP source guard ACL. The **static arp** command allows you to configure both the vlan id and port parameters on a layer 2 interface.

To configure a static arp entry for a vlan id, enter the following command.

```
device(config)# arp 10.1.0.2 aabb.cc00.0100 vlan 10
```

## *Enabling trust on a port*

The default trust setting for a port is untrusted. For ports that are connected to host ports, leave their trust settings as untrusted.

To enable trust on a port, enter commands such as the following.

```
device(config)# interface ethernet 1/4
device(config-if-e10000-1/4)# arp-inspection-trust
```

The commands change the CLI to the interface configuration level of port 1/4 and set the trust setting of port 1/4 to trusted.

**Syntax: [no] arp-inspection-trust**

## *Creating ARP entries*

Static entries are useful in cases where you want to pre-configure an entry for a device that is not connected to the Extreme device, or you want to prevent a particular entry from aging out. The software removes a dynamic entry from the ARP cache if the ARP aging interval expires before the entry is refreshed. Static entries do not age out, regardless of whether the device receives an ARP request from the device that has the entry's address.

To create a static ARP entry for a static MAC entry, enter a command such as the following.

```
device(config)# arp 10.53.4.2 1245.7654.2348 vlan 10
```

The command adds a static ARP entry that maps IP address 10.53.4.2 to MAC address 1245.7654.2348. The entry is for a MAC address connected to VLAN 10 of the Extreme device.

**Syntax: [no] arp ip-addr mac-addr** [ **ethernet slot/port** | **vlan vlan_id** ]

The *ip-addr* parameter specifies the IP address of the device that has the MAC address of the entry.

The *mac-addr* parameter specifies the MAC address of the entry.

The **ethernet***slot/port* command specifies the port number attached to the device that has the MAC address of the entry.

The **vlan** *vlan_id* variable specifies the ID of a configured VLAN or VLAN range. Valid VLAN ranges are 1-4090.

## *Creating multi-port ARP entries*

On the device devices, multiple ports belonging to the same VE can be assigned to a single static ARP.

NOTE
The multi-port ARP feature can be used in a pure Layer 3 forwarding environment to forward IPv4 traffic to multiple ports and should not be used in conjunction with Multi-port static MAC.

To create a multi-port static ARP entry, enter a command such as the following.

```
device(config)# arp 10.53.4.2 1245.7654.2348 multi-ports ethernet 2/1 to 2/7 ethernet 3/1 to 3/2
```

The command above adds a static ARP entry that maps IP address 10.53.4.2 to MAC address 1245.7654.2348. If all conditions are met and the multi-port static ARP entry is instantiated in the dynamic ARP table, then packets with a destination IP address of 10.53.4.2 will be sent out on Ethernet ports 2/1-2/7 and 3/1-3/2.

**Syntax: [no] arp ip address mac address** [ **port** | **multi-ports ethernet** [ **slot1/port1** | [ **slot1/port1 to slot1/port** ] **.. ethernet** [ **slot/port to slot/port** ] ]

The *ip-address* parameter specifies the IP address of the device that has the MAC address of the entry.

The *mac-address* parameter specifies the MAC address of the entry.

The **ethernet***slot/port* command specifies the port number attached to the device that has the MAC address of the entry. (The **ethernet** keyword is repeated before each individual port or range of ports to be included in the multi-port ARP entry.)

### Consideration for VRF multi-port ARP entries

The configuration command above creates a static ARP entry associated with the default VRF. To configure the multi-port ARP for a non-default VRF, first enter the configuration mode for the non-default VRF, then enter address family command mode using commands such as the following.

```
device(config)# vrf test
device(config-vrf-test)# address-family ipv4
device(config-vrf-test-ipv4)# arp 10.6.6.7 0001.0001.0001 multi-ports ethernet 2/1 to 2/7
device(config-vrf-test-ipv4)# ethernet 3/2 to 3/2
device(config-vrf-test-ipv4)# exit-address-family
device(config-vrf-test)# exit vrf
```

The above commands create a multi-port ARP entry associated with a non-default VRF called "test."

NOTE
This feature is supported on both the XMR Series, MLX Series and CER 2000 Series, CES 2000 Series series platforms.

### Instantiation in the ARP table

NOTE
Configuring a multi-port static ARP entry does not automatically create a dynamic ARP entry.

NOTE
The multi-port ARP feature can be used in a pure Layer 3 forwarding environment to forward IPv4 traffic to multiple ports and should not be used in conjunction with Multi-port static MAC.

The following four conditions must be met in order for a user-created multi-port static ARP entry to be instantiated in the dynamic ARP table:

1. All the ports configured in the multi-port static ARP entry need to belong to the same VE.

2. The IP address of the multi-port static ARP entry needs to match the subnet of the VE to which the ports belong, and it must be in the same VRF.
3. At least one of the ports in the configured port list needs to be up.
4. MPLS uplink must not be configured on the VE that subnets the static ARP IP address.

   If these four conditions are met, a conflict check is performed before adding the static ARP entry to the dynamic ARP table. If a dynamic entry already exists with the same IP address and VRF, the static ARP will override the dynamic entry and packets will be forwarded to the FID for this dynamic ARP entry.

   Changes in these conditions (VE port membership changes, port up/down status changes, etc.) can trigger reevaluation of the static ARP and may result in the entry being added to or removed from the ARP table.

### Supported applications

- **PBR** PBR supports use of a multi-port static ARP entry as an IP next hop.
- **Trunk ports** Primary trunk ports can be configured in multi-port static ARPs. If a secondary trunk port is included in a multi-port ARP entry, however, the trunk will not be deployed.
- **ARP inspection** ARP inspection is performed for multi-port static ARPs the same as for normal static ARP entries.

### Unsupported applications

- **IP tunnel** If an IP tunnel's next hop is resolved to a multi-port static ARP entry, the tunnel will not be brought up.
- **MPLS next-hop** Configuring an MPLS uplink on the VE interface associated with a multi-port static ARP will prevent instantiation of the ARP.
- **MCT** The ICL ports in MCT and clients are not supported by multi-port static ARP and MAC.
- **PB/PBB** The non-default port types are not supported by multi-port static ARP and MAC on PB/PBB.

## *Creating a floating static ARP entry*

You can create a static ARP entry without port assignments.

When a floating static ARP entry (Static ARP Inspection entry without port defined) is added to ARP Inspection table, the mapping is checked against the current static ARP table. If ARP entry with a matching IP but mismatch MAC is found, it will be deleted and a re-arp on the IP will be issued.

When an ARP entry is deleted from ARP Inspection table, the corresponding entry in the static ARP table will also be deleted.

To create a floating static ARP entry for a static MAC entry, enter a command such as the following.

```
device(config)# arp 10.53.4.2 1245.7654.2348
```

The command adds a floating static ARP entry that maps IP address 10.53.4.2 to MAC address 1245.7654.2348.

**Syntax: [no] arp ip-addr mac-addr** [ **ethernet portnum** | **vlan vlan_id** ]

The *ip-addr* parameter specifies the IP address of the device that has the MAC address of the entry.

The *mac-addr* parameter specifies the MAC address of the entry.

The **ethernet** *portnum* parameter specifies the port number attached to the device that has the MAC address of the entry, and is only valid for original static ARP entries.

The **vlan** *vlan_id* parameter specifies the ID of a configured VLAN.

### *Configuring a Virtual Routing Instance (VRF)*

To configure a virtual routing instance (VRF), enter a command such as the following.

```
device(config)# vrf vpn1
```

**Syntax: [no] vrf vrf-name**

The **vrf** parameter specifies the virtual routing instance (VRF) specified by the variable *vrf-name*.

### *Adding an ARP entry for a VRF*

IP Addresses can be uniquely determined by VRF. The VLAN number is not needed because the VLAN information is obtained through the ARP protocol. To define an ARP inspection entry for a specific VRF, enter commands such as the following.

```
device(config)# vrf vpn1
device(config-vrf-vpn1))#arp 10.53.4.2 1245.7654.2348 e 3/5
```

This command creates an ARP entry for vrf with IP address 10.53.4.2 and MAC address of 1245.7654.2348 on ethernet 3/5.

**Syntax: [no] arp ip-addr mac-addr** [ **ethernet slot/port** ]

The *vrf-name* parameter specifies the VRF you are configuring a static ARP entry for.

The *ip-addr* parameter specifies the IP address of the device that has the MAC address of the entry.

The *mac-addr* parameter specifies the MAC address of the entry.

The e**thernet** *slot/port* variable specifies the port number attached to the device that has the MAC address of the entry.

## Displaying ARP inspection information

You can display ARP inspection information using the **show ip arp-inspection** and the **show ip static-arp** commands as shown in the following.

### *Displaying ARP inspection status and ports*

To display the ARP inspection status for a VLAN and the trusted/untrusted ports in the VLAN, enter the following command.

```
device# show ip arp-inspection
ARP inspected VLANs:
1000
ARP inspection trusted ports:
ethe 2/1
```

**Syntax: show ip arp-inspection** [ **vlan vlan_id** ]

The **vlan** *vlan_id* parameter specifies the ID of a configured VLAN.

### *Displaying ARP inspection statistics*

You can use the **show ip arp-statistics** command to display ARP inspection counters for all ports on the device, as shown in the following.

```
device# show ip arp-inspection-statistics
Module 1:
Port     Arp Packets Captured      Arp Packets Failed Inspection
1/1         0                        0
1/2         0                        0
1/3         0                        0
1/4         0                        0
```

```
1/5         0                       0
1/6         0                       0
1/7         0                       0
1/8         0                       0
1/9         0                       0
1/10        0                       0
1/11        0                       0
1/12        0                       0
1/13        0                       0
1/14        0                       0
1/15        0                       0
1/16        0                       0
1/17        0                       0
1/18        0                       0
1/19        0                       0
1/20        0                       0
Module 3:
Port     Arp Packets Captured      Arp Packets Failed Inspection
3/1         0                       0
3/2         0                       0
3/3         0                       0
3/4         690                     153
```

Specifying a port number with the **show ip arp-statistics** command displays the statistics for that port only, along with details of the last five ARP packets that failed inspection, as shown in the following.

```
device# show ip arp-inspection-statistics ethernet 3/4
Arp packets captured: 695
Arp packets failed inspection: 158
Last 5 packets failed inspection:
Time                 Op  Target IP Target Mac     Source IP Source Mac     Vlan
2007-10-24  18:53:28 2   10.1.1.1 000c.dbe2.9353 10.1.1.2 0000.0900.0005 1
2007-10-24  18:53:29 2   10.1.1.1 000c.dbe2.9353 10.1.1.2 0000.0900.0005 1
2007-10-24  18:53:30 2   10.1.1.1 000c.dbe2.9353 10.1.1.2 0000.0900.0005 1
2007-10-24  18:53:32 2   10.1.1.1 000c.dbe2.9353 10.1.1.2 0000.0900.0005 1
2007-10-24  18:53:33 2   10.1.1.1 000c.dbe2.9353 10.1.1.2 0000.0900.0005 1
```

**Syntax: show ip arp-inspection-statistics** [ **slot** *slot-num* | **ethernet** *slot/port* ]

The **slot** option allows you to limit the display of ARP inspection statistics to the Ethernet interface module in the slot specified by the *slot-num* variable.

The **ethernet** option allows you to limit the display of ARP inspection statistics to the port specified by the *slot/port* variable. It also provides details of the last five ARP packets received by the specified port that failed inspection.

This display shows the following information.

**TABLE 2** Show ip arp-inspection-statistics

| This field... | Displays... |
|---|---|
| Port | The slot/port number. |
| Arp packets captured | The number of ARP packets captured for the specified port. |
| Arp packets failed inspection | The number of captured ARP packets that failed inspection for the specified port. |
| **The following fields apply to the first five packets that failed inspection on the specified port** . | |
| Time | The date and time that the packet was received on the port. |
| Op | The ARP operation mode. |
| Target IP | The destination IP address of the ARP rejected packet. |
| Target MAC | The destination MAC address of the ARP rejected packet. |
| Source IP | The source IP address of the ARP rejected packet. |
| Source MAC | The source MAC address of the ARP rejected packet. |
| VLAN | The VLAN number of the ARP rejected packet. |

### *Displaying the ARP table*

To display the ARP Inspection table, enter the following command.

```
device# show ip static-arp
Total no. of entries: 4
  Index   IP Address          MAC Address         Port      VLAN  ESI
  1       10.1.1.1             0001.0001.0001      1/1
  2       10.6.6.2             0002.0002.0002      1/2
  3       10.6.6.7             1111.1111.1111      2/1...
          Ports  : ethe 2/1 to 2/7 ethe 3/1 to 3/2
  4       10.7.7.7             0100.5e42.7f40      3/3
```

The command displays all ARP entries in the system.

The above output includes a multi-port static ARP entry.

**Syntax: show ip static-arp**

## Clearing ARP inspection counters

You can use the clear arp-inspection-statistics command to clear the ARP inspection statistics counters for all ports on the device or for a specified module or port as shown in the following.

```
clear arp-inspection-statistics ethernet 3/1
```

**Syntax: clear ip arp-inspection-statistics** [ **slot slot-num** | **ethernet slot/port** ]

The **slot** option allows you to clear ARP inspection statistics for a single Ethernet interface module in a slot specified by the *slot-num* variable.

The **ethernet** option allows you to clear ARP inspection statistics for a single port specified by the *slot/port* variable.

# ARP Guard

Internet exchange points (IXPs) are designed based on a flat Layer 2 topology to provide any-to-any connectivity among BGP routers from different ISPs, CSPs, and enterprises connected to it.

As an IP host, each BGP peering router makes use of ARP protocol to determine the MAC address of its BGP peers. Since ARP is not a secure protocol, any BGP router can reply to the ARP request for any IP address, and any BGP router can generate gratuitous ARP to claim ownership of any IP address in the router.

When the network administrator of a BGP border router connecting to the IXP incorrectly configures the router IP address or unknowingly turns on the proxy-ARP feature on the interface facing the IXP, valid traffic may be sent to the wrong destination until the ARP cache expires on the other routers.

The ARP Guard feature uses a set of ACL-like commands to build a table of allowed IP addresses on the link. As a result, when an ARP reply, either due to gratuitous ARP or in response to a normal ARP request, is received on a port facing the BGP router, the ARP reply is inspected based on the IP address parameters configured for the **permit** command. ARP packets that do not match the entries in the ACL are dropped. Matching ARP packets are forwarded.

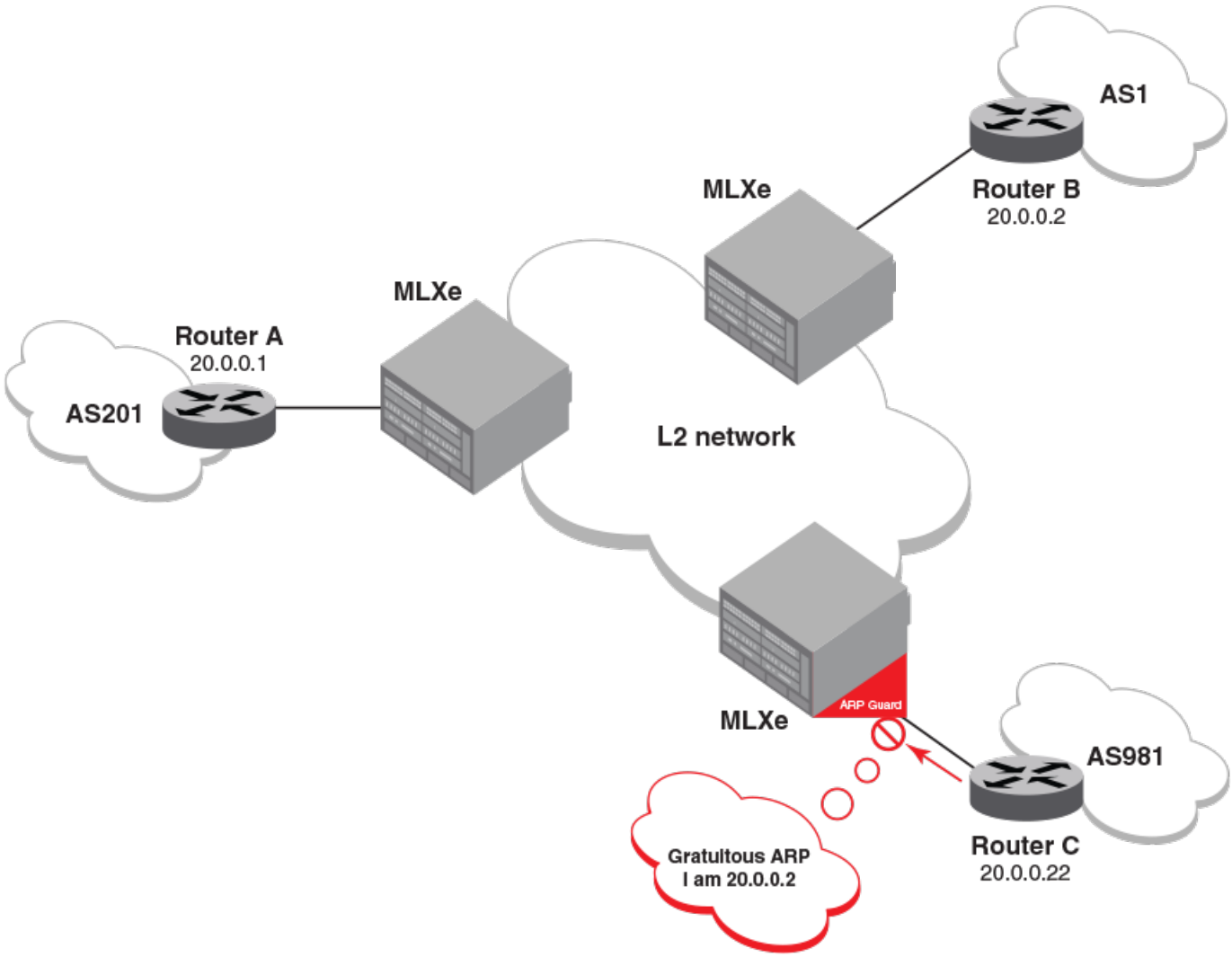
# ARP Guard use-case scenarios

The following scenarios are handled by ARP Guard.

## *Hijacking due to wrong configuration of IP address*

The network diagram explains how ARP guard functions when IP is incorrectly configured on the network.

**FIGURE 3** Layer 2 network with ARP Guard




In the previous figure, assume that the correct IP address for router C is 20.0.0.22, but the network administrator has mis-configured the router IP address as 20.0.0.2 (which happens to be the IP address of router B from AS 1).

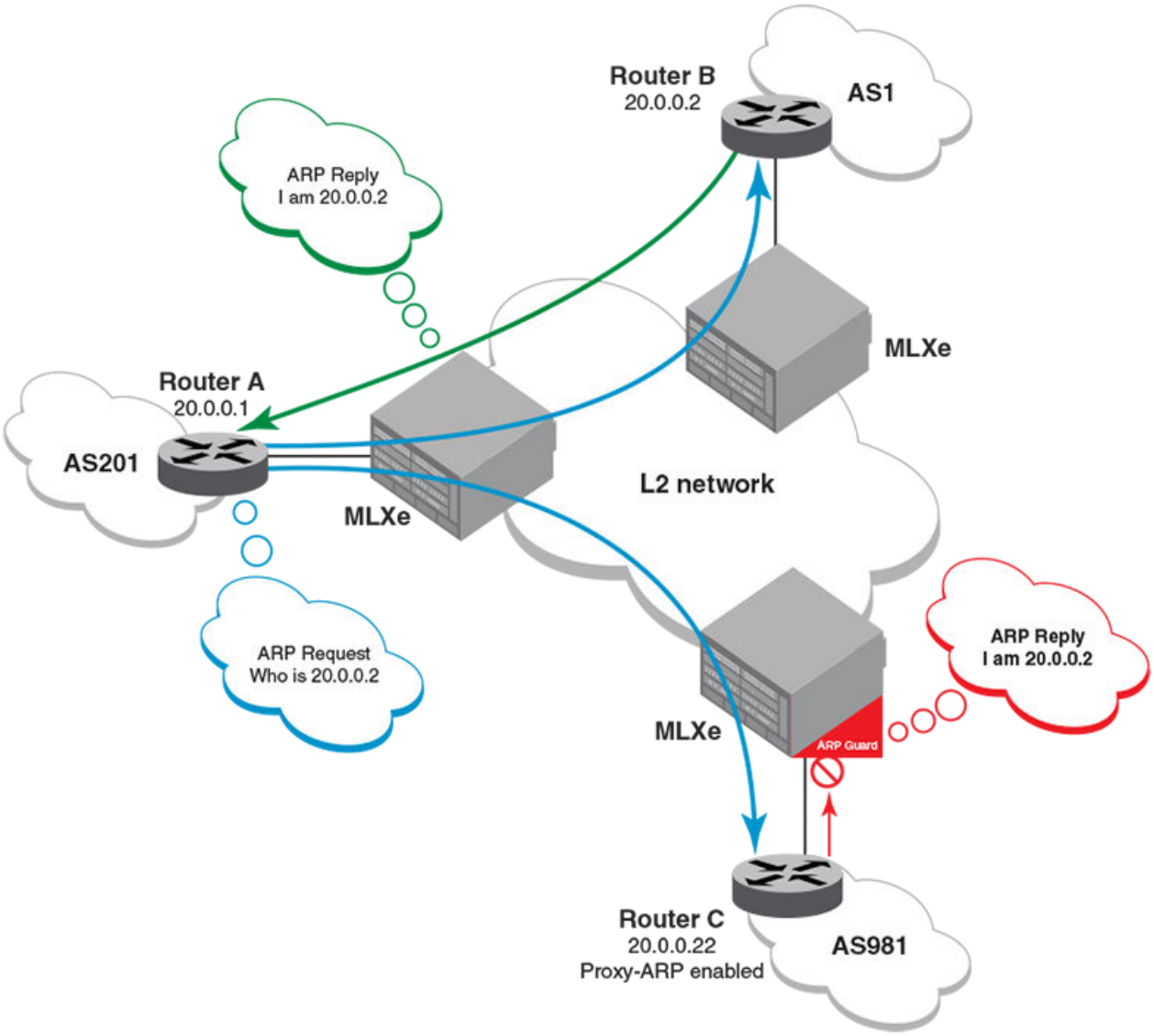
After entering the wrong IP address or after the link from router C to the MLXe comes up, router C sends out gratuitous ARP packets to claim ownership of IP address 20.0.0.2. Without the ARP Guard enabled, router A may update its ARP entry for 20.0.0.2 with the MAC address of router C, which causes traffic originally destined to router B to be black-holed on router C. When ARP Guard is enabled on MLXe devices, the devices are configured to allow gratuitous ARP packets for IP address 20.0.0.22 only from the link connecting to

router C to enter the VLAN or VPLS Layer 2 network. When gratuitous ARP packets for IP address 20.0.0.2 are received from router C, they are dropped, and a message may be logged.

## *ARP hijacking with proxy-ARP*

The network diagram illustrates how ARP guard functions when proxy-ARP is enabled on the network.

**FIGURE 4** Layer 2 network with ARP Guard




In the previous figure, assume that the network administrator for router C configures its IP address correctly as 20.0.0.22 but unknowingly turns on proxy-ARP. When router A tries to resolve the MAC address for the IP address 20.0.0.2 on router B through ARP, router C sends an ARP reply claiming ownership of the IP address.

When ARP Guard is not enabled, router A may mistake the MAC address of router C as the MAC address for the IP address 20.0.0.2 of router B, causing traffic originally destined to router B to be black-holed on router C. When ARP Guard is enabled on MLXe devices, the

devices are configured to allow ARP replies for 20.0.0.22 only from the link connecting to router C to enter the VLAN or VPLS L2 network. ARP replies for IP address 20.0.0.2 from router C are dropped, and a message may be logged.

# Configuration considerations and limitations for ARP Guard

ARP Guard configuration issues are as follows.

- ARP Guard configuration is limited to "permit" options for the following parameters:
  - VLAN ID
  - Source MAC address
  - Source IP address
- If a Layer 2 ACL is used to specify the **arp-guard** keyword to shunt ARP packets to the CPU, incoming traffic cannot be rate limited. Because available CPU may be constrained when multiple Layer 2 ACLs are applied to the same interface, any increase in the rate of ARP traffic to the CPU may create high CPU conditions.
- Extreme recommends that you use a Layer 2 ACL to limit the rate of ARP traffic on an interface where ARP Guard is enabled.
- ARP Guard is supported only on physical interfaces.
- ARP Guard and IPv4 cannot be configured on the same physical interface.
- ARP Guard cannot be configured on a route-only interface.
- ARP Guard statistics are not retained after a switchover.
- When ARP Guard is enabled on a LAG and a member port is removed from the LAG, ARP Guard properties are retained on the port that has been removed from the LAG.
- The **show running-config** command does not display ARP Guard configurations with default conditions.
- When ARP Guard is enabled on any interface, the global route-only option cannot be configured on a device.
- ARP Guard **show** commands are supported only for the management processor (MP).

# Configuring ARP guard

The following commands configure ARP guard on a NetIron device.

**Enabling filtering of incoming ARP packets to LP-CPU:**

The following command enables the required L2 ACL rules to filter the incoming ARP packets to the CPU. In the following option, an additional key word "arp-guard" is supported in the existing L2 access-list command syntax. The user should specify this key word when creating the L2 ACL rules for filtering the ARP packets to CPU.

```
device(config)# access-list 400 permit any any any etype arp arp-guard
```

**Syntax: [no] access-list** *num* **permit** *src-mac / mask* | **any** *dest-mac/mask* | **any** *vlan-id* | **any etype arp arp-guard**

This configuration creates a standard Layer-2 ACL with an ID of 400.

*etype arp* : L2 ACL applied only for the ARP packets.

*arp-guard* : ARP-Guard will filter all the ARP packets to the LP-CPU.

**Bind Layer 2 ACL to an interface:**

Layer 2 ACL needs to be bound to an interface where ARP guard would be required. The below configuration will punt all incoming ARP packets to LP-CPU based on the L2-ACL rules provided.

```
device(config-if-e10000-1/1)# mac access-group 400 in
```

NOTE

The keyword "arp-guard" is mandatory for MLX Series and XMR Series series of device to handle the programming of the ARP guard ACLs in hardware in order to punt the received ARP packets to LP CPU for processing.
For CER 2000 Series and CES 2000 Series platforms, by default all ARP packets are trapped to CPU. Hence, the keyword "arp-guard" is not required to handle the programming of ARP guard ACLs in hardware. Also the binding of ACLs to the associated interface is not required in CER 2000 Series and CES 2000 Series devices.

**Creating ARP guard access-list table:**

The rules for filtering of ARP packets are done through ACL like commands in the global configuration mode, which are configured through the following commands. The **no** form of the command would disable that particular rule.

```
device(config)# arp-guard-access-list AS201
device(config-arp-guard-access-list-AS201)# permit 1.1.1.1
device(config-arp-guard-access-list-AS201)# permit 1.1.1.2 1111.1111.1111
device(config-arp-guard-access-list-AS201)# permit 1.1.1.3 any
device(config-arp-guard-access-list-AS201)# permit 10 1.1.1.4
device(config-arp-guard-access-list-AS201)# permit 10 1.1.1.4 1111.2222.3333
```

**Syntax: [no] arp-guard-access-list** *arp-guard-access-list*

**Syntax: [no] permit** *[vlan-id] [src-ip-addr] [src-mac-addr]* | **[any]**

**Parameters**

*arp-guard-access-list* specifies the name of the ARP guard access-list.

**permit** specifies the required set of rules for the associated ARP guard group.

**Binding of ARP guard to an interface:**

The following commands are used to enable ARP guard under the interface configuration mode. Using "log" option, would capture the log information of the dropped ARP packet such as "the name of the port", "vlan-id"(if any), "name of the ACL" which detected the violation, "MAC-address", and "IP address". The **no** form of the command would disable ARP guard.

```
device(config-if-e10000-1/1)# arp-guard AS201
device(config-if-e10000-1/1)# arp-guard AS201 log
device(config-if-e10000-1/1)# arp-guard AS201 log 20
```

**Syntax: [no] arp-guard** *arp-guard-access-list* **log** *number of violations to cache*

**Parameters**

**arp-guard** enables ARP guard in the interface configuration mode.

*arp-guard-access-list* specifies the name of the ARP guard access-list which contains the list of rules.

**log** option is used to log the information about the dropped packets.

*number of violations to cache* specifies the number of dropped packets to cache. Range is 5 to 32.

NOTE

If user does not specify number of violations, then by default; last 5 violated dropped packets will be printed on the active console at every default interval or configured interval.

**Modifying ARP guard rules:**

If user modifies an existing bound ARP-Guard access-list, then **apply-arp-guard** command should be used to apply the changed rules on the associated interfaces.

```
device(config-arp-guard-access-list-AS201)#apply-arp-guard
```

**Syntax: apply-arp-guard**

**apply-arp-guard** will program all the newly updated rules (if any) for that session (console/telnet/SSH) to all the associated ports.

NOTE
Invalid rules will be discarded in the following scenarios.

- When user jumps into different prompt from the ARP-Guard prompt.
- When user triggers “end”/”exit” from the ARP-Guard prompt.
- When the current session is closed (telnet/SSH).
- When the active MP switches over to the standby MP.

**Steps to configure ARP guard:**

**1.** Configure L2-ACL rules for ARP packet filtering to LP-CPU. The example below uses MAC ACLs, even RL ACLs can be used for the same.

```
device(config)#access-list 400 permit any any any etype arp arp-guard
device(config-if-e1000-1/24)#mac access-group 400 in
```

**2.** Configure arp-guard-access-list to specify the set of rules/filters for this ARP ACL.

```
device(config)#arp-guard-access-list AS201
device(config-arp-guard-access-list)#permit 20.0.0.2 0001.0002.0003
device(config-arp-guard-access-list)#exit
```

**3.** Apply the arp-guard-access-list on the interface using the arp-guard command as shown below.

```
device(config)#interface ethe 1/1
device(config-if)#arp-guard AS201 log
```

**Syslog Information**

If **log** option is specified in the **arp-guard** command, then a syslog message is generated to log the dropped ARP packet. The **arp-guard-syslog-timer** command can be used to modify the interval at which the syslogs need to be generated

Syslog message contains the following:

- Port ID
- arp-guard-group name
- VLAN-id (if any)
- MAC address and the IP address

All violations are noted down in Software and at the configured syslog interval for the ARP Guard entry the violations are logged.

Following are the Syslog message output display:

```
SYSLOG: <14>Mar 14 1905 22:37:21 MLX-Dist1 ARP_GUARD DROP LOG:10 Violations occurred on port=4/1 having
Access_Grp= AS201 Most recent 5 violations are:

SYSLOG: <14>Mar 14 1905 22:37:21 MLX-Dist1 ARP_GUARD DROP LOG:Violation occured at time Mar 14 22:37:20: on
Trunk port=4/1 having Access_Grp=AS201, for the incoming packet with MAC_ADDR=0000.5822.bf78
IP_ADDR=1.1.1.2 VLAN: 1

SYSLOG: <14>Mar 14 1905 22:37:21 MLX-Dist1 ARP_GUARD DROP LOG:Violation occured at time Mar 14 22:37:20: on
Trunk port=4/1 having Access_Grp= AS201, for the incoming packet with MAC_ADDR=0000.5823.0a9b
IP_ADDR=2.1.1.2 VLAN: 1

SYSLOG: <14>Mar 14 1905 22:37:21 MLX-Dist1 ARP_GUARD DROP LOG:Violation occured at time Mar 14 22:37:20: on
Trunk port=4/1 having Access_Grp= AS201, for the incoming packet with MAC_ADDR=0000.5822.bf78
IP_ADDR=1.1.1.2 VLAN: 1

SYSLOG: <14>Mar 14 1905 22:37:21 MLX-Dist1 ARP_GUARD DROP LOG:Violation occured at time Mar 14 22:37:20: on
Trunk port=4/1 having Access_Grp= AS201, for the incoming packet with MAC_ADDR=0000.5823.0a9b
IP_ADDR=2.1.1.2 VLAN: 1
```

# IP Addressing



## The IP packet flow

Figure 5 shows how an IP packet moves through this device.

**FIGURE 5** IP Packet flow

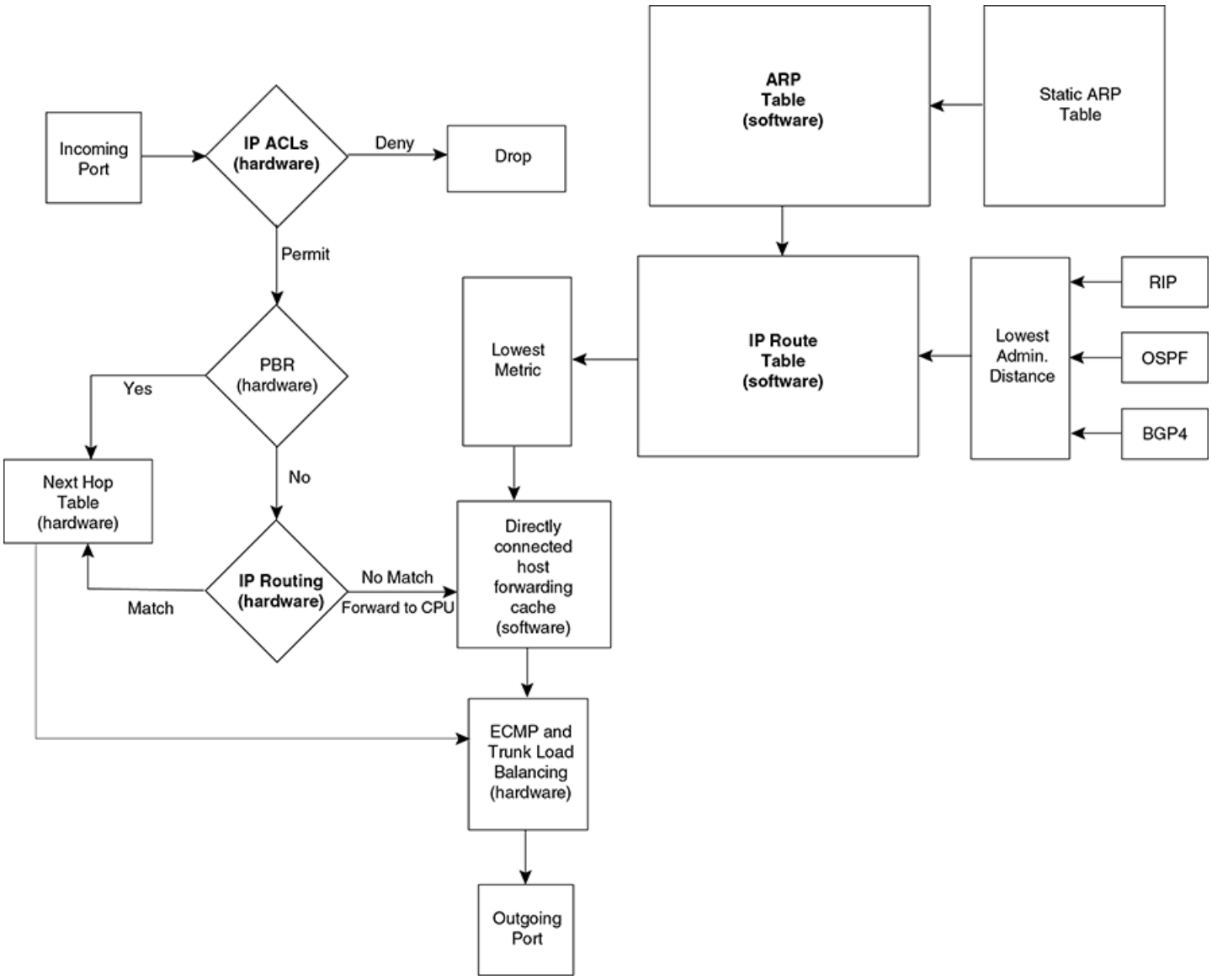



Figure 5 shows the following packet flow.

1. When the Extreme device receives an IP packet, the Extreme device checks for IP ACL filters on the receiving interface. If a deny filter on the interface denies the packet, the Extreme device discards the packet and performs no further processing. If logging is enabled for the filter, then the Extreme device generates a Syslog entry and SNMP trap message.
2. If the packet is not denied, the Extreme device checks for Policy Based Routing (PBR). If the packet matches a PBR policy applied on the incoming port, the PBR processing is performed and either drops the packet or forwards it to a port, based on the route map rules.

3. If the incoming packet does not match PBR rules, the Extreme device looks in the hardware IP routing table to perform IP routing. The hardware routing table is pre-loaded with the complete routing table, except for the directly connected host entries. Default and statically defined routes are also pre-loaded in the hardware routing table. If the incoming packet matches a route entry, the packet is routed according to the information provided in the route entry. The ECMP and LAG load balancing is done by the hardware, if needed, to select the outgoing port.
4. If there is no match in the IP routing table and a default route is not configured, the packet is dropped. For an IP packet whose destination IP address is to a directly connected host, the first packet is forwarded to the CPU. If the ARP is resolved and the host is reachable, the CPU creates a route entry in the hardware to route subsequent packets in hardware.

   The software enables you to display the ARP cache and static ARP table, the IP route table, the IP forwarding cache.

   You also can change the capacity of the following tables by changing the memory allocation for the table:

   - ARP cache table on page 49
   - Static ARP table on page 49
   - IP route table on page 50
   - IP forwarding cache on page 50

# ARP cache table

The Address Resolution Protocol (ARP) is supported on the Extreme device. Refer to Basic ARP configuration on page 27.

The ARP cache contains entries that map IP addresses to MAC addresses. Generally, the entries are for devices that are directly attached to the Extreme device.

An exception is an ARP entry for an interface-based static IP route that goes to a destination that is one or more router hops away. For this type of entry, the MAC address is either the destination device's MAC address or the MAC address of the router interface that answered an ARP request on behalf of the device, using proxy ARP.

The ARP cache can contain dynamic (learned) entries and static (user-configured) entries. The software places a dynamic entry in the ARP cache when the Extreme device learns a device's MAC address from an ARP request or ARP reply from the device.

The software can learn an entry when the Extreme device receives an ARP request from another IP forwarding device or an ARP reply.

: Dynamic entry

```
        IP Address          MAC Address         Type        Age     Port
1       10.95.6.102         0800.5afc.ea21      Dynamic     0         6
```

Each entry contains the destination device's IP address and MAC address.

# Static ARP table

In addition to the ARP cache, the Extreme device has a static ARP table.

Entries in the static ARP table are user-configured. You can add entries to the static ARP table regardless of whether the device the entry is for is connected to the Extreme device.

The software places an entry from the static ARP table into the ARP cache when the entry's interface comes up.

: Static ARP entry

```
    Index   IP Address          MAC Address         Port
    1       10.95.6.111         0800.093b.d210      1/1
```

Each entry lists the information you specified when you created the entry.

To display ARP entries, refer to Displaying ARP entries on page 32.

To configure other ARP parameters, refer to Basic ARP configuration on page 27.

To increase the size of the ARP cache and static ARP table, refer to the following:

- For dynamic entries, refer to the "Displaying and modifying default settings for system parameters". The ip-arp parameter controls the ARP cache size.

# IP route table

The IP route table contains paths to IP destinations.

The IP route table can receive the paths from the following sources:

- A directly-connected destination, which means there are no router hops to the destination
- A static IP route, which is a user-configured route
- A route learned through RIP
- A route learned through OSPF
- A route learned through IS-IS
- A route learned through BGP4

The IP route table contains the best path to a destination:

- When the software receives paths from more than one of the sources listed above, the software compares the administrative distance of each path and selects the path with the lowest administrative distance. The administrative distance is a protocol-independent value from 1 - 255.
- When the software receives two or more best paths from the same source and the paths have the same metric (cost), the software can load share traffic among the paths based on Layer 2, Layer 3 and TCP/UDP information.

IP route table

```
Destination         Gateway           Port     Cost   Type    Uptime
10.0.0.0/8          10.20.176.1       mgmt 1    1/1    S      11m59s
```

Each IP route table entry contains the destination's IP address and subnet mask and the IP address of the next-hop router interface to the destination. Each entry also indicates the port attached to the destination or the next-hop to the destination, the route's IP metric (cost), and the type. The type indicates how the IP route table received the route.

Consider the following:

- For learned routes, modify the ip-route parameter.
- For static routes, modify the ip-static-route parameter.

# IP forwarding cache

The Extreme device maintains a software cache table for fast processing of IP packets that are forwarded or generated by the CPU. The cache also contains forwarding information that is normally contained in the IP routing table. For example, the cache contains information on the physical outgoing port, priority, VLAN, and the type of cache entry. Also, cache entries have hardware information, which is useful for debugging and aging.

There are two types of IP cache entries.

1. Directly connected host entries - These entries are created when the CPU receives the first packet destined to a directly connected host. Host entries are set to age out after a certain period if no traffic is seen for that entry.

2. Network entries - These entries are created when a route table entry is created in software. These entries are not subjected to aging. A route table entry is created when routes are learned by routing protocols such as OSPF or when routes are statically configured.

   : IP forwarding cache

```
        IP Address       Next Hop         MAC              Type  Port  Vlan  Pri
1       192.168.1.11     DIRECT           0000.0000.0000   PU    n/a         0
```

   Each IP forwarding cache entry contains the IP address of the destination, and the IP address and MAC address of the next-hop router interface to the destination. If the destination is actually an interface configured on the Extreme device itself, as shown here, then next-hop information indicates this. The port through which the destination is reached is also listed, as well as the VLAN and Layer 4 QoS priority associated with the destination if applicable.

   To display the IP forwarding cache, refer to Displaying the forwarding cache on page 126.

## IP packet queuing

When the user wants to send a packet to a local host, the software looks up the IP in the ARP cache. If the address is found, it gets the MAC address, constructs an Ethernet header with the correct source or destination MAC addresses, and sends it.

If the address is not found in the table, ARP broadcasts a packet to every host on the Ethernet, except the one from which it received the packet. The packet contains the IP address for which an Ethernet address is sought. If a receiving host identifies the IP address as its own, it will send its Ethernet address back to the requesting host.

For management of IP packet queuing when a packet is received for a directly connected host when there is no MAC address available, the **ip drop-arp-pending-packets** command has been added to allow the packets in the CPU to be dropped.

To set all packets in the LP buffer to be dropped when ARP resolution is going on, enter a command such as the following:

```
device(confi
g)#ip drop-arp-pending-packets
```

**Syntax: [no] ip drop-arp-pending-packets**

Use the**no ip drop-arp-pending-packets command** to return to the default behavior of continue with pending IP packets while ARP resolution.

# Basic IP parameters and defaults

The following protocols are disabled by default:

- Route exchange protocols (RIP, OSPF, IS-IS, BGP4)
- Multicast protocols (IGMP, PIM-DM, PIM-SM)
- Router redundancy protocols (VRRP-E, VRRP, FSRP)

## Parameter changes in effect

Most IP parameters described in this chapter are dynamic. They take effect immediately, as soon as you enter the CLI command. You can verify that a dynamic change has taken effect by displaying the running configuration.

To display the running configuration, enter the **show running-config** command.

To save a configuration change permanently so that the change remains in effect following a system reset or software reload, save the configuration to the startup configuration file.

To change the memory allocation, you must reload the software after you save the changes to the startup configuration file.

# IP global parameters

The following table lists the IP global parameters, their default values, and where to find configuration information.

**TABLE 3** IP global parameters

| Parameter | Description | Default |
|---|---|---|
| IP state | The Internet Protocol, version 4 | Enabled<br>**Note:** You cannot disable IP. |
| IP address and mask notation | Format for displaying an IP address and its network mask information. You can enable one of the following:<br>• Class-based format; example: 192.168.1.1 255.255.255.0<br>• Classless Interdomain Routing (CIDR) format; example: 192.168.1.1/24 | Class-based<br>**Note:** Changing this parameter affects the display of IP addresses, but you can enter addresses in either format regardless of the display setting. |
| Router ID | The value that routers use to identify themselves to other routers when exchanging route information. OSPF and BGP4 use router IDs to identify routers. RIP does not use the router ID. | The IP address configured on the lowest-numbered loopback interface.<br>If no loopback interface is configured, then the lowest-numbered IP address configured on the device. |
| IP Maximum Transmission Unit (MTU) | The maximum length an Ethernet packet can be without being fragmented. | 1500 bytes for Ethernet II encapsulation<br>1492 bytes for SNAP encapsulation |
| Address Resolution Protocol (ARP) | A standard IP mechanism that routers use to learn the Media Access Control (MAC) address of a device on the network. The router sends the IP address of a device in the ARP request and receives the device's MAC address in an ARP reply. | Enabled |
| ARP rate limiting | Lets you specify a maximum number of ARP packets the device will accept each second. If the device receives more ARP packets than you specify, the device drops additional ARP packets for the remainder of the one-second interval. | Disabled |
| ARP age | The amount of time the device keeps a MAC address learned through ARP in the device's ARP cache. The device resets the timer to zero each time the ARP entry is refreshed and removes the entry if the timer reaches the ARP age.<br>**Note:** You also can change the ARP age on an individual interface basis. Refer to IP interface parameters on page 54. | Ten minutes |
| Proxy ARP | An IP mechanism a router can use to answer an ARP request on behalf of a host, by replying with the interface's own MAC address instead of the host's. | Disabled |
| Static ARP entries | An ARP entry you place in the static ARP table. Static entries do not age out. | No entries |
| Time to Live (TTL) | The maximum number of routers (hops) through which a packet can pass before being discarded. Each router decreases a packet's TTL by 1 | 64 hops |

**TABLE 3** IP global parameters (continued)

| Parameter | Description | Default |
|---|---|---|
| | before forwarding the packet. If decreasing the TTL causes the TTL to be 0, the router drops the packet instead of forwarding it. | |
| Directed broadcast forwarding | A directed broadcast is a packet containing all ones (or in some cases, all zeros) in the host portion of the destination IP address. When a router forwards such a broadcast, it sends a copy of the packet out each of its enabled IP interfaces.<br>**Note:** You also can enable or disable this parameter on an individual interface basis. Refer to IP interface parameters on page 54. | Disabled |
| Directed broadcast mode | The packet format the router treats as a directed broadcast. The following formats can be directed broadcast:<br>• All ones in the host portion of the packet's destination address.<br>• All zeroes in the host portion of the packet's destination address. | All ones<br>**Note:** If you enable all-zeroes directed broadcasts, all-ones directed broadcasts remain enabled. |
| Source-routed packet forwarding | A source-routed packet contains a list of IP addresses through which the packet must pass to reach its destination. | Enabled |
| Internet Control Message Protocol (ICMP) messages | The Extreme device can send the following types of ICMP messages:<br>• Echo messages (ping messages)<br>• Destination Unreachable messages<br>• Redirect messages<br>**Note:** You also can enable or disable ICMP Redirect messages on an individual interface basis. Refer to IP interface parameters on page 54. | Enabled |
| ICMP Router Discovery Protocol (IRDP) | An IP protocol a router can use to advertise the IP addresses of its router interfaces to directly attached hosts. You can enable or disable the protocol, and change the following protocol parameters:<br>• Forwarding method (broadcast or multicast)<br>• Hold time<br>• Maximum advertisement interval<br>• Minimum advertisement interval<br>• Router preference level<br>**Note:** You also can enable or disable IRDP and configure the parameters on an individual interface basis. Refer to IP interface parameters on page 54. | Disabled |
| Maximum BootP relay hops | The maximum number of hops away a BootP server can be located from a router and still be used by the router's clients for network booting. | Four |
| Domain name for Domain Name Server (DNS) resolver | A domain name (example: extreme.router.com) you can use in place of an IP address for certain | None configured |

**TABLE 3** IP global parameters (continued)

| Parameter | Description | Default |
|---|---|---|
| | operations such as IP pings, trace routes, and Telnet management connections to the device. | |
| DNS default gateway addresses | A list of gateways attached to the device through which clients attached to the device can reach DNS. | None configured |
| IP load sharing | A feature that enables the device to balance traffic to a specific destination across multiple equal-cost paths.<br>Load sharing is based on a combination of destination MAC address, source MAC address, destination IP address, source IP address, and IP protocol.<br>**Note:** Load sharing is sometimes called Equal Cost Multi Path (ECMP). | Enabled |
| Maximum IP load sharing paths | The maximum number of equal-cost paths across which the Extreme device is allowed to distribute traffic. | Four |
| Origination of default routes | You can enable a device to originate default routes for the following route exchange protocols, on an individual protocol basis:<br>• RIP<br>• OSPF<br>• BGP4 | Disabled |
| Default network route | The device uses the default network route if the IP route table does not contain a route to the destination and also does not contain an explicit default route (0.0.0.0 0.0.0.0 or 0.0.0.0/0). | None configured |
| Static route | An IP route you place in the IP route table. | No entries |
| Source interface | The IP address the device uses as the source address for Telnet, RADIUS, or TACACS/TACACS+ packets originated by the device. The device can select the source address based on either of the following:<br>• The lowest-numbered IP address on the interface the packet is sent on.<br>• The lowest-numbered IP address on a specific interface. The address is used as the source for all packets of the specified type regardless of interface the packet is sent on. | The lowest-numbered IP address on the interface the packet is sent on. |

# IP interface parameters

The following table lists the interface-level IP parameters, their default values, and where to find configuration information.

**TABLE 4** IP interface parameters

| Parameter | Description | Default |
|---|---|---|
| IP state | The Internet Protocol, version 4 | Enabled<br>**Note:** You cannot disable IP. |
| IP address | A Layer 3 network interface address | None configured |

**TABLE 4** IP interface parameters (continued)

| Parameter | Description | Default |
|---|---|---|
| | The device has separate IP addresses on individual interfaces. | |
| Encapsulation type | The format of the packets in which the device encapsulates IP datagrams. The encapsulation format can be one of the following:<br>• Ethernet II<br>• SNAP | Ethernet II |
| IP Maximum Transmission Unit (MTU) | The maximum length (number of bytes) of an encapsulated IP datagram the device can forward. | 1500 for Ethernet II encapsulated packets<br>1492 for SNAP encapsulated packets |
| ARP age | Locally overrides the global setting. Refer to IP global parameters on page 52. | Ten minutes |
| Directed broadcast forwarding | Locally overrides the global setting. Refer to IP global parameters on page 52. | Disabled |
| ICMP Router Discovery Protocol (IRDP) | Locally overrides the global IRDP settings. Refer to IP global parameters on page 52. | Disabled |
| ICMP Redirect messages | Locally overrides the global setting. Refer to IP global parameters on page 52. | Enabled |
| DHCP gateway stamp | The device can assist DHCP/BootP Discovery packets from one subnet to reach DHCP/BootP servers on a different subnet by placing the IP address of the device interface that receives the request in the request packet's Gateway field.<br>You can override the default and specify the IP address to use for the Gateway field in the packets.<br>**Note:** UDP broadcast forwarding for client DHCP/BootP requests (bootpc) must be enabled and you must configure an IP helper address (the server's IP address or a directed broadcast to the server's subnet) on the port connected to the client. | The lowest-numbered IP address on the interface that receives the request |
| UDP broadcast forwarding | The device can forward UDP broadcast packets for UDP applications such as BootP. By forwarding the UDP broadcasts, the device enables clients on one subnet to find servers attached to other subnets.<br>**Note:** To completely enable a client's UDP application request to find a server on another subnet, you must configure an IP helper address consisting of the server's IP address or the directed broadcast address for the subnet that contains the server. Refer to the next row. | The device helps forward broadcasts for the following UDP application protocols:<br>• bootps<br>• dns<br>• netbios-dgm<br>• netbios-ns<br>• tacacs<br>• tftp<br>• time |
| IP helper address | The IP address of a UDP application server (such as a BootP or DHCP server) or a directed broadcast address. IP helper addresses allow the device to forward requests for certain UDP applications from a client on one subnet to a server on another subnet. | None configured |

# GRE IP tunnel

NetIron software supports the tunneling of packets with the Generic Routing Encapsulation (GRE) mechanism over an IP network, as described in RFC 2784. With GRE, packets are encapsulated in a transport protocol packet at a tunnel source and delivered to a tunnel destination, where they are unpacked and made available for delivery.

## Considerations in implementing this feature

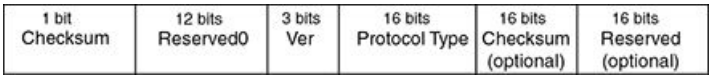
The considerations in implementing this feature are as follows:

- As a point-to-point tunnel configuration, GRE requires both ends of the tunnel to be configured.
- Only four-byte GRE headers are supported at the ingress (even though eight-byte headers can be processed at a transit node or the egress point).
- This device does not support the key and sequence numbering option with GRE (per RFC 2890).
- The current maximum number of tunnels is 8192 (with default as 256 tunnels).

NOTE
Do not forward packets from one type of tunnel to another type of tunnel in XPP. Packets may not be routed properly.

Figure 6 describes the GRE header format.

**FIGURE 6** GRE header format




**Checksum** - This field is assumed to be zero in this version. If set to 1 means that the **Checksum** (optional) and **Reserved** (optional) fields are present and the Checksum (optional) field contains valid information.

**Reserved0** - Bits 6:0 of the field are reserved for future use and must be set to 0 in transmitted packets. If bits 11:7 of the field are non-0, then a receiver must discard the packet. This field is assumed to be 0 in this version.

**Ver** - This field must be set to 0. This field is assumed to be 0 in this version.

**Protocol Type** - This field contains the EtherType of the payload protocol.

For details on configuring a GRE IP tunnel, refer Examples on page 122.

## GRE MTU enhancements

Enhancements have been introduced to support GRE MTU in support of RFC 4459. This includes support for the following:

- Signaling the Lower MTU to the Sources as described in Section 3.2 of RFC 4459
- Fragmentation of the Inner packet as described in Section 3.4 of RFC 4459

This enhancement also allows you to set a specific MTU value for packets entering a configured GRE tunnel. Packets whose size is greater than the configured value are fragmented and encapsulated with IP/GRE headers for transit through the tunnel. This feature supports Jumbo packets although they may be fragmented based on the MTU value configured.

# Configuring a GRE IP Tunnel

To configure a GRE IP Tunnel, configure the following parameters:

- CAM restrictions on page 57
- Maximum Number of Tunnels (optional)
- Tunnel Interface
- Source Address or Source Interface for the Tunnel
- Destination address for the Tunnel
- GRE Encapsulation
- IP address for the Tunnel
- Keep Alive Support (optional)
- TTL Value (optional)
- TOS Value (optional)
- MTU Value (optional)

## *Configuration considerations*

1. To enable keepalive when a GRE source and destination are directly connected, you must disable ICMP redirect on the tunnel source port on the GRE nodes. Otherwise, the keepalive packets go to the CPU where they can degrade CPU performance.
2. Whenever multiple IP addresses are configured on a tunnel source, the primary address of the tunnel is always used for forming the tunnel connections. Consequently, you must carefully check the configurations when configuring the tunnel destination.
3. GRE tunneling is supported for non-default VRFs.
4. When a GRE tunnel is configured, you cannot configure the same routing protocol on the tunnel through which you learn the route to the tunnel destination. For example, if the Extreme device learns the tunnel destination route through OSPF protocol, you cannot configure the OSPF protocol on the same Tunnel and vice-versa. When a tunnel has OSPF configured, the Extreme device cannot learn the tunnel destination route through OSPF. This could cause the system to become unstable.

   NOTE

   With GRE Dynamic-cam mode, at the Egress node, when a GRE packet is received, the Extreme device programs the CAM entries to forward the packets based on Inner DPA. These host CAM entries will be aging even if the traffic is hitting that CAM entries. This will cause the CAM entries to become aged out and recreated which could cause a small packet loss.

## *Configuring ECMP for routes through an IP GRE tunnel*

If multiple routes are using IP GRE tunnels to a destination, packets are automatically load-balanced between tunnels. This feature allows for load distribution of traffic among the available IP GRE tunnels. If the routes to a destination are both normal IP routes and routes through IP GRE tunnels, ECMP is not enabled.

## *CAM restrictions*

CAMs are partitioned on this device by a variety of profiles that you can select for your specific application.

To implement a CAM partition for a GRE tunnel, enter a command such as the following.

```
device(config)# cam-partition profile ipv4
```

**Syntax: [no] cam-partition profile** { **ipv4** | **ipv4-ipv6** | **ipv4-vpls** | **ipv4-vpn** | **ipv6** | **l2-metro** | **l2-metro-2** | **mpls-l3vpn** | **mpls-l3vpn-2** | **mpls-vpls** | **mpls-vpls-2** | **mpls-vpn-vpls** | **multi-service** | **multi-service-2** | **multi-service-3** | **multi-service-4** }

The **ipv4** parameter adjusts the CAM partitions, as described in the tables below, to optimize the device for IPv4 applications.

NOTE
The **ipv4** parameter is effective only if you first entered the following command:

```
device(config)# system-max ipv6-mcast-cam 0
```

The **ipv4-ipv6** parameter adjusts the CAM partitions, as described in the tables below to optimize the device for IPv4 and IPv6 dual stack applications.

The **ipv4-vpls** parameter adjusts the CAM partitions, as described in the tables below to optimize the device for IPv4 and MPLS VPLS applications.

The **ipv4-vpn** parameter adjusts the CAM partitions, as described in the tables below to optimize the device for IPv4 and MPLS Layer-3 VPN applications.

The **ipv6** parameter adjusts the CAM partitions, as described in the tables below to optimize the device for IPv6 applications.

The **l2-metro** parameter adjusts the CAM partitions, as described in the tables below to optimize the device for Layer 2 Metro applications.

The **l2-metro-2** parameter provides another alternative to **l2-metro** to optimize the device for Layer 2 Metro applications. It adjusts the CAM partitions, as described in the tables below for the XMR Series.

The **mpls-l3vpn** parameter adjusts the CAM partitions, as described in the tables below, to optimize the device for Layer 3, BGP or MPLS VPN applications.

The **mpls-l3vpn-2** parameter provides another alternative to **mpls-l3vpn** to optimize the device for Layer 3, BGP or MPLS VPN applications.

The **mpls-vpls** parameter adjusts the CAM partitions, as described in the tables below to optimize the device for MPLS VPLS applications.

The **mpls-vpls-2** parameter provides another alternative to **mpls-vpls** to optimize the device for MPLS VPLS applications. It adjusts the CAM partitions, as described in the tables below.

The **mpls-vpn-vpls** parameter adjusts the CAM partitions, as described in the tables below, to optimize the device for MPLS Layer-3 and Layer-2 VPN applications.

The **multi-service** parameter adjusts the CAM partitions, as described in the tables below to optimize the device for Multi-Service applications.

The **multi-service-2** parameter provides another alternative to multi-service to optimize the device for Multi-Service applications.

The **multi-service-3** parameter provides another alternative to multi-service to optimize the device for Multi-Service applications to support IPv6 VRF.

The **multi-service-4** parameter provides another alternative to multi-service to optimize the device for Multi-Service applications to support IPv6 VRF.

## *CAM partition profiles for the XMR Series and MLX Series devices*

Not all CAM profiles are compatible for running Layer 2 switching and configuring GRE tunnels simultaneously on this device.

**TABLE 5** Partition profiles for the XMR Series and MLX Series devices

| Compatible CAM profiles |
|---|
| default |
| ipv4 |
| ipv6 |
| ipv4-vpn |
| mpls-l3vpn |
| mpls-l3vpn-2 |
| multi-service-2 (Does not support GRE tunnel with VE interface as the source address in this profile.) |
| multi-service-3 (Does not support GRE tunnel with VE interface as the source address in this profile.) |
| multi-service-4 (Does not support GRE tunnel with VE interface as the source address in this profile.) |

## *Configuring the maximum number of tunnels supported*

You can configure the devices to support a specified number of tunnels using the following command.

```
device(config)# system-max ip-tunnels 512
device(config)# write memory
```

**Syntax: system-max ip-tunnels number**

The *number* variable specifies the number of GRE tunnels that can be supported.

The XMR Series and MLX Series permissible range is 1 - 8192. The default value is 256. The permissible range for CES 2000 Series devices is 32 - 128. The default value is 32. The permissible range for CER 2000 Series devices is 32 - 256. The default value is 32.

NOTE
Multicast over GRE tunnels for PIM can support up to the default system max of 256 tunnels if the required hardware resources are available.

NOTE
You must write this command to memory and perform a system reload for this command to take effect.

## *Configuring a tunnel interface*

To configure a tunnel interface, use a the following command.

```
device(config)# interface tunnel 1
device(config-tnif-1)
```

**Syntax: [no] interface tunnel tunnel id**

The *tunnel-id* variable is numerical value that identifies the tunnel being configured.Possible range is from 1 to the maximum configured tunnels in the system.

## *Configuring a source address or source interface for a tunnel interface*

To configure a source address for a specific tunnel interface, enter the following command.

```
device(config)# interface tunnel 1
device(config-tnif-1)tunnel source 10.0.8.108
```

To configure a source interface for a specific tunnel interface, enter the following command.

```
device(config)# interface tunnel 100
device(config-tnif-100)tunnel source ethernet 3/1
```

**Syntax: [no] tunnel source ip-address | port-no**

You can specify either of the following:

The *ip-address* variable is the source IP address being configured for the specified tunnel.The*port-no* variable is the source slot or port of the interface being configured for the specified tunnel. When you configure a source interface, there must be at least one IP address configured on that interface. Otherwise, the interface will not be added to the tunnel configuration and an error message like the following will be displayed: Error - Tunnel source interface 3/1 has no configured ip address.

It can be a physical or virtual interface (ve).

## *Configuring a destination address for a tunnel interface*

To configure a destination address for a specific tunnel interface, enter the following command.

```
device(config)# interface tunnel 1
device(config-tnif-1)tunnel destination 10.108.5.2
```

**Syntax: [no] tunnel destination ip-address**

The *ip-address* variable is destination IP address being configured for the specified tunnel.

NOTE
If GRE is configured with a tunnel destination reachable over LAG ports, load balancing will only work with the following LAG types: server LAG or LACP with server LAG. Traffic cannot be load-balanced across multiple ports of a switch LAG.

NOTE
Traffic from a GRE tunnel entering a MPLS tunnel is not supported.

## *Configuring a tunnel interface for GRE encapsulation*

To configure a specified tunnel interface for GRE encapsulation, enter the following command.

```
device(config)# interface tunnel 1
device(config-tnif-1)tunnel mode gre ip
```

**Syntax: [no] tunnel mode gre ip**

The **gre** parameter specifies that the tunnel will use GRE encapsulation

The **ip** parameter specifies that the tunnel protocol is IP.

## *Configuring an IP address for a tunnel interface*

To configure an IP address for a specified tunnel interface, enter the following command.

```
device(config)# interface tunnel 1
device(config-tnif-1)ip address 10.10.3.1/24
```

**Syntax: [no] ip address ip-address**

The *ip-address* variable is the IP address being configured for the specified tunnel interface.

## Configuring keep alive support

This parameter is optional. It lets the device maintain a tunnel in an up or down state based upon the periodic sending of keep alive packets and the monitoring of responses to the packet. If the packets fail to reach the tunnel's far end more frequently than the configured number of retries, the tunnel is placed in a down state. A keep alive packet is a GRE IP packet with no payload.

To configure the keep alive option, enter the following command.

```
device(config)# interface tunnel 1
device(config-tnif-1)keepalive 5 4
```

**Syntax: [no] keepalive seconds retries**

The *seconds* variable specifies the number of seconds between each initiation of a keep alive message. The range for this interval is 1 - 32767 seconds. The default value is 10 seconds.

The *retries* variable specifies the number of times that a packet is sent before the system places the tunnel in the down state. Possible values are from 1 - 255. The default number of retries is 3.

## Configuring a TTL value

This is an optional parameter that allows you to set the Time-to-Live value for the outer IP header of the GRE tunnel packets.

To configure the TTL value, enter the following command.

```
device(config)# interface tunnel 1
device(config-tnif-1)tunnel ttl 100
```

**Syntax: [no] tunnel ttl ttl-value**

The*ttl--value* variable specifies a TTL value for the outer IP header. Possible values are 1 - 255. The default value is 255.

## Configuring a TOS value

This is an optional parameter that allows you to set the TOS value for the outer IP header of the GRE tunnel packets.

To configure the TOS value, enter the following command.

```
device(config)# interface tunnel 1
device(config-tnif-1)tunnel tos 100
```

**Syntax: [no] tunnel tos tos-value**

The *tos-value* variable specifies a TOS value for the outer IP header.

The XMR Series and MLX Series possible values are 0 - 255. The default value is 0.

The CES 2000 Series and CER 2000 Series devices possible values are 0 - 63. The default value is 0.

## Configuring GRE session enforce check

The **gre-session-enforce-check** command lets you enable the GRE session enforce check. When a GRE packet arrives and this feature is enabled, the system tries to match the GRE packet source and destination address pair with the tunnel configured destination and source pair. If the pairs do not match, the packet is dropped in the hardware. The default behavior when this command is disabled is to terminate the GRE tunnel based on the destination IP address.

> NOTE
> The CES 2000 Series and CER 2000 Series devices currently do not support the **ip-tunnel-policy** and the **accounting-enable commands.**

To configure the GRE session enforce check, go to the IP tunnel policy context, and then enter the **gre-session-enforce-check** command.

```
device(config)#ip-tunnel-policy
device(config-ip-tunnel-policy)#gre-session-enforce-check
```

**Syntax: [no] gre-session-enforce-check**

To disable the GRE session enforce check, use the **no** form of this command. This command is disabled by default. You might have to write the configuration to memory and reload the system whenever the configuration of this command is changed because a one-time creation of a source-ingress CAM partition is necessary. The system prompts you if the memory write and reload are required.

The first-time execution of certain commands necessitates the creation of a source-ingress CAM partition, after which you write to memory and reload. These commands are **gre-session-enforce-check** , **ipv6-session-enforce-check** , and **accounting-enable** . After this CAM partition is created, it is not necessary to follow either of the other two commands with a memory write and reload. The CAM partition is created out of the Layer 4 CAM and has no impact on the Layer 3 route scalability.

## *Configuring a maximum MTU value for a tunnel interface*

You can set an MTU value for packets entering the tunnel. Packets that exceed either the default MTU value of 1476 bytes or the value that you set using this command are fragmented for transit through the tunnel. The default MTU value is set to 1476.

> NOTE
> The tunnel MTU should be configured explicitly for packet size greater than 1476 bytes.

The following command allows you to change the MTU value for packets transiting "tunnel 1".

```
device(config)# interface tunnel 1
device(config-tnif-1)tunnel mtu 1500
```

**Syntax: [no] tunnel mtu packet-size**

The *packet-size* variable specifies the maximum MTU size in bytes for the packets transiting the tunnel.

> NOTE
> To prevent packet loss after the 24 byte GRE header is added, make sure that any physical interface that is carrying GRE tunnel traffic has an IP MTU setting at least 24 bytes greater than the tunnel MTU setting.

## *Bypassing ACLs in a GRE tunnel*

Use this procedure to disable IPv4 and IPv6 ACLs on the terminating node of a GRE tunnel for internal traffic coming over the tunnel.

> NOTE
> Disabling ACL processing on GRE tunnels also disables support for the following features on all GRE tunnels:
> - All features employing IPv4 or IPv6 ACLs
> - BFD over MPLS
> - Multicast
> - PBR
> - OpenFlow

1. Access global configuration mode.

```
device# configure terminal
```

2. Access ACL global policy configuration mode.

```
device(config)# acl-policy
```

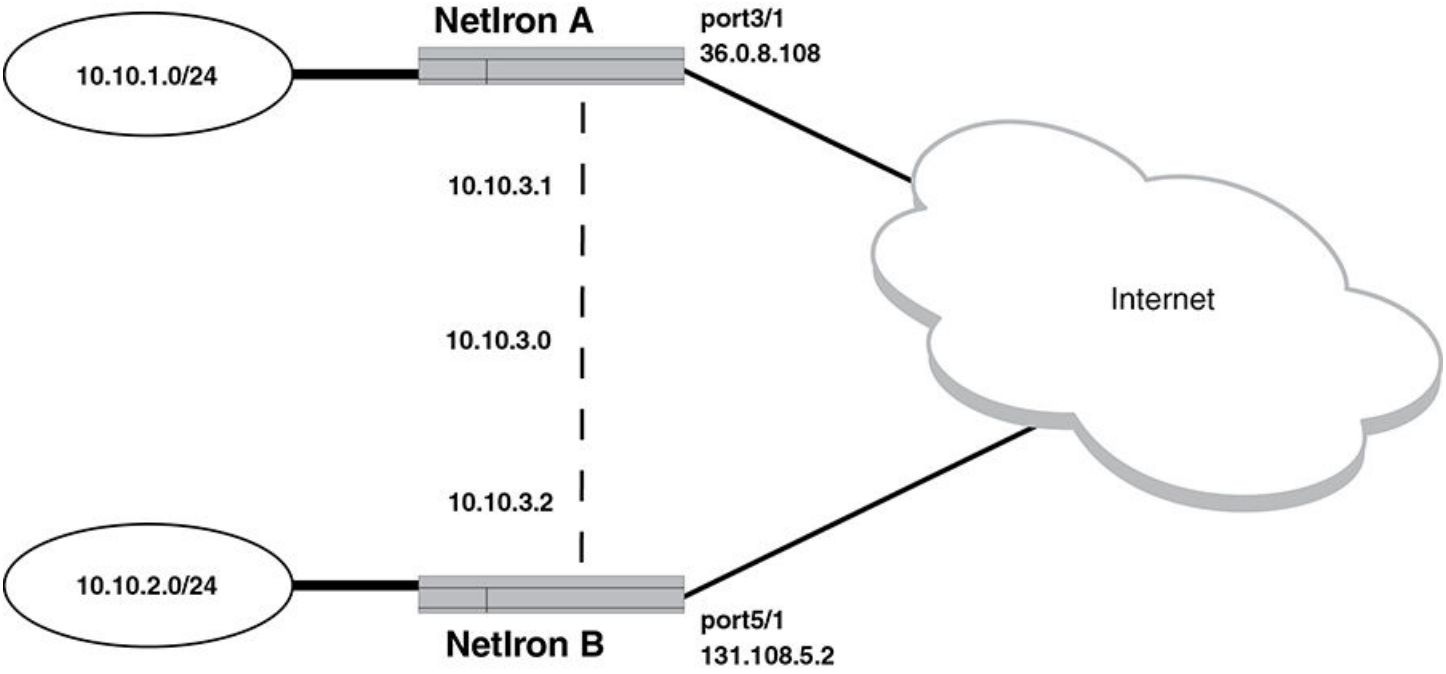
3. Enter the **disable-acl-for-gre** command.

```
device(config-acl-policy)# disable-acl-for-gre
```

## *Example of a GRE IP tunnel configuration*

In this example, a GRE IP Tunnel is configured between the NetIron A device and the NetIron B device. Traffic between networks 10.10.1.0/24 and 10.10.2.0/24 is encapsulated in a GRE IP packet sent through the tunnel on the 10.10.3.0 network. and unpacked and sent the destination network. A static route is configured at each device to go through the tunnel interface to the target network.

**FIGURE 7** GRE IP tunnel configuration example




### Configuration example for NetIron A

```
device(config)# interface ethernet 3/1
device(config-int-e10000-3/1)# ip address 36.0.8.108/24
device(config)# interface tunnel 1
device(config)# vrf forwarding red
device(config-tnif-1)# tunnel source 36.0.8.108
device(config-tnif-1)# tunnel destination 131.108.5.2
device(config-tnif-1)# tunnel mode gre ip
device(config-tnif-1)# ip address 10.10.3.1/24
device(config-tnif-1)# int loopback 1
device(config-tnif-1)# vrf forwarding red
device(config-tnif-1)# ip address 10.10.1.1/32
device(config-tnif-1)# keepalive 5 4
device(config-tnif-1)# vrf red
device(config-tnif-1)# rd 1:1
device(config-tnif-1)# address-family ipv4
```

```
device(config-tnif-1)# ip route 10.10.2.0/24 10.10.3.2
device(config-tnif-1)# exit
device(config)# ip route 10.10.2.0/24 10.10.3.2
```

### Configuration example for NetIron B

```
device(config)# interface ethernet 5/1
device(config-if-e10000-5/1)# ip address 131.108.5.2/24
device(config)# interface tunnel 1
device(config)# vrf forwarding red
device(config-tnif-1)# tunnel source ethernet 5/1
device(config-tnif-1)# tunnel destination 36.0.8.108
device(config-tnif-1)# tunnel mode gre ip
device(config-tnif-1)# ip address 10.10.3.2/24
device(config-tnif-1)# int loopback 1
device(config-tnif-1)# vrf forwarding red
device(config-tnif-1)# ip address 10.10.1.1/32
device(config-tnif-1)# keepalive 5 4
device(config-tnif-1)# vrf red
device(config-tnif-1)# rd 2:2
device(config-tnif-1)# address-family ipv4
device(config-tnif-1)# ip route 10.10.2.0/24 10.10.3.2
device(config-tnif-1)# exit
device(config)# ip route 10.10.2.0/24 10.10.3.2
```

**NOTE**

Traffic from a GRE tunnel entering a MPLS tunnel is not supported.

## *Displaying GRE tunneling information*

You can display GRE tunneling information using the **show ip interface,show ip route** and **show interface tunnel** commands as shown in the following.

```
device# show ip interface tunnel 1
Interface Tunnel 1
  port enabled
  port state: UP
  ip address: 10.255.255.13/24
  Port belongs to VRF: red
  encapsulation: ETHERNET, mtu: 1476
  directed-broadcast-forwarding: disabled
  ip icmp redirect: enabled
  No inbound ip access-list is set
  No outbound ip access-list is set
  No Helper Addresses are configured.
```

**Syntax: show ip interface tunnel tunnel-no**

The **show ip route** command displays routes that are pointing to a GRE tunnel as shown in the following.

**Syntax: show interface tunnel tunnel-no**

```
device# show ip route
     Total number of IP routes: 2
     Type Codes - B:BGP D:Connected I:ISIS S:Static R:RIP O:OSPF; Cost -Dist/Metric
             Destination        Gateway              Port          Cost    Type Uptime src-vrf
1             10.10.2.0/24       10.10.3.2          gre_tnl 1        1/1           S    7h55m      -
2             10.10.3.0/24       DIRECT             gre_tnl 1        0/0           D    7h55m      -

device# show interface tunnel 1
    Tunnel1 is up, line protocol is up
    Hardware is Tunnel
    Tunnel source 10.45.3.3
    Tunnel destination is 10.45.48.1
    Tunnel mode gre ip
```

```
 No port name
 Internet address is 10.255.255.13/24,
Tunnel TOS 0, Tunnel TTL 255 MTU 1476 bytes
Keepalive is not Enabled
VRF Forwarding: Red
```

**Syntax: show ip tunnel tunnel-no**

```
device# show ip-tunnels 1
IPv4 tnnl 1 UP : src_ip 36.0.8.108, dst_ip 131.108.5.2
TTL 255, TOS 0, NHT 0, MTU 1480, vrf: red
```

# GRE tunnel VRF support

GRE tunnel VRF support maintains end - end VRF autonomy with the GRE tunnel. You can also create separate GRE tunnels on a per-VRF basis.

### GRE tunnel VRF support overview

VRFs are used to segment the traffic associated with various customers of interest (CIs). These CIs are spread across geographical areas. Hence CIs enable use of GRE tunnels for non-default VRFs.

### Configuration considerations

- The **vrf forwarding** command is optional. If this command is not specified, then the VRF is assumed as default VRF.
- The configured VRF must exist in the MLX Series device.
- Configured VRFs should be same on both nodes of the GRE tunnel, for proper working of GRE.
- Configuration is allowed for two tunnels when the tunnel destination addresses are the same and the corresponding source addresses are different. Also, configuration is allowed for two tunnels when the tunnel source addresses are the same and the corresponding destination addresses are different.
- The **vrf forwarding** configuration is supported only for GRE tunnel.
- L3VPN ID information with respect to each tunnel is configured by the **vrf forwarding** command under the tunnel interface.

### Configuring the GRE VRF tunnel

**Syntax: vrf forwarding** *vrf-name*

### Error messages

The following messages are displayed for different VRF configurations.

1. If a tunnel is configured with the VRF configuration and tunnel mode is non-GRE IP, then the following error message is displayed.
   - Error: Tunnel mode should be GRE IP/ IPSec when VRF forwarding is configured on tunnel.
2. If the tunnel source interface is on a non-supported card, then the configuration will be rejected, if the tunnel source is a physical interface or a virtual interface.
   - Error: Tunnel source interface eth 1/2 or ve103 cannot be a BR-MLX-10Gx24-DM/Gen1.1 port.
3. If the tunnel source is a loopback interface, a warning will be displayed if a BR-MLX-10Gx24-DM/Gen1.1 card is present in the chassis.
   - Warning: Tunnel source configured as loopback could be using a BR-MLX-10Gx24-DM/Gen1.1 port.
4. The VRF forwarding configuration is supported only if tunnel source is pre-configured. Otherwise, an error message is displayed.
   - Error: Please configure tunnel source before configuring tunnel VRF.

5. The VRF forwarding configuration is rejected if GRE is configured as MPLS interface and GRE is part of the VRF.
    - Error: GRE configured as MPLS interface with VRF not supported .

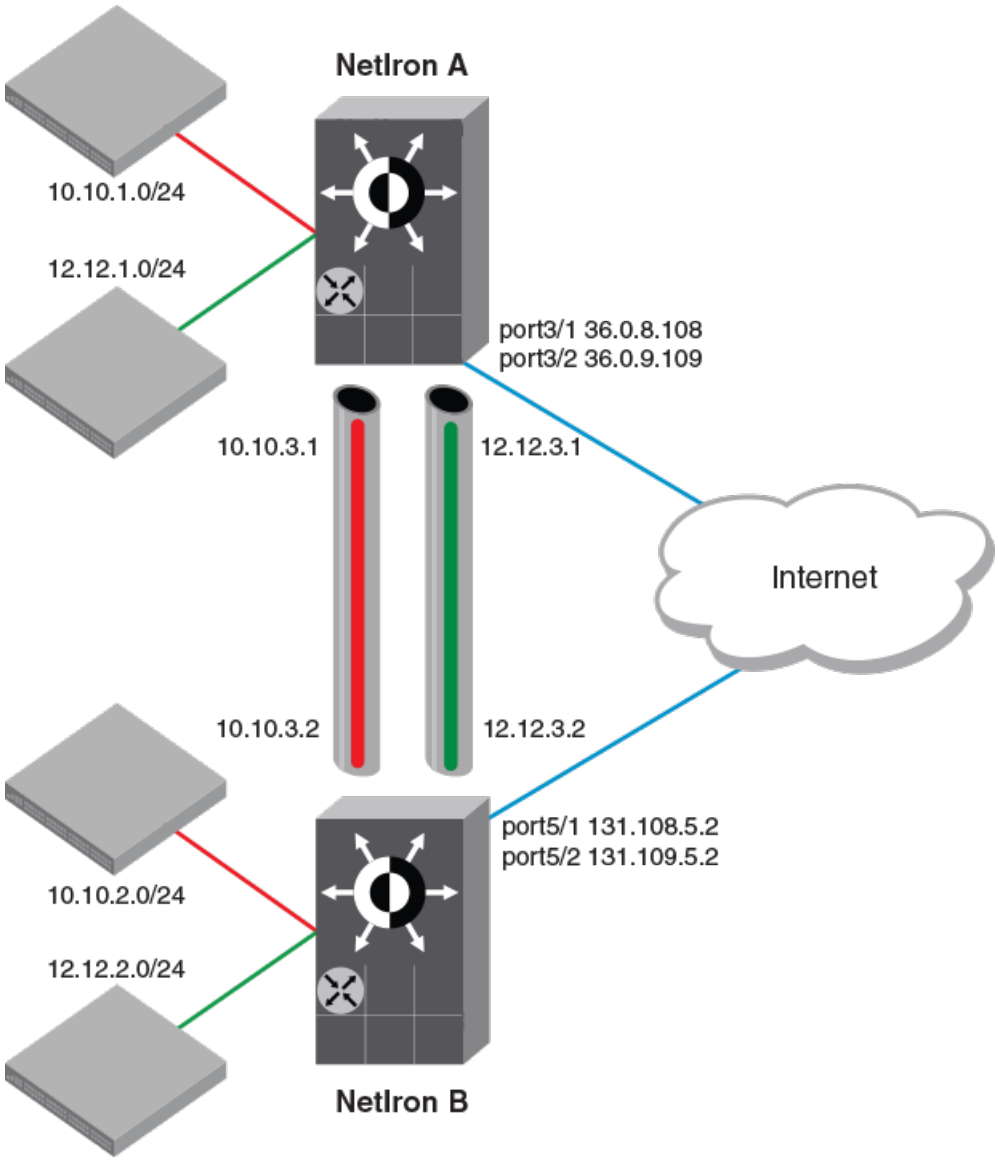
**Example of a GRE VRF tunnel configuration:**

In the following example, a GRE VRF tunnel is configured between the NetIron OS A device and the NetIron OS B device. Traffic between networks 10.10.1.0/24 (VRF red) and 10.10.2.0/24 (VRF red) is encapsulated in a GRE IP packet (Tunnel 1 corresponding to VRF red) sent through the tunnel on the 10.10.3.0 network and unpacked and sent to the destination network. A static route is configured at each device to go through the tunnel interface to the target network.

On the B device, the GRE tunneled packet is received in default VRF. It is unpacked and sent to the destination network on VRF red.

In this example, VRF is configured to the tunnel interface configuration using the **vrf forwarding** command (as done for all other interfaces like physical interface, the loopback interface, and so on.

**GRE VRF tunnel configuration example**



### Configuration example for NetIron A

```
(NetIron A)

device(config)# interface eth 3/1
device(config-int-e10000-3/1)# ip address 36.0.8.108/32
device(config)# interface tunnel 1
device(config-tnif-1)# tunnel mode gre ip
device(config-tnif-1)# tunnel source 36.0.8.108
device(config-tnif-1)# tunnel destination 131.108.5.2
device(config-tnif-1)# vrf forwarding  red
device(config-tnif-1)# ip address 10.10.3.1/24
device(config-tnif-1)# int loopback 1
device(config-lbif-1)# vrf forwarding red
device(config-lbif-1)# ip address 10.10.1.1/32
device(config-tnif-1)# vrf red
device(config-vrf-red)# rd 1:1
device(config-vrf-red)# address-family ipv4
device(config-vrf-red-ipv4)# ip route 10.10.2.0/24 10.10.3.2
device(config-vrf-red-ipv4)# exit-vrf

(NetIron A)

device(config)# interface eth 3/1
device(config-int-e10000-3/1)# ip address 36.0.9.108/32
device(config-tnif-1)# interface tunnel 1
device(config-tnif-1)# tunnel mode gre ip
device(config-tnif-1)# tunnel source 36.0.9.108
device(config-tnif-1)# tunnel destination 131.109.5.2
device(config-tnif-1)# vrf forwarding  green
device(config-tnif-1)# ip address 12.12.3.1/24
device(config-tnif-1)# interface eth 3/1
device(config-if-e10000-3/1)# ip address 36.0.9.108/32
device(config-if-e10000-3/1)# int loopback 2
device(config-lbif-2)# vrf forwarding green
device(config-lbif-2)# ip address 12.12.1.1/32
device(config-tnif-1)# vrf red
device(config-vrf-red)# rd 1:1
device(config-vrf-red)# address-family ipv4
device(config-vrf-red-ipv4)# ip route 12.12.2.0/24 12.12.3.2
device(config-vrf-red-ipv4)# exit-vrf
```

### Configuration example for NetIron B

```
(NetIron B)

device(config)# interface eth 3/1
device(config-int-e10000-3/1)# ip address 131.108.5.2/32
device(config)# interface tunnel 1
device(config-tnif-1)# tunnel mode gre ip
device(config-tnif-1)# tunnel source 131.108.5.2
device(config-tnif-1)# tunnel destination 36.0.8.108
device(config-tnif-1)# vrf forwarding  red
device(config-tnif-1)# ip address 10.10.3.2/24
device(config-tnif-1)# int loopback 1
device(config-lbif-1)# vrf forwarding red
device(config-lbif-1)# ip address 10.10.2.1/32
device(config-tnif-1)# vrf red
device(config-vrf-red)# rd 2:2
device(config-vrf-red)# address-family ipv4
device(config-vrf-red-ipv4)# ip route 10.10.1.0/24 10.10.3.1
device(config-vrf-red-ipv4)# exit-vrf

(NetIron B)

device(config)# interface eth 3/1
device(config-int-e10000-3/1)# ip address 131.109.5.2/32
device(config-tnif-1)# interface tunnel 1
device(config-tnif-1)# tunnel mode gre ip
device(config-tnif-1)# tunnel source 131.109.5.2
device(config-tnif-1)# tunnel destination 36.0.9.108
```

```
device(config-tnif-1)# vrf forwarding  green
device(config-tnif-1)# ip address 12.12.3.2/24
device(config-tnif-1)# interface eth 3/1
device(config-if-e10000-3/1)# ip address 36.0.9.108/32
device(config-if-e10000-3/1)# int loopback 2
device(config-lbif-2)# vrf forwarding green
device(config-lbif-2)# ip address 12.12.2.1/32
device(config-tnif-1)# vrf red
device(config-vrf-red)# rd 2:2
device(config-vrf-red)# address-family ipv4
device(config-vrf-red-ipv4)# ip route 12.12.1.0/24 12.12.3.1
device(config-vrf-red-ipv4)# exit-vrf
```

Once the configuration is completed, the tunnel interface will come up operationally and become part of the corresponding VRF. Both MP and LP will have VRF information corresponding to the tunnel.

The route entry in that VRF shows the tunnel interface as a directly connected interface. Once a static route is configured with a destination as the CI in a VRF, the next hop will point to the corresponding tunnel interface for that VRF.

### MP CPU forwarding

When the MP has to send a packet over the GRE tunnel, it creates the GRE encapsulated IP packet and sends it to the LP for transmission out of the port. The MP also supports fragmentation of packets going out of GRE.

With respect to GRE support for VRF, the MP does a route lookup on the packet for that VRF. The route look points to GRE tunnel as next hop. Control packets, such as ping and routing protocol packets for a VRF, will be encapsulated by the GRE and sent across the GRE tunnel, and sent to the LP for transmission out of the port.

### LP CPU forwarding

When the incoming IP packet is more than 1476 bytes (in the default IP MTU scenario) or exceeds the IP MTU of the tunnel interface, the packets must be fragmented and sent with GRE encapsulation. The LP does the fragmentation and sends out the packets. To forward the packets to the correct GRE tunnel as per the VRF of incoming packet, mapping is provided by route entry. This works once the route entry in VRF points to the GRE tunnel as the next hop.

> **NOTE**
> Other tunnel optional configurable parameters for tunnel like Keep alive, TTL, TOS, and so on, are supported by the GRE tunnel.

### GRE tunnel VRF limitations

- The GRE tunnel VRF supports only the IPv4 addresses.
- Multicast is not supported on GRE tunnel.
- There is no dynamic CAM model for the IP GRE.
- GRE encapsulation of MPLS packet is also not supported.
- The GRE tunnel VRF support is applicable to all Gen 2 cards except BR-MLX-10Gx24-DM.
- ISIS is not supported for interface having VRF configuration. Hence only static, OSPF, BGP and RIP protocols are supported.
- PBR does not support VRF in current release. However, if we apply a PBR policy to an interface with VRF configured, then PBR will not work, but PBR policies' next-hop can be a tunnel interface irrespective of the tunnel being in any VRF.
- CER 2000 Series and CES 2000 Series devices do not support VRF over GRE tunnel.

Following **show** commands display the following VRF information:

```
device(config)#show interface tunnel 1
Tunnel1 is up, line protocol is up
  Hardware is Tunnel
  Tunnel source  36.0.8.108
  Tunnel destination is 131.108.5.2
  Tunnel mode gre ip
```

```
  No port name
  Internet address is: 10.10.3.1/24
  Tunnel TOS 0, Tunnel TTL 255, Tunnel MTU 1476 bytes
  Keepalive is Enabled : Interval 10, No.of Retries 3
  Total Keepalive Pkts Tx: 2, Rx: 2
  VRF Forwarding: Red

device(config)#show ip interface tunnel 1
Interface Tunnel 1
  port enabled
  port state: UP
  ip address: 10.10.3.1/24
  Port belongs to VRF: red
  encapsulation: ETHERNET, mtu: 1476
  directed-broadcast-forwarding: disabled
  ip icmp redirect: enabled
  ip local proxy arp: disabled
  ip ignore gratuitous arp: disabled
  No inbound ip access-list is set
  No outbound ip access-list is set
  No Helper Addresses are configured.

device(config)# show ip-tunnels 1
IPv4 tnnl 1 UP : src_ip 36.0.8.108, dst_ip 131.108.5.2
TTL 255, TOS 0, NHT 0, MTU 1480, vrf: red
```

# Multicast over GRE tunnel

NOTE

MTU fragmentation for multicast traffic is not enabled over a GRE tunnel. Packets are transmitted without MTU fragmentation. This behavior is applicable on MLX Series, XMR Series, CER 2000 Series, and CES 2000 Series devices.

NetIron software supports Multicast over a point-to-point GRE tunnel. Multicast over a GRE tunnel allows multicast packets to be transported through a GRE tunnel across an IP cloud towards its receiver. A GRE tunnel is provisioned at each end of the IP cloud. A GRE tunnel is a virtual IP tunnel; the IP tunnel source can also be a VE interface. The IP cloud sitting in between the two GRE endpoints serves as a PIM enabled logical link. As bidirectional control messages are sent over the GRE tunnel, the multicast distribution tree is established across the IP cloud. Multicast data is encapsulated with a predefined GRE header at the ingress node. The GRE packet is routed within the IP cloud using the outer unicast GRE destination address. As the packet reaches the egress node of the tunnel, the packet is decapsulated. The multicast packet continues on its way to the multicast distribution tree to reach its receivers.

## Configuring PIM GRE tunnel

The Extreme device PIM GRE tunnel configuration allows you to enable PIM Sparse (PIM-SM) and PIM Dense (PIM-DM) on a GRE tunnel.

### *Enabling PIM-SM on a GRE tunnel interface*

To enable PIM-SM on a GRE Tunnel Interface, enter the following command.

```
device(config)#interface tunnel 20
device(config-tnif-20)#ip pim-sparse
```

**Syntax: [no] ip pim-sparse**

### *Enabling PIM-DM on a GRE tunnel interface*

To enable PIM-DM on a GRE Tunnel Interface, enter the following command.

```
device(config)#interface tunnel 20
device(config-tnif-20)#ip pim
```

**Syntax: [no] ip pim**

## Configuring PIM GRE tunnel using the strict RPF check

The device PIM GRE tunnel configuration allows you to enforce strict rpf check rules on (s,g) entry on a GRE tunnel interface. The (s,g) entry uses the GRE tunnel as a RPF interface. During unicast routing transit, GRE tunnel packets may arrive at different physical interfaces. The **ip pim tunnel rpf-strict** command allows you to limit a specific port to accept the (s,g) GRE tunnel traffic.

> NOTE
> The configuration is not recommended for all users, it is only needed if the user wants to override the default behavior.

When the GRE encapsulated multicast packet is received, hardware processing attempts to find a match in the CAM session based on the inner (s,g) entry. If hardware processing cannot find the inner (s,g) entry in the CAM session, the packet will be dropped. If the **ip pim tunnel rpf-strict** command is configured on a GRE tunnel interface, hardware processing will check on the (s,g) entry, and verify that the packet matches the physical port on the GRE tunnel interface, and the GRE tunnel vlan id.

To limit a specific port to accept the (s,g) GRE tunnel traffic, enter the following command.

```
device(config)#interface tunnel 20
device(config-tnif-20)#ip pim tunnel rpf-strict
```

**Syntax: [no] ip pim tunnel** [ **rpf-strict** ]

The rpf-strict option allows you to set the strict rpf check on the multicast entry.

# Tunnel statistics for a GRE tunnel or IPv6 manual tunnel

At a global level, you can enable the collection of statistics for generic routing encapsulation (GRE) tunnels and manual IPv6 tunnels. With this feature, the Extreme device collects the statistics for GRE and IPv6 manual tunnels and displays packet counters for tunnels at the management processor (MP). This feature collects and displays unicast and multicast packets over both directions of the tunnels.

Statistics collection is not enabled by default, so you need to enter the IP tunnel policy configuration level and then issue the **accounting-enable** command to start collecting the statistics for GRE and IPv6 manual tunnels. This procedure is described in Enabling tunnel statistics on page 73. This required preliminary ensures that the source-ingress CAM partition is not allocated unless statistics collection or tunnel session enforcement checks are actually needed. (Because the statistics enable does not enforce the GRE and IPv6 tunnel session checks by default, these capabilities have their own enable commands in the IP tunnel policy CLI level. The applicable commands are described in Configuring GRE session enforce check on page 61 and Configuring IPv6 session enforce check on page 73.) You can view examples of related show command output in Displaying GRE tunnel information and statistics on page 134.

The remainder of this introduction to tunnel statistics describes reload behavior for certain commands and detailed notes and restrictions that apply to the support for tunnel statistics.

# Reload behavior and the source-ingress CAM partition

When one of the three tunnel-related commands is configured at the CLI level for IP tunnel policy (entered by use of the **ip-tunnel-policy** command), you might need to save the configuration and reload the device to create the required source-ingress-CAM partition. If the memory write and reload are needed, the system prompts for these steps after you finish the enable commands. The condition for which you might need to write and reload is the absence of the source-ingress-CAM partition. If this partition does not exist, the first time that you run either the **gre-session-enforce-check** , **ipv6-session-enforce-check** , or **accounting-enable** command, the system prompts you. Thereafter, when you run any of these three commands to disable or enable a feature, the system does not prompt. Removing any of the configurations can be done at anytime and does not necessitate a reload. The new configuration immediately becomes effective, but the source-ingress CAM partition is removed only upon the next reload.

# Operational notes

The subsections that following describe operational characters that relate to the statistics collection.

## *Source-ingress-CAM partition*

The CAM profile restrictions for this feature are the same as those for the tunnel session enforce-check configuration. This feature is not supported in those CAM profiles for which the system cannot allocate the source-ingress-CAM partition that is needed to support the accounting and session check enforcement. The CLI engine checks for compliance and rejects an attempt to enable statistics in this situation. Currently the following CAM profiles are not supported for IP tunnel statistics:

- IPv6
- L2-metro-2
- MPLS-L3VPN-2
- MPLS-VPLS-2
- MPLS-VPN-VPLS
- multi-service-2
- multi-service-3
- multi-service-4

## *6to4 automatic tunnels*

Statistics collection is supported only for manual IPv6 and GRE tunnels. The system does not support statistics collection for 6to4 automatic IPv6 tunnels because, for automatic 6to4 tunnels, only tunnel source-ip is configured, and the destination is known only at runtime when a remote node tries to use this tunnel. The destination points can come up or go down without the local router having any information on how many destinations are to be used for 6to4 tunnels. This uncertainty can cause scalability issues, so neither statistics collection nor session-enforce check are not supported for 6to4 automatic tunnels.

## *Multicast-over-GRE packets*

This feature counts multicast over GRE packets. You can see the multicast packet count by using the **show interface tunnel** command. You can use other CLI commands to display the aggregate unicast and multicast statistics for the GRE tunnels. For a description of all the applicable show commands, refer to Displaying GRE tunnel information and statistics on page 134.

## Statistics polling on the MP and LP

The LP module polls the statistics once every second. For every one second, the module polls the statistics either for 5000 entries or until the completion of a specific application. (The same polling mechanism is also used for other applications, such as IP, MPLS, L3VPN, VLL, VPLS and IP Tunnel.) After all the applications are polled, the system waits for 220 seconds to schedule the next polling event. However, the LP module synchronizes statistics to the MP every 30 seconds, so 30 seconds is the granularity of statistics.

The LP synchronizes statistics to the MP in background every 30 seconds, and the MP stores the statistics for all tunnels for every LP module. If a LP module at either the tunnel ingress or egress, the system uses the current stored statistics for that LP module for display (and continue to poll the rest of working modules to get the latest statistics). This mechanism ensures that the tunnel counters never go down (if no clear statistics command is performed on the tunnel).

When a tunnel is down, the LP does not poll the statistics for that tunnel. The LP keeps the old counters as is until you explicitly clear them on the CLI. These counters are displayed when the tunnel is down. When the tunnel comes back up, it resumes polling and adds the new packet counts to the stored statistics and displays the updated statistics.

## Clearing the statistics

When you issue the **clear statistics tunnel** command with specific parameters, the operation clears statistics for either one or all of the tunnels regardless of the circumstance-- whether the tunnel is up or down, on an ingress or egress module, and so on. Refer to Clearing GRE tunnel and manual IPv6 tunnel statistics on page 73 for a description of the clear statistics tunnel command.

## Tunneled packets that encounter an ACL

If a packet reaches the ACL permit or deny clauses for the inner IPv4 or IPv6 addresses when it comes through the IP tunnel at the egress node, the packet is not counted as a receive-from-tunnel packet. Instead, it is counted as an ACL packet. You can view ACL packets by using the show access-list accounting command.

IPv6 ACL lookup is performed on the inner IPv6 packet at the tunnel egress. This depends on the port register for the Layer 2 ACL or Layer 3 ACL control, which is performed in parallel.

## Switchover behavior

The LP sends statistics to both the active and the standby MP modules. If an MP switches over, the new-active MP polls the statistics again so it can display the latest statistics. The counters are equal to or greater than the statistics before the switchover for the working modules. If any module goes down before the switchover, the new active MP uses the stored counters to display the statistics for that module.

## Hitless operating system upgrade behavior

When a hitless operating system (OS) upgrade occurs, the tunnel statistics are saved and retrieved after the reset of the LP is complete. The system can retrieve the old statistics and do the polling to get the latest PRAM statistics. After the hitless upgrade, the system can display the correct packet counters.

## Behavior after an LP failure

If LP module goes down, the counters for that LP are preserved. After the LP comes back up, the preserved counters for that LP can be displayed.

### *Feature scalability*

An XMR Series device supports 256 tunnels by default and 8000 tunnels for its maximum number of tunnels. The system supports statistics for all tunnels because the source ingress CAM partition has 16000 entries that can support the statistics for all tunnels.

## Enabling IP tunnel or manual IPv6 statistics

This section describes how to enable and clear statistics for GRE or manual IPv6 tunnels. The enable for this feature is global in scope. The enabling command is one of three enable commands that you run in the IP tunnel policy context of the CLI. (These commands are **gre-session-enforce-check** , **ipv6-session-enforce-check** , and **accounting-enable** . The **ip-tunnel-policy** command puts the CLI in the mode for executing them.) To see examples of tunnel statistics, refer to Displaying GRE tunnel information and statistics on page 134.

### *Enabling tunnel statistics*

> NOTE
> The CES 2000 Series and CER 2000 Series devices currently do not support the **ip-tunnel-policy** and the **accounting-enable commands.**

To enable the GRE tunnel or manual IPv6 tunnel statistics, go to the IP tunnel policy mode of the CLI and issue the **accounting-enable** command, as the following example illustrates.

```
device(config)#ip-tunnel-policy
device(config-ip-tunnel-policy)#accounting-enable
```

**Syntax: [no] accounting-enable**

To turn off tunnel statistics gathering, use the keyword **no** to the accounting-enable command.

The system might prompt you to write the configuration to memory and reload the system. If the system has not yet allocated a source-ingress CAM partition, it prompts you to write the results of the current configuration to memory and reload the system.

The first-time execution of certain commands can prompt the allocation of a source-ingress CAM partition that is required by certain features. These commands are **gre-session-enforce-check** , **ipv6-session-enforce-check** , and **accounting-enable** . After this CAM partition is allocated, you do not need to do the memory write and reload after the first-time execution of the other two commands.

### *Clearing GRE tunnel and manual IPv6 tunnel statistics*

You can clear all of the statistics for either one or all tunnels by using the clear statistics tunnel command, as the following example illustrates.

```
device#clear statistics tunnel 1
```

**Syntax: clear statistics tunnel** [ **tunnel ID** ]

To clear statistics for a specific tunnel, include the ID of that tunnel.

### *Configuring IPv6 session enforce check*

You can enable the IPv6 session enforce check by using the **ipv6-session-enforce-check** command. When an IPv6 packet arrives and this feature is enabled, the system tries to match the IPv6 packet source and destination address pair with the tunnel configured destination and source pair. If the pairs do not match, the packet is dropped in hardware.

NOTE
The CES 2000 Series and CER 2000 Series devices currently do not support the **ip-tunnel-policy** command or IPv6 **tunnels.**

To configure the IPv6 session enforce check, go to the IP tunnel policy context and enter the **ipv6-sessionenforce-check** command.

```
device(config)#ip-tunnel-policy
device(config-ip-tunnel-policy)#ipv6-session-enforce-check
```

**Syntax: [no] ipv6-session-enforce-check**

To disable the IPv6 session enforce check, use the no form of this command.

The system might prompt you to write the configuration to memory and reload the system. If the system has not yet allocated a source-ingress CAM partition, it prompts you to write the results of the current configuration to memory and reload the system.

The first-time execution of certain commands can prompt the allocation of a source-ingress CAM partition that is required by certain features. These commands are gre-session-enforce-check, ipv6-session-enforce-check, and accounting-enable. After this CAM partition is allocated, you do not need to do the memory write and reload after the first-time execution of the other two commands.

NOTE
The **ipv6-sessions-enforce-check** command is not supported for 6to4 automatic tunnels.

# GRE tunnels and MPLS handoff

Two MPLS networks can communicate using GRE tunnels with the handoff occurring at the same device.

Generic Routing Encapsulation (GRE) encapsulates data packets inside of a transport protocol and transmit the packets from one tunnel endpoint to another. Multiprotocol Label Switching (MPLS) is used in large data centers to control and forward traffic. In the situation where a data center exists without an MPLS connection, and MPLS traffic must be forwarded to another MPLS network, GRE tunnels can be deployed. The handoff to and from MPLS occurs at the same node as the GRE tunnel configuration.

Three main configuration steps are required for the handoff to and from a GRE tunnel to MPLS:

- Configure a GRE tunnel
- Configure MPLS LSP on the same node as the GRE tunnel ingress and egress nodes
- Configure an IP route to handoff the traffic from MPLS to the GRE tunnel and from the GRE tunnel to MPLS

NOTE
The GRE tunnel handoff to MPLS is only supported on MLXe and XMR devices. Not all interface cards on these devices are supported. See the Feature Support Matrix for more details.

## Restrictions for GRE tunnel handoff to MPLS

Some Multiprotocol Label Switching (MPLS) technologies are not supported by the GRE tunnel handoff to MPLS feature.

The following MPLS technologies are not supported:

- GRE MPLS handoff to Layer 3 Virtual Private Network (L3VPN)
- GRE MPLS handoff to Virtual Ethernet (VE) over Virtual Private LAN Service (VPLS)
- GRE MPLS handoff with Virtual Routing and Forwarding (VRF) over VE over VPLS

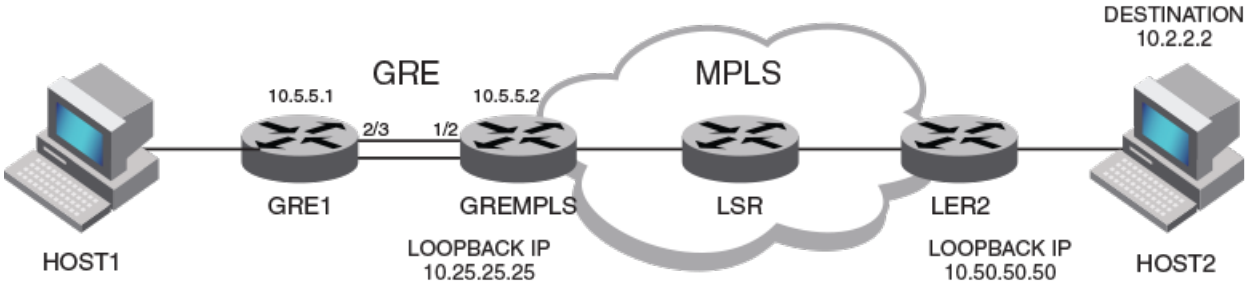
NOTE
The number of GRE tunnels supported is 512.

# GRE MPLS handoff without VRF configuration example

Configuration example for a handoff of MPLS data from a GRE tunnel. No VRF is configured.

This example configuration shows how to configure a GRE handoff to MPLS when Virtual routing and forwarding (VRF) is not configured. In the diagram the node at the edge of the MPLS network has both MPLS and GRE configured. MPLS traffic from Host1 is encapsulated at the GRE1 routing device and travels through the GRE tunnel for a handoff at the GREMPLS routing device where it can travel across the MPLS network. The Label Edge Router (LER) device forwards it to its destination at Host2.

**FIGURE 8** GRE MPLS tunnel diagram without VRF




## *GRE1 configuration*

The following example configures the GRE1 device in the above diagram as one endpoint of the GRE tunnel.

```
interface ethernet 2/3
  enable
  ip address 10.5.5.1/24

interface tunnel 1
  tunnel mode gre ip
  tunnel source 10.5.5.1
  tunnel destination 10.5.5.2
  ip address 10.10.3.1/24

ip route 10.2.2.0/24 10.10.3.2
```

### *GREMPLS configuration*

The following example configures the GREMPLS device in the above diagram as the other endpoint of the GRE tunnel. The configuration covers both GRE and MPLS because this is the device on which the MPLS handoff occurs.

```
ip route 10.2.2.0/24 10.50.50.50
ip route next-hop-enable-mpls

interface ethernet 1/2
  ip address 10.5.5.2/24

interface tunnel 1
 tunnel mode gre ip
  tunnel source 10.5.5.2
  tunnel destination 10.5.5.1
  ip address 10.10.3.2/24

interface loopback 1
  ip address 10.25.25.25/24

router mpls
  lsp to_dut4
   to 10.50.50.50
   shortcuts ospf
   tunnel-interface 1
   enable
```

### *LER2 configuration*

The following example configures the LER2 device in the above diagram.

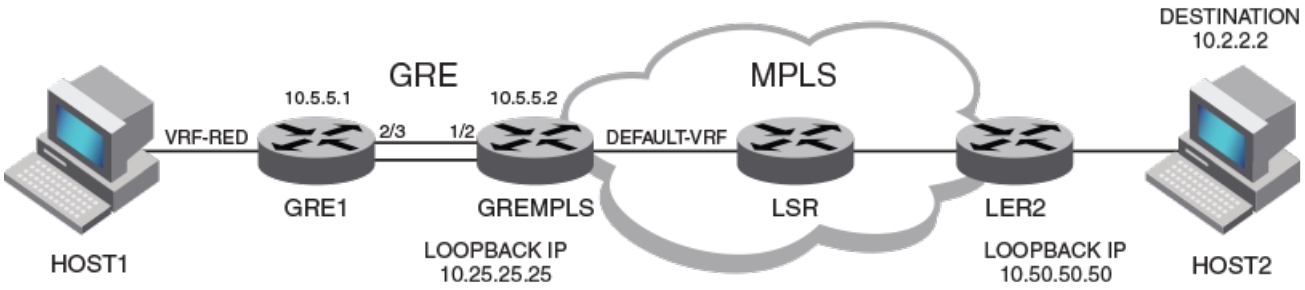
```
interface loopback 1
  ip address 10.50.50.50/24

router mpls
  lsp to_dut2
   to 10.25.25.25
   shortcuts ospf
   tunnel-interface 1
   enable
```

## GRE MPLS handoff with VRF configuration example

Configuration example for a handoff of Multiprotocol Label Switching (MPLS) data from a Generic Routing Encapsulation (GRE) tunnel. VRFs are configured.

This example configuration shows how to configure a GRE handoff to MPLS when Virtual Routing and Forwarding (VRF) is configured. In the diagram the node at the edge of the MPLS network has both MPLS and GRE configured. MPLS traffic from Host1 is encapsulated at the GRE1 routing device and travels through the GRE tunnel for a handoff at the GREMPLS routing device where it can travel across the MPLS network. The Label Edge Router (LER) device forwards it to its destination at Host2. The default VRF is configured as the next hop from the GREMPLS device and the VRF-RED red is configured at GRE1.

**FIGURE 9** GRE MPLS tunnel diagram with VRF




## *GRE1 configuration*

The following example configures the GRE1 routing device in the above diagram as one endpoint of the GRE tunnel.

```
vrf red
  rd 1:1
  address-family ipv4
  ip route 10.2.2.0/24 10.10.3.2
  exit-address-family
exit-vrf

interface ethernet 2/3
  enable
  vrf forwarding red
  ip address 10.5.5.1/24


interface tunnel 1
  tunnel mode gre ip
  tunnel source 10.5.5.1
  tunnel destination 10.5.5.2
  vrf forwarding red
  ip address 10.10.3.1/24
  keepalive 10 3
```

## GREMPLS configuration

The following example configures the GREMPLS device in the above diagram as the other endpoint of the GRE tunnel. The configuration covers both GRE and MPLS because this is the device on which the MPLS handoff occurs.

```
vrf red
  rd 2:2
  address-family ipv4
  ip route 10.2.2.0/24 next-hop-vrf default-vrf 10.50.50.50
  exit-address-family
exit-vrf

ip route next-hop-enable-mpls

interface tunnel 1
  tunnel mode gre ip
  tunnel source 10.5.5.2
  tunnel destination 10.5.5.1
  vrf forwarding red
  ip address 10.10.3.2/24
  keepalive 10 3

interface loopback 1
  ip address 10.25.25.25/24

router mpls
  lsp to_dut4
   to 10.50.50.50
   shortcuts ospf
   tunnel-interface 1
  enable
```

## LER2 configuration

The following example configures the LER2 device in the above diagram.

```
interface loopback 1
 ip address 10.50.50.50/24

router mpls
  lsp to_dut2
   to 10.25.25.25
   shortcuts ospf
   tunnel-interface 1
   enable
```

## Verifying GRE tunnel handoff to MPLS

The configuration of GRE tunnel handoff to MPLS can be verified using various show commands.

1. To view tunnel interface configuration information, use the **show interface tunnel** command. In the following example, information about the GRE tunnel and the VRF named Red is shown.

```
device(config)# show interface tunnel 1

Tunnel1 is up, line protocol is up
  Hardware is Tunnel
  Tunnel source  10.5.5.1
  Tunnel destination is 10.5.5.2
  Tunnel mode gre ip
  Configured BW is 0 kbps
  No port name
  Internet address is: 10.10.3.1/24
  Tunnel TOS 0, Tunnel TTL 255, Tunnel MTU 1476 bytes
  Keepalive is Enabled : Interval 10, No.of Retries 3
  Total Keepalive Pkts Tx: 2, Rx: 2
  VRF Forwarding: Red

  Tunnel Packet Statistics:
                        Unicast Packets                    Multicast Packets
   In-Port(s)     [Rcv-from-tnnl    Xmit-to-tnnl]    [Rcv-from-tnnl    Xmit-to-tnnl]
   e4/1 - e4/8    37537             0                0                 0
```

2. To view MPLS Label Switching Protocol (LSP) information, use the following command.

```
device(config)# show mpls lsp

Note: LSPs marked with * are taking a Secondary Path
                                Admin Oper  Tunnel   Up/Dn Retry Active
Name            To              State State Intf     Times No.   Path
To_dut4         10.50.50.50     UP    DOWN  tnl0     0     0     --
```

# Restart global timers

Restart contains two global timers that:

- Limit the amount of time used for re-syncing routes between the backup Management module and Interface modules (LPs) within the same chassis
- Allow a buffer time for protocols to converge and solve dependencies among each other

If the protocol-based restart features are configured when a Management module (MP) performs a switchover to the its backup, routes are maintained on the LPs through the protocol-based restart processes for a specified period of time while the new MP learns the network routes. Once the MP learns all of its routes, the routes from the MP are synced with the routes on the LPs.

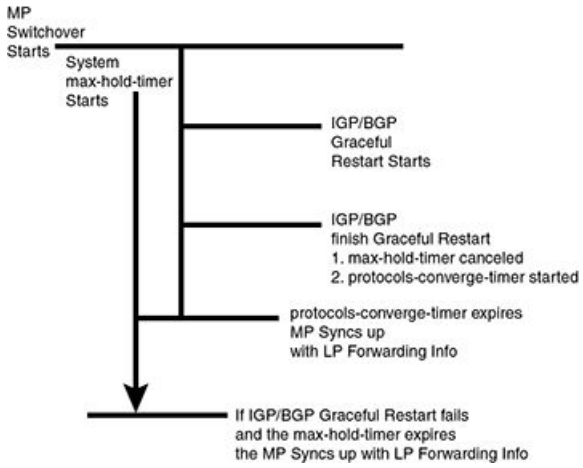
The two timers introduced here are called the **max-hold-timer** and the **protocols-converge-timer** .

The process of syncing routes between a new MP and its LPs using the new timers are illustrated in Figure 10 and described in the following steps.

1. The MP switchover from active to redundant MP begins.
2. The system **max-hold- timer** starts.
3. The IGP/BGP restart process begins.
4. If the IGP/BGP restart process is completed before the system **max-hold-timer** expires, the system **max-hold-timer** is cancelled and the **protocols-converge-timer** starts.

5. Once the **protocols-converge-timer** expires, the MP syncs up forwarding information with the LPs.
6. If the system **max-hold-timer** expires before the IGP/BGP restart process is completed, the MP syncs up forwarding information with the LPs at that time and the **protocols-converge-timer** is never started.

**FIGURE 10** MP to LP re-syncing process




# Configuring the graceful-restart max-hold-timer

This timer defines the maximum hold time before a management module syncs up new forwarding information to interface modules during the restart process. While the default value of 300 seconds will work in most cases, if a device is loaded with a very large number of routes and OSPF/BGP peering adjacencies you might want to fine-tune your device's performance by increasing this value.

The value of this timer can be set using the command shown in the following.

```
device(config)# graceful-restart max-hold-timer 500
```

**Syntax: [no]graceful-restart max-hold-timer hold-time**

The *hold-time* variable is the maximum number of seconds that a management routing module waits before it syncs up new forwarding information to the interface modules during a restart. The range for the hold time is 30 - 3600 seconds. The default time is 300 seconds.

# Graceful-restart protocols-converge-timer

This timer defines the time that this device waits for restarting protocols to converge at the final step in the restart process. In a heavily loaded system where BGP/OSPF/GRE/Static protocols can have a dependency on each other, their restart procedures may also depend on each other. This timer is to allow protocols to solve inter-dependencies after individual restart processes and before routing modules sync up new forwarding information to interface module. The default value of 5 seconds will work in most cases but if a system is heavily loaded and has protocols that depend on each other, you might want to fine-tune your system by increasing this value.

The value of this timer can be set using the command shown in the following.

```
device(config)# graceful-restart protocols-converge-timer 20
```

**Syntax: [no] graceful-restart protocols-converge-timer hold-time**

The *hold-time* variable is the maximum hold time in seconds before management routing modules sync up new forwarding information to interface modules during restart. The range of permissible values is 0 to 1200 seconds. The default value is 5 seconds.

# Configuring IP parameters

Some parameters can be configured globally while others can be configured on individual interfaces. Some parameters can be configured globally and overridden for individual interfaces.

## Configuring IP addresses

You can configure an IP address on the following types of the Extreme device interfaces:

- Ethernet port
- Virtual routing interface (also called a Virtual Ethernet or "VE")
- Loopback interface

By default, you can configure up to 24 IP addresses on each interface.

> NOTE
> After you configure a virtual routing interface on a VLAN, you cannot configure Layer 3 interface parameters on individual ports in the VLAN. Instead, you must configure the parameters on the virtual routing interface itself. Also, after an IP address is configured on an interface, the hardware is programmed to route all IP packets that are received on the interface. Consequently, all IP packets not destined for this device's MAC address are not bridged and are dropped.

The Extreme device supports both classical IP network masks (Class A, B, and C subnet masks, and so on) and Classless Interdomain Routing (CIDR) network prefix masks.

- To enter a classical network mask, enter the mask in IP address format. For example, enter "10.157.22.99 255.255.255.0" for an IP address with a Class-C subnet mask.
- To enter a prefix network mask, enter a forward slash ( / ) and the number of bits in the mask immediately after the IP address. For example, enter "10.157.22.99/24" for an IP address that has a network mask with 24 significant bits (ones).

By default, the CLI displays network masks in classical IP address format (example: 255.255.255.0). You can change the display to prefix format.

### *Assigning an IP address to an Ethernet port*

To assign an IP address to port 1/1, enter the following commands.

```
device(config)# interface ethernet 1/1
device(config-if-e1000-1/1)# ip address 10.45.6.1 255.255.255.0
```

> NOTE
> You also can enter the IP address and mask in CIDR format, as follows.

```
device(config-if-e10000-1/1)# ip address 10.45.6.1/24
```

**Syntax: [no] interface ethernet slot/port**

**Syntax: [no] ip address ip-addr ip-mask** | **ip-addr/mask-bits** [ **ospf-ignore** | **ospf-passive** | **secondary** ]

The **ospf-ignore** and **ospf-passive** parameters modify the Extreme device defaults for adjacency formation and interface advertisement. Use one of these parameters if you are configuring multiple IP subnet addresses on the interface but you want to prevent OSPF from running on some of the subnets:

- **ospf-passive** - disables adjacency formation with OSPF neighbors (but does not disable advertisement of the interface into OSPF). By default, when OSPF is enabled on an interface, the software forms OSPF router adjacencies between each primary IP address on the interface and the OSPF neighbor attached to the interface.

- **ospf-ignore** - disables OSPF adjacency formation and advertisement of the interface into OSPF. The subnet is completely ignored by OSPF.

Use the **secondary** parameter if you have already configured an IP address within the same subnet on the interface.

NOTE
When you configure more than one address in the same subnet, all but the first address are secondary addresses and do not form OSPF adjacencies.

## *Assigning an IP address to a loopback interface*

Loopback interfaces are always up, regardless of the states of physical interfaces. They can add stability to the network because they are not subject to route flap problems that can occur due to unstable links between this device and other devices.

You can configure up to 64 loopback interfaces on this device.

You can add up to 24 IP addresses to each loopback interface.

NOTE
If you configure the Extreme device to use a loopback interface to communicate with a BGP4 neighbor, you also must configure a loopback interface on the neighbor and configure the neighbor to use that loopback interface to communicate with the Extreme device.

To add a loopback interface, enter commands such as those shown in the following example.

```
device(config-bgp-router)# exit
device(config)# int loopback 1
device(config-lbif-1)# ip address 10.0.0.1/24
```

**Syntax: [no] interface loopback num**

For the syntax of the IP address, refer to Assigning an IP address to an Ethernet port on page 81.

## *Assigning an IP address to a virtual interface*

A virtual interface is a logical port associated with a Layer 3 Virtual LAN (VLAN) configured on this device.

NOTE
Other sections in this chapter that describe how to configure interface parameters also apply to virtual interfaces.

NOTE
The Extreme device uses the lowest MAC address on the device (the MAC address of port 1 or 1/1) as the MAC address for all ports within all virtual interfaces you configure on the device.

To add a virtual interface to a VLAN and configure an IP address on the interface, enter commands such as the following.

```
device(config)# vlan 2 name IP-Subnet_10.1.2.0/24
device(config-vlan-2)# untag e1/1 to 1/4
device(config-vlan-2)# router-interface ve1
device(config-vlan-2)# interface ve1
device(config-vif-1)# ip address 10.1.2.1/24
```

The first two commands create a Layer 3 protocol-based VLAN named "IP-Subnet_1.1.2.0/24" and add a range of untagged ports to the VLAN. The **router-interface** command creates virtual interface 1 as the routing interface for the VLAN. The last two commands change to the interface configuration level for the virtual interface and assign an IP address to the interface.

**Syntax: [no] router-interface ve num**

**Syntax: [no] interface ve num**

The *num* parameter specifies the virtual interface number. You can specify from 1 to the maximum number of virtual interfaces supported on the device. To display the maximum number of virtual interfaces supported on the device, enter the **show default values** command. The maximum is listed in the System Parameters section, in the Current column of the virtual-interface row.

For the syntax of the IP address, refer to Assigning an IP address to an Ethernet port on page 81.

### *Assigning a MAC address to a virtual interface*

By default, the Extreme device uses the MAC address of the first port (1 or 1/1) as the MAC address for all virtual routing interfaces configured on the device. You can specify a different MAC address for the virtual routing interfaces. If you specify another MAC address for the virtual routing interfaces, the address applies to all the virtual routing interfaces configured on the device. To specify the MAC address for virtual routing interfaces, enter commands such as the following.

```
device(config)# virtual-interface-mac aaaa.bbbb.cccc
device(config)# write memory
device(config)# end
device# reload
```

**Syntax: [no] virtual-interface-mac mac-addr**

Enter the MAC address in the following format: HHHH.HHHH.HHHH

> NOTE
> You must save the configuration and reload the software to place the change into effect.

### *Deleting an IP address*

To delete an IP address, enter a command such as the following.

```
device(config-if-e1000-1/1)# no ip address 10.1.2.1
```

This command deletes IP address 10.1.2.1. You do not need to enter the subnet mask.

To delete all IP addresses from an interface, enter the following command.

```
device(config-if-e1000-1/1)# no ip address *
```

**Syntax: no ip address ip-addr**

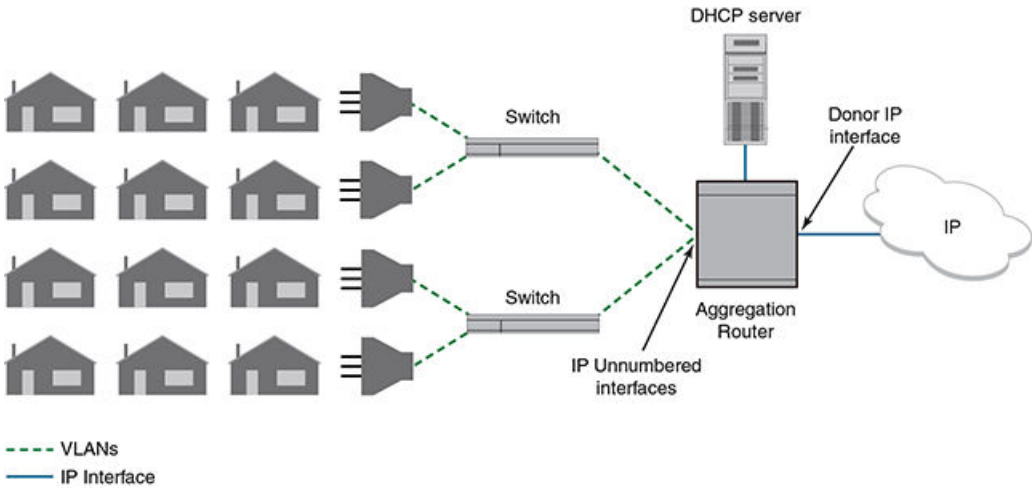
# IP Unnumbered Interfaces

The IP Unnumbered Interfaces feature saves IPv4 address space by allowing unnumbered interfaces to inherit the IP address of a donor interface, thus allowing all ports to share the same subnet. This feature not only preserves IP addresses, but also reduces the IP routing table size. This feature also provides ARP suppression (reducing the number of ARP requests sent to hosts) on unnumbered interfaces, thus increasing the number of hosts that are supported under the same subnet.

- The *donor interface* is the interface with an IP address configured on it.
- The *unnumbered interface* is the interface with no IP address configured on it. The unnumbered interface inherits the IP address of the donor interface.

For example, consider a DSLAM deployment scenario with multiple users connected to a device (refer to Figure 11). Instead of configuring IP addresses for every VE on the Extreme device, you can designate one VE as the donor interface and configure all the other VEs to inherit the IP address of the donor VE interface.

**FIGURE 11** IP Unnumbered Interfaces feature




The donor interface must be one of the following:

- Loopback interface
- VE interface
- Ethernet interface (can be part of a LAG interface; must be untagged if in a VLAN)

The unnumbered interfaces can be the following:

- VE interface
- Ethernet interface (must be untagged if in a VLAN)

# Configuring an unnumbered interface

To enable an unnumbered interface to inherit the IP address of a donor interface, enter commands such as the following:

```
device(config)# interface ve 10
device(config-vif-10)# ip unnumbered ve 9
```

The commands enable interface ve 10 to inherit the IP address of ve 9. Interface ve 10 is the unnumbered interface and interface ve 9 is the donor interface.

**Syntax: [no] ip unnumbered** [ **ethernet** *slot/port* | **ve** *num* | **loopback** *num* ]

The **ethernet** *slot/port* parameter specifies the donor interface by an Ethernet port number.

The **ve** *num* parameter specifies the donor interface by virtual interface number.

The **loopback** *num* parameter specifies the donor interface by loopback interface number.

Use the **no ip unnumbered** command to remove the IP address from the unnumbered interface.

NOTE
You do not need to configure an interface to be a donor interface. An interface becomes a donor interface automatically when an unnumbered interface inherits its IP address.

# Displaying unnumbered interfaces

The **show ip interface** command displays information about unnumbered interfaces.

In the following example, note that interfaces ve 9 and ve 10 have the same IP address. Interface ve 10 is an unnumbered interface, as indicated by the **U** in the **Flag** column.

```
device# show ip interface

Interface       IP-Address      OK?  Method Status    Protocol VRF           FLAG
mgmt 1          10.21.108.35    YES  NVRAM  up        up       default-vrf
ve 6            6.1.1.1         YES  NVRAM  up        up       default-vrf
ve 9            1.1.1.1         YES  NVRAM  up        up       default-vrf
ve 10           1.1.1.1         YES  NVRAM  up        up       default-vrf   U
```

In the following example, the first highlighted line indicates that interface ve 10 is an unnumbered interface, inheriting the IP address of ve 9, which is the donor interface.

```
device# show ip interface ve 10
Interface Ve 10
  members: ethe 4/1
  active: ethe 4/1
  port enabled
  port state: UP
  ip address: 1.1.1.1/24
  Unnumbered interface, Using IP address of  ve 9
  unnumbered arp-suppression is enabled
  Port belongs to VRF: default-vrf
(output truncated)
```

The second highlighted line indicates whether ARP suppression is enabled or disabled. Refer to ARP suppression on unnumbered interfaces on page 85 for information about ARP suppression.

# ARP suppression on unnumbered interfaces

By default, ARP suppression is enabled on unnumbered interfaces, and ARP requests are not sent.

If many VLANs belong to the same subnet, ARP suppression avoids an ARP storm.

Donor interfaces continue to send out ARP requests because ARP suppression is disabled on donor interfaces.

ARP suppression is achieved by implementing Configuring ARP suppression for unnumbered interfaces on page 86 and performing one of the following:

- Configure DHCP option 82 (recommended)

  This configuration must be enabled on each VLAN belonging to the donor or unnumbered interface. When DHCP option 82 is enabled, the ARP request is sent only to the corresponding VLAN (identified in the Dynamic ARP Inspection (DAI) table) instead of all the unnumbered VLANs. For details of DHCP option 82 and the DAI table, refer to DHCP option 82 insertion on page 384.

- Configure static DAI entries

  You must configure static DAI entries for scenarios where the host is not discovered through DHCP, such as when a host is provided with a static IP address. Refer to Configuring DAI on page 34 for instructions.

  NOTE

  If you enable ARP suppression on a donor interface, you must configure static Dynamic ARP Inspection (DAI) entries for protocol neighbor IP addresses to ensure that protocol operations on the donor interface succeed.

### *Configuring ARP suppression for unnumbered interfaces*

To enable or disable ARP suppression on an unnumbered interface, enter commands such as the following:

```
device(config)# interface ve 9
device(config-vif-9)# ip unnumbered-arp-suppression
device(config)# interface ve 10
device(config-vif-10)# no ip unnumbered-arp-suppression
```

The commands enable ARP suppression on VE 9 and disable ARP suppression on VE 10. To fully achieve ARP suppression, configure one of the following:

- DHCP option 82 (refer to Enabling DHCP snooping on a VLAN on page 382)
- Static DAI entries (refer to Configuring DAI on page 34)

**Syntax: [no] ip unnumbered-arp-suppression**

Use the **no ip unnumbered-arp-suppression** command to disable ARP suppression on the interface.

This command is applicable only to donor and unnumbered interfaces. It has no effect on other interfaces.

You can use the **show ip interface** command to display whether ARP suppression is enabled or disabled, as shown in Displaying unnumbered interfaces on page 85.

### *ARP for non-DHCP hosts on IP unnumbered interfaces*

You can override ARP suppression for non-DHCP hosts on IP unnumbered interfaces.

( MLX Series and XMR Series) Even if ARP suppression is configured on an IP unnumbered interface, ARP resolution still occurs in the following scenario:

- Non-DHCP host and client (static IP address assigned)
- Static route installed for the IP address (32 bit subnet mask).

For CER 2000 Series and CES 2000 Series devices, you can override ARP suppression by configuring static ARP entries. For details, refer to Static ARP entries on page 31.

## Caveats and limitations for IP Unnumbered Interfaces

- The IP Unnumbered Interfaces feature is not supported for IPv6 addresses.
- Multicast and MPLS protocols are not supported on donor interfaces.
- Routing protocols, multicast, and MPLS are not supported on the unnumbered interfaces.
- The IP Unnumbered Interfaces feature is supported only on the default VRF. Both donor and the unnumbered interfaces must be in the default VRF.
- If the donor interface is down (link state or administrative state), a ping to the donor IP address fails, even if the unnumbered interfaces that inherited the IP address are up.
- VRRP and VRRP-E operations are not supported on unnumbered interfaces.
- RPF strict mode is not supported on unnumbered interfaces.

## Configuration considerations for IP Unnumbered Interfaces

- You can have multiple donor interfaces in the device. A donor interface can have multiple unnumbered interfaces inheriting its IP address. An unnumbered interface can have only one donor interface.

- You can configure multiple primary and multiple secondary IP addresses on the donor interface. The unnumbered interface inherits all primary and secondary addresses of the donor interface.
- The unnumbered interface inherits only the IP address from the donor interface. All other donor interface configurations are not passed on to the unnumbered interface. You must configure other features, such as IP Source Guard and forwarding of directed broadcasts, on the unnumbered interfaces separately.
- The following routing protocols are supported on the donor interface:
  - Open Shortest Path First (OSPF)
  - Intermediate System - Intermediate System (IS-IS)
  - Routing Information Protocol (RIP)
  - Border Gateway Protocol (BGP)
- If DHCP clients are configured on an unnumbered interface, then DHCP option 82 must be configured on that interface; otherwise, the DHCP client cannot get the IP address from the DHCP server.
- If reachability is needed between two hosts within the same subnet, you must configure local proxy ARP on the unnumbered interfaces. Refer to Enabling local proxy ARP on page 30 for more information.

### *Static route considerations for unnumbered interfaces*

- If you configure a static route with an unnumbered interface or donor interface as the next hop, it is recommended that you configure a standard static route instead of an interface-based static route.
- If you configure an interface-based static route on a donor or unnumbered interface, you must ensure that ARP suppression is disabled on the interface.

### *DHCP host subnet selection*

If the donor interface is configured with multiple subnets, and the DHCP clients need to receive addresses in a specific subnet, use the **ip bootp-gateway** command to select the local donor interface IP address of the specific subnet.

This functionality can be used when the DHCP clients are moved from one subnet to another subnet.

Refer to Changing the IP address used for stamping BootP or DHCP requests on page 120 for instructions on using the **ip bootp-gateway** command. Note that the **ip bootp-gateway** command is used only when the hosts are DHCP hosts.

### *Support for other features*

IP address configurations are the only configurations that the unnumbered interfaces inherit from the donor interface.

All other configurations (such as ICMP, ACLs, DHCP, and PBR) that are configured on the donor interface apply only to the donor interface and are not inherited by the unnumbered interfaces. You must configure these features separately on the unnumbered interfaces.

## Sample configuration for IP Unnumbered Interfaces

This example shows how to configure IP unnumbered interfaces with a DHCP server. In this example, loopback 1 is the donor interface, and ve 20 and ve 30 are the unnumbered interfaces.

After configuring an IP address on the donor interface, configure the two VE interfaces to inherit the IP address of the donor interface as shown in the following example.

```
device(config)# interface loopback 1
device(config-lbif-1)# ip address 10.10.10.1/24
device(config-lbif-1)# vlan 20
```

```
device(config-vlan-20)# router-interface ve 20
device(config-vlan-20)# interface ve 20
device(config-vif-20)# ip unnumbered loopback 1
device(config-vif-20)# vlan 30
device(config-vlan-30)# router-interface ve 30
device(config-vlan-30)# interface ve 30
device(config-vif-30)# ip unnumbered loopback 1
```

Configure the DHCP server. In this example, the DHCP server address is 10.40.40.4.

```
device(config-vif-30)# interface ethernet 1/2
device(config-if-e1000-1/2)# ip address 10.40.40.1/24
device(config-if-e1000-1/2)# dhcp-snooping-trust
```

Configure the DHCP server address in the unnumbered interfaces.

```
device(config-if-e1000-1/2)# interface ve 20
device(config-vif-20)# ip helper-address 10.40.40.4
device(config-vif-20)# interface ve 30
device(config-vif-30)# ip helper-address 10.40.40.4
device(config-vif-30)# exit
```

Configure DHCP option 82 in the unnumbered interface VLANs.

```
device(config)# ip dhcp-snooping vlan 20 to 30 insert-relay-information
device (config)# ip dhcp-snooping vlan 1 insert-relay-information
```

# Support for a 31-bit subnet mask on point-to-point networks

NOTE
The configuration of an IPv4 address with a 31-bit subnet mask is supported on MLX Series, XMR Series, and CER 2000 Series and CES 2000 Series devices.

In an effort to conserve IPv4 address space, a 31-bit subnet mask can be assigned to point-to-point networks. Support for an IPv4 address with a 31-bit subnet mask is described in RFC 3021. Previously, four IP addresses with a 30-bit subnet mask were allocated on point-to-point networks. A 31-bit subnet mask uses only two IP addresses; all zero bits and all one bits in the host portion of the IP address. The two IP addresses are interpreted as host addresses, and do not require broadcast support because any packet that is transmitted by one host is always received by the other host at the receiving end. Therefore, directed broadcast on a point-to-point interface is eliminated. Also, a broadcast address with all one bits in the host portion of the IP address is not allocated for point-to-point interface configuration.

NOTE
IP-directed broadcast CLI configuration at the global level, or the per- interface level, is not applicable on interfaces configured with a 31-bit subnet mask IP address.

## *Configuring an IPv4 address with a 31-bit subnet mask*

To configure an IPv4 address with a 31-bit subnet mask, enter the following commands.

NOTE
You can configure an IPv4 address with a 31-bit subnet mask on any interface (for example, Ethernet, loopback, VE, or tunnel interfaces), and on all VRFs (default and non-default VRFs).

```
device(config)# interface ethernet 1/5
device(config-if-e1000-1/5)# ip address 10.9.9.9 255.255.255.254
```

You can also enter the IP address and mask in the Classless Interdomain Routing (CIDR) format, as follows.

```
device(config-if-e1000-1/5)# ip address 10.9.9.9/31
```

**Syntax: [no] ip address** *ip-addressip-mask*

**Syntax: [no] ip address** *ip-address/subnetmask-bits*

The *ip-address* variable specifies the host address. The *ip-mask* variable specifies the IP network mask. The *subnet mask-bits* variable specifies the network prefix mask.

To disable configuration for an IPv4 address with a 31-bit subnet mask on any interface, use the **no** form of the command.

You cannot configure a secondary IPv4 address with a 31-bit subnet mask on any interface. The following error message is displayed when a secondary IPv4 address with a 31-bit subnet mask is configured.

```
device(config-if-e1000-1/5)#ip address 10.8.8.8/31 secondary
IP/Port: Errno(10) Cannot assign /31 subnet address as secondary
```

### *Displaying the configuration for an IPv4 address with a 31-bit subnet mask*

To display the interface running configuration when an IPv4 address with a 31-bit subnet mask is configured, enter the following command at any level of the CLI.

```
device(config-if-e1000-1/5)# show run interface ethernet 1/5
interface ethernet 1/5
enable
ip address 10.2.2.3/31
ip address 10.4.4.4/31
```

In the previous example, interface ethernet 1/5 is assigned two IPv4 addresses (10.2.2.3/31 and 10.4.4.4/31) with a 31-bit subnet mask.

To display the configuration for an IPv4 address with a 31-bit subnet mask on a virtual ethernet (VE) interface, enter the following command at any level of the CLI. In the example below, VE interface 10 is assigned two IPv4 addresses (10.25.25.255/31 and 10.168.32.0/31) with a 31-bit subnet mask.

```
device(config-if-e1000-2/5)#show run interface ve 10
interface ve 10
ip ospf area 0
ip address 10.25.25.255/31
ip address 10.168.32.0/31
```

**Syntax: show run interface** [ **ethernet** *slot/port* | **loopback** *number* | **tunnel** *number* | **ve** *number* ]

The **show ip route** command displays routes that are directly connected with interfaces configured with IPv4 addresses with a 31-bit subnet mask.

```
device(config-if-e1000-2/5)# show ip route
Total number of IP routes: 21
       Destination        Gateway        Port         Cost     Type  Uptime
1       10.2.2.2/31        DIRECT         eth 1/5      0/0        D    2h19m
2       10.4.4.4/31        DIRECT         eth 1/5      0/0        D    2h19m
3       10.25.25.254/31    DIRECT         ve 10        0/0        D    2h25m
4       10.168.32.0/31     DIRECT         ve 10        0/0        D    2h25m
```

**Syntax: show ip route**

# Enabling hardware forwarding of IP option packets based on Layer 3 destination

The IP option field in an IP header is variable in length. A packet can have zero or more options and an option can have either of the following forms:

- a single octet of option-type

- an option-type octet, an option-length octet, and option-data octets

The option-type octet consists of the following three fields:

- 1 bit copied flag
- 2 bits option class
- 5 bits option number

By default, IP option packets are sent to the CPU for forwarding. When configured on a physical interface, the **ignore-options** command directs the device to ignore all options in IP option packets that are received at the configured port. These packets are then treated as if there were no options configured and forwarded based on their Layer-3 destination. The **ignore-options** command is configured as shown in the following.

```
device(config)# interface ethernet 1/1
device(config-if-e1000-1/1)# ignore-options
```

**Syntax: [no] ignore-options**

This command only applies to IP option packets in the default VRF.

When the **ignore-options** command is configured on a port, RSVP router alert packets incoming on that port will not be sent to the CPU. Consequently, MPLS should not be configured on a physical port where the **ignore-options** command is configured.

## *Using the ignore-options command in a LAG configuration*

The **ignore-options** command can be used on a LAG but it must apply to all ports on the LAG. This applies to both static and LACP LAGs as described in the following:

### Configuring the ignore-options command on a static LAG

To configure the **ignore-options** command on a static LAG, each port on the LAG must be configured with the command. You can do this by configuring the command on each port before the LAG configuration or configuring the **ignore-options** command on the primary port of the LAG which automatically applies the command to all ports on the LAG as shown in the following.

```
device(config)# trunk e 3/1 to 3/4
trunk transaction done.
device(config-trunk-3/1-3/4)# exit
device(config)# interface ethernet 3/1
device(config-if-e1000-3/1)# ignore-options
```

If the LAG is removed, the **ignore-options** command will be propagated to all ports that were previously in the LAG.

If you try to create a LAG where some of the ports have the **ignore-options** command configured and some do not, the LAG will not be allowed as shown in the following example.

```
device(config)# trunk e 3/1 to 3/2
port 3/1 ignore-options is Enabled, but port 3/2 ignore-options is Disabled
Error: port 3/1 and port 3/2 have different configurations
trunk transaction failed:  trunk Config Vetoed
```

### Configuring the ignore-options command on a LACP LAG

Just as with static LAGs, if you want to configure the **ignore-options** command on an LACP LAG, the command must be enabled on all ports within the LAG. If it is not, the LACP LAG will not be accepted as shown in the following.

```
device(config)#lag sta_lag static
device(config-lag-sta_lag)#ports e 1/3 to 1/4
device(config-lag-sta_lag)#primary-port 1/3
device(config-lag-sta_lag)#deploy
```

```
device(config-lag-sta_lag)#int e 1/3
device(config-if-e1000-1/3)#ignore-options
device(config)#lag sta_lag static
device(config-lag-sta_lag)#ports e 1/3 e 1/4
device(config-lag-sta_lag)#primary-port 1/3
device(config-lag-sta_lag)#deploy
port 1/3 ignore-options is Enabled, but port 1/4 ignore-options is Disabled
Error: port 1/3 and port 1/4 have different configurations
LAG sta_lag deployment failed!
device(config)#int e 1/3
device(config-if-e1000-1/3)#ignore-options
device(config-if-e1000-1/3)#lag dyn_lag dynamic
device(config-lag-dyn_lag)#ports e 1/3 e 1/4
device(config-lag-dyn_lag)#primary-port 1/3
device(config-lag-dyn_lag)#deploy
port 1/3 ignore-options is Enabled, but port 1/4 ignore-options is Disabled
Error: port 1/3 and port 1/4 have different configurations
LAG dyn_lag deployment failed!
```

# Configuring domain name server (DNS) resolver

The DNS resolver lets you use a host name to perform Telnet, ping, and traceroute commands. You can also define a DNS domain on this device and thereby recognize all hosts within that domain. After you define a domain name, the device automatically appends the appropriate domain to the host and forwards it to the domain name server.

For example, if the domain "newyork.com" is defined on a device and you want to initiate a ping to host "NYC01" on that domain, you need to reference only the host name in the command instead of the host name and its domain name. For example, you could enter either of the following commands to initiate the ping.

```
device# ping nyc01
device# ping nyc01.newyork.com
```

Multiple DNS queries can be executed simultaneously, making it possible for the device to run multiple simultaneous Telnet, ping or traceroute commands using host names.

## *Defining an IPv4 DNS entry*

You can define up to four DNS servers for each DNS entry. The first entry serves as the primary default address. If a query to the primary address fails to be resolved after three attempts, the next gateway address is queried (also up to three times). This process continues for each defined gateway address until the query is resolved. The order in which the default gateway addresses are polled is the same as the order in which you enter them.

Suppose you want to define the domain name of abc.com on a device and then define four possible default DNS gateway addresses. To do so using IPv4 addressing, you would enter the following commands.

```
device(config)# ip dns domain-name abc.com
device(config)# ip dns server-address 10.157.22.199 10.96.7.15 10.95.7.25 10.98.7.15
```

**Syntax: [no] ip dns server-address ip-addr** [ **ip-addr** ] [ **ip-addr** ] [ **ip-addr** ]

In this example, the first IP address in the **ip dns server-address** command becomes the primary gateway address and all others are secondary addresses. Because IP address 10.98.7.15 is the last address listed, it is also the last address consulted to resolve a query.

## *DNS queries of IPv4 and IPv6 DNS servers*

IPv4 and IPv6 DNS record queries search through IPv4 and IPv6 DNS servers as described in the following:

**For IPv4 DNS record queries:**

- Loop thru all configured IPv4 DNS servers,

- If no IPv4 DNS servers were configured, then loop through all configured IPv6 DNS servers (if any).

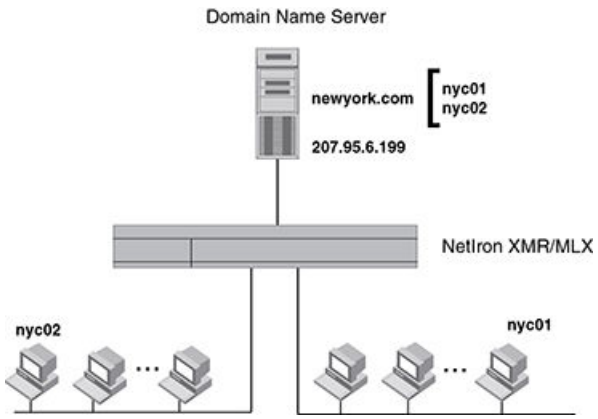
**For IPv6 DNS record queries:**

- Loop thru all configured IPv6 DNS servers,
- If no IPv6 DNS servers were configured, then loop through all configured IPv4 DNS servers (if any).

### *Using a DNS name to initiate a trace route*

Suppose you want to trace the route from this device to a remote server identified as NYC02 on domain newyork.com.

**FIGURE 12** Querying a host on the newyork.com domain




Because the newyork.com domain is already defined on the Extreme device, you need to enter only the host name, NYC02, as noted below.

```
device# traceroute nyc02
```

**Syntax: [no] traceroute host-ip-addr** [ **maxttl value** ] [ **minttl value** ] [ **numeric** ] [ **timeout value** ] [ **source-ip ip addr** ]

The only required parameter is the IP address of the host at the other end of the route.

After you enter the command, a message indicating that the DNS query is in process and the current gateway address (IP address of the domain name server) being queried appear on the screen.

```
Type Control-c to abort
Sending DNS Query to 10.157.22.199
Tracing Route to IP node 10.157.22.80
To ABORT Trace Route, Please use stop-traceroute command.
 Traced route to target IP node 10.157.22.80:
   IP Address          Round Trip Time1    Round Trip Time2
  10.95.6.30         93 msec             121 msec
```

NOTE

In the above example, 10.157.22.199 is the IP address of the domain name server (default DNS gateway address), and 10.157.22.80 represents the IP address of the NYC02 host.

# Using Telnet and Secure Shell

Up to six inbound and five outbound Telnet connections can be supported simultaneously by the Extreme device. The Extreme device also supports Secure Shell (SSH) access to management functions.

# Changing the encapsulation type for IP packets

The Extreme device encapsulates IP packets into Layer 2 packets, to send the IP packets on the network. A Layer 2 packet is also called a MAC layer packet or an Ethernet frame. The MAC address of the Extreme device interface sending the packet is the source address of the Layer 2 packet. The Layer 2 packet's destination address can be one of the following:

- The MAC address of the IP packet's destination. In this case, the destination device is directly connected to the Extreme device.
- The MAC address of the next-hop gateway toward the packet's destination.
- An Ethernet broadcast address.

The entire IP packet, including the source address, destination address, other control information, and the data, is placed in the data portion of the Layer 2 packet. Typically, an Ethernet network uses one of two different formats of Layer 2 packet:

- Ethernet II
- Ethernet SNAP (also called IEEE 802.3)

The control portions of these packets differ slightly. All IP devices on an Ethernet network must use the same format. The Extreme device uses Ethernet II by default. You can change the IP encapsulation to Ethernet SNAP on individual ports if needed.

NOTE
All devices connected to the Extreme device port must use the same encapsulation type.

To change the IP encapsulation type on interface 1/5 to Ethernet SNAP, enter the following commands.

```
device(config)# int e 1/5
device(config-if-e1000-1/5)# ip encapsulation snap
```

**Syntax: [no] ip encapsulation snap | ethernet-2**

# Setting the maximum frame size globally

You can set the default maximum frame size to control the maximum size of Ethernet frames that the Ethernet MAC framers will accept or transmit. The size is counted from the beginning of Ethernet header to the end of CRC field. The default maximum frame size must be greater than an IP MTU value set using the Globally changing the IP MTU on page 95.

To set a maximum frame size that applies to the device for Ethernet ports, enter a command such as the following.

```
device(config)# default-max-frame-size 2000
device(config)# write memory
device(config)# reload
```

**Syntax: [no] default-max-frame-size bytes**

Enter 1298 - 9216 for *bytes*. On XMR Series and MLX Series devices, the default is 1548 bytes for Ethernet ports.

On CES 2000 Series devices, the *bytes* variable specifies a even number of bytes between 1298 - 9216. The default value is 1548 bytes.

NOTE
You must run the **write memory** command and reload the Extreme device for the **default-max-frame-size** command to take effect.

NOTE
In a VLAN-tagged port, the device can accept a frame size up to the default maximum frame size with or without the VLAN-tagged frame. However, it can only transmit a frame size up to the default maximum frame size plus vlan tag 4 bytes.

# Changing the MTU

The IP MTU is the maximum length of an IP packet that a Layer 2 packet can contain. If an IP packet is larger than the IP MTU allowed by the Layer 2 packet, the Extreme device fragments the IP packet into multiple parts that will fit into Layer 2 packets, and sends the parts of the fragmented IP packet separately, in different Layer 2 packets. The device that receives the multiple fragments of the IP packet reassembles the fragments into the original packet. The default IP MTU is 1500 bytes for Ethernet II packets. You can change the IP MTU globally or for individual IP interfaces. You can increase the IP MTU size to accommodate large packet sizes, such as jumbo packets, globally or on individual IP interfaces. However, IP MTU cannot be set higher than the maximum frame size, minus 18.

NOTE
For multicast data traffic, frames are not fragmented and the IP MTU setting is ignored.

For jumbo packets, the Extreme device supports hardware forwarding of Layer 3 jumbo packets. Layer 3 IP unicast jumbo packets received on a port that supports the frame's IP MTU size and forwarded to another port that also supports the frame's IP MTU size are forwarded in hardware.

NOTE
Policy Based Routing (PBR) currently does not support this IP MTU feature.

## *Configuration considerations for increasing the IP MTU:*

- The maximum value of an IP MTU cannot exceed the configured maximum frame size, minus 18. For example, global IP MTU cannot exceed the value of **default-max-frame-size** , minus 18 bytes. IP MTU for an interface cannot exceed the value of the maximum frame size configured, minus 18 bytes. The 18 bytes are used for Ethernet header and CRC.
- When you increase the IP MTU size of for an IP interface, the increase uses system resources. Increase the IP MTU size only on the IP interfaces that need it. For example, if you have one IP interface connected to a server that uses jumbo frames and two other IP interfaces connected to clients that can support the jumbo frames, increase the IP MTU only on those three IP interfaces. Leave the IP MTU size on the other IP interfaces at the default value (1500 bytes). Globally increase the IP MTU size only if needed.
- The difference between IP MTU and **default-max-frame size** should be as follows.
  - 18 bytes for untagged packets
  - 22 bytes for single-tagged packets and
  - 26 bytes for dual-tagged packets

## *How To determine the actual MTU value*

An IPv4 interface can obtain it's MTU value from any of the following sources:

- Default IP MTU setting
- Global MTU Setting
- Interface MTU Setting

An interface determines its actual MTU value through the process described below.

1. If an IPv4 Interface MTU value is configured, that value will be used.
2. If an IPv4 Interface MTU value is not configured and an IPv4 Global MTU value is configured, the configured global MTU value will be used.
3. If neither an IPv4 Interface MTU value or an IPv4 Global MTU value are configured, the default IPv4 MTU value of 1500 will be used.

### Globally changing the IP MTU

To globally enable jumbo support on all IP interfaces, enter commands such as the following.

```
device(config)# ip global-mtu 5000
device(config)# write memory
```

**Syntax: [no] ip global-mtu bytes**

The *bytes* parameter specifies the maximum IP packet size to be forwarded on a port. You may enter any number within the range of 576 - 9198. However, this value must be 18 bytes less than the value of the global maximum frame size.

NOTE
The global IP MTU change does not get applied to IP tunnel interfaces such as GRE interface. The MTU for these interfaces has to be changed on interface level.

### Changing the maximum transmission unit on an individual interface

By default, the maximum IP MTU sizes are as follows:

- 1500 bytes - The maximum for Ethernet II encapsulation

NOTE
The IP MTU configured at the IP interface level takes precedence over the IP MTU configured at the global level for that IP interface.

To change the IP MTU for interface 1/5 to 1000, enter the following commands.

```
device(config)# int e 1/5
device(config-if-e10000-5)# ip mtu 1000
```

**Syntax: [no] ip mtu bytes**

The *bytes* variable specifies the IP MTU. However, the value of IP MTU on an interface cannot exceed the configured value **default-max-frame-size** , minus 18 bytes. The default IP MTU for Ethernet II packets is 1500.

If the interface is part of a VLAN, then ensure that you change the IP MTU only at the VE interface and not at the physical port. To change the IP MTU at the VE interface, enter the following commands:

```
device(config)# int ve 103
device(config-vif-103)# ip mtu 1000
```

NOTE
All member ports of a VLAN will have the same IP MTU value as the VE interface.

## Changing the router ID

In most configurations, this Extreme device has multiple IP addresses, usually configured on different interfaces. As a result, a device's identity to other devices varies depending on the interface to which the other device is attached. Some routing protocols, including OSPF and BGP4, identify a device by just one of the IP addresses configured on the device, regardless of the interfaces that connect the devices. This IP address is the router ID.

NOTE
RIP does not use the router ID.

NOTE
If you change the router ID, all current BGP4 sessions are cleared.

By default, the router ID on the Extreme device is one of the following:

- If the device has loopback interfaces, the default router ID is the IP address configured on the lowest numbered loopback interface configured on the device. For example, if you configure loopback interfaces 1, 2, and 3 as follows, the default router ID is 10.9.9.9/24:
  - Loopback interface 1, 10.9.9.9/24
  - Loopback interface 2, 10.4.4.4/24
  - Loopback interface 3, 10.1.1.1/24
- If the IP address from loopback1 interface (lowest numbered loopback interface) is removed, the next lowest loopback interface IP address is selected as router-id.
- If a loopback interface is not configured, then the lowest IP address configured over the physical interface is selected as the router ID.

If you prefer, you can explicitly set the router ID to any valid IP address. The IP address should not be in use on another device in the network.

You can set a router ID for a specific VRF as described within this section. In order to make the route ID calculation more deterministic, the device calculates the router-id value during bootup and does not calculate or change the router-id value unless the IP address used for the router-id value on the device is deleted, or the **clear router-id** command is issued. Additionally, setting a router-id value overrides the existing router-id value and takes effect immediately. Once a router-id value set by a user is removed using the **no ip router-id** command, the device will again recalculate the router-id value based on current information.

> NOTE
> The Extreme device uses the same router ID for both OSPF and BGP4. If the device is already configured for OSPF, you may want to use the router ID that is already in use on the device rather than set a new one. To display the router ID, enter the **show ip** command at any CLI level.

To change the router ID, enter a command such as the following.

```
device(config)# ip router-id 10.157.22.26
```

**Syntax: [no] ip router-id ip-addr**

The *ip-addr* can be any valid, unique IP address.

To set the router ID within a VRF, enter a command such as the following.

```
device(config)# vrf blue
device(config-vrf-blue)# ip router-id 10.157.22.26
```

**Syntax: [no] ip router-id ip-addr**

> NOTE
> The command for setting the router ID for a specified VRF is exactly the same as for the default VRF. The only difference is that when setting it for a specific VRF, the **ip router-id** command is configured within the VRF as shown in the example.

> NOTE
> You can specify an IP address used for an interface, but do not specify an IP address in use by another device.

# Recalculating the router ID

You can use the **clear ip router-id** command to direct a device to recalculate the IP router ID. This can be done for the default VRF or for a specified VRF, as shown in the following.

```
device(config)# clear ip router-id
```

**Syntax: clear ip router-id** [ **vrf vrf-name** ]

Using this command without the **vrf** option recalculates the IP router ID for the default VRF.

You can use the **vrf** option to recalculate the IP router ID for a specific VRF that is specified by the *vrf-name* variable.

# IPv6 ND Global Router Advertisement Control

IPv6 ND Global Router Advertisement Control allows for disabling sending out router advertisements at the global system level. The **no ipv6 nd global-suppress-ra** command at the interface level allows the user to disable and enable the sending of the ND Router Advertisement on an interface. By default, the sending of ND Router Advertisement (RA) is enabled on all interfaces, except for the tunnel and loopback interfaces, providing that the IPv6 Unicast Routing is enabled and the interfaces are active for IPv6.

The IPv6 ND Global Router Advertisement Control gives the ability to quickly turn off the sending of IPv6 ND Router Advertisement message on all IPv6 enabled interfaces.

By default,

- The ND Router Advertisement is enabled.
- Interface is enabled to send ND Router Advertisements.
- The **ipv6 nd suppress-ra** and **ipv6 nd send-ra** interface commands, when configured, override the system and VRF global **ipv6 nd global-suppress-ra** command.

Users sometimes require the ability to quickly turn off the sending of IPv6 ND Router Advertisement message on all IPv6 enabled interfaces. This is achieved by providing the following additional configuration command at system and VRF level:

```
device(config-vrf-red-ipv6)[no]ipv6 nd global-suppress-ra
```

The **ipv6 nd send-ra** command is a new interface level command added as part of this enhancement. This allows the user to configure the sending of RA messages on some selected interfaces when the **ipv6 nd global-suppress-ra** command is set to disable the sending of RA messages on all other interfaces.

**Syntax: [no] ipv6 nd global-suppress-ra**

## *Configuring IPv6 ND global router advertisement globally on the default VRF*

When configuring the **ipv6 nd global-suppress-ra** command, the ND Router Advertisement messages is not sent out on any interface in the default VRF, unless the **ipv6 nd send-ra** is set on the interface. By default, **ipv6 nd global-suppress-ra** is not set for the IPv6 VRF.

Use the following command under **address-family ipv6** for a specific VRF is added and applies to the IPv6 VRF:

```
device(config)# vrf red
device(config-vrf-red)#address-family ipv6
device(config-vrf-red-ipv6)#ipv6 nd global-suppress-ra
```

**Syntax: [no] ipv6 nd global-suppress-ra**

The following command when set ensures that IPv6 ND Router Advertisement messages are sent out on the interface regardless of the setting of the **ipv6 nd global-suppress-ra** for the interface's VRF.

**Syntax: [no] ipv6 nd send-ra**

By default, **ipv6 nd send-ra** is not set on the interface. When **ipv6 nd send-ra** is set, the **ipv6 nd suppress-ra** command is unset. However, **ipv6 nd suppress-ra** is not set when **ipv6 nd send-ra** is issued on the interface. This is similar to when a user issue existing **ipv6 nd suppress-ra** command is on an interface, the **ipv6 nd send-ra** is unset. By default, **ipv6 nd suppress-ra** is not set.

If sending of RA messages is required on some selected interfaces to continue, then you must set the **ipv6 nd send-ra** command on these interfaces before setting the **ipv6 nd global-suppress-ra** command to disable the sending of RA messages on all other interfaces. Otherwise, the RA messages are not sent out until the **ipv6 nd send-ra** command is set on each of the selected interfaces.

The interface **ipv6 nd send-ra** and **ipv6 nd suppress-ra** commands are sticky in that they are independent of the **ipv6 nd global-suppress-ra** command and **either ipv6 nd send-ra** or **ipv6 nd suppress-ra** can still be present in configuration even when the **ipv6 nd global-suppress-ra** is also in configuration.

## *Show commands*

The output of **show ipv6 interface** command is modified when the sending of router advertisement is disabled on the interface or globally. Use the **show ipv6 interface** command to display the output of the interface.

```
device#show ipv6 int eth 2/1
Interface Ethernet 2/1 is up, line protocol is up
  IPv6 is enabled, link-local address is fe80::200:ff:fe03:c030 [Preferred]
  Global unicast address(es):
    31:1:1::3 [Preferred],  subnet is 31:1:1::/64
    31:1:1:: [Anycast],  subnet is 31:1:1::/64
  Joined group address(es):
    ff02::6
    ff02::5
    ff02::1:ff00:3
    ff02::1:ff03:c030
    ff02::2
    ff02::1
  Port belongs to VRF: default-vrf
  MTU is 1500 bytes
  ICMP redirects are disabled
  ND DAD is enabled, number of DAD attempts: 3
  ND reachable time is 30 seconds
  ND advertised reachable time is 0 seconds
  ND retransmit interval is 1000 milliseconds
  ND advertised retransmit interval is 0 milliseconds
  ND router advertisements suppressed
    No Inbound Access List Set
  No Outbound Access List Set
  IPv6 RPF mode: None  IPv6 RPF Log: Disabled
  OSPF enabled
  RxPkts:       0                       TxPkts:   0
  RxBytes:      0                       TxBytes:  0
  IPv6 unicast RPF drop: 0
  IPv6 unicast RPF suppressed drop: 0
device#
```

**Syntax: show ipv6 interface** [ *interface* [ *port-number* | *number* ] ]

The *interface* parameter displays detailed information for a specified interface. For the interface, the user can specify the Ethernet, loopback, tunnel, or VE keywords. If the user specifies an Ethernet interface, then the user must also specify the port number associated with the interface. If the user specifies a loopback, tunnel, or VE interface, the user must also specify the number associated with the interface.

Table 6 defines the **show ipv6 interface** command output display that shows the following information:

**TABLE 6** General IPv6 interface information fields

| This field... | Displays... |
|---|---|
| Routing protocols | A one-letter code that represents a routing protocol that can be enabled on an interface. |
| Interface | The interface type, and the port number or number of the interface. |
| Status | The status of the interface. The entry in the Status field will be either |

**TABLE 6** General IPv6 interface information fields (continued)

| This field... | Displays... |
|---|---|
| | "up/up" or "down/down". |
| Routing | The routing protocols enabled on the interface. |
| Global Unicast Address | The global unicast address of the interface. |

## Specifying a single source interface for Telnet, SSH, NTP, TFTP, TACACS/TACACS+, or RADIUS packets

When the Extreme device originates a Telnet, SSH, NTP, TFTP, TACACS/TACACS+, or RADIUS packet, the source address of the packet is the lowest-numbered IP address on the interface that sends the packet. You can configure the Extreme device to always use the lowest-numbered IP address on a specific interface as the source addresses for these types of packets. When you configure the Extreme device to use a single source interface for all Telnet, TACACS/TACACS+, or RADIUS packets, the Extreme device uses the same IP address as the source for all packets of the specified type, regardless of the ports that actually sends the packets.

Identifying a single source IP address for Telnet, SSH, NTP, TFTP, TACACS/TACACS+, or RADIUS packets provides the following benefits:

- If your Telnet, SSH, NTP, TFTP, TACACS/TACACS+, or RADIUS server is configured to accept packets only from specific IP addresses, you can use this feature to simplify configuration of the server by configuring the device to always send the packets from the same link or source address.
- If you specify a loopback interface as the single source for Telnet, SSH, NTP, TFTP, TACACS/TACACS+, or RADIUS packets, servers can receive the packets regardless of the states of individual links. Thus, if a link to the server becomes unavailable but the client or server can be reached through another link, the client or server still receives the packets, and the packets still have the source IP address of the loopback interface.

The software contains separate CLI commands for specifying the source interface for Telnet, SSH, NTP, TFTP, TACACS/TACACS+, or RADIUS packets. You can configure a source interface for one or more of these types of packets separately.

# Configuring an interface as the source for Syslog packets

You can configure the device to use the lowest-numbered IP or IPv6 address configured on a loopback interface, virtual interface, or Ethernet port as the source for all Syslog packets from the device. The software uses the lowest-numbered IP or IPv6 address configured on the interface as the source IP address for the packets.

For example, to specify the lowest-numbered IP address configured on a virtual interface as the device's source for all Syslog packets, enter commands such as the following.

```
device(config)# int ve 1
device(config-vif-1)# ip address 10.0.0.4/24
device(config-vif-1)# exit
device(config)# ip syslog source-interface ve 1
```

The commands in this example configure virtual interface 1, assign IP address 10.0.0.4/24 to the interface, then designate the interface's address as the source address for all Syslog packets.

**Syntax: [no] ip syslog source-interface ethernet** [ **slotnum/** ] **portnum** | **loopback num** | **ve num**

The *num* parameter is a loopback interface or virtual interface number. If you specify an Ethernet, the *slotnum*/]*portnum* is the port's number including the slot number, if you are configuring a device.

The default is the lowest-numbered IP or IPv6 address configured on the port through which the packet is sent. The address therefore changes, by default, depending on the port.

With this new command, the source ip of syslog is no longer controlled by the **snmp-server trap-source** command.

# Configuring forwarding parameters

The following configurable parameters control the forwarding behavior of the Extreme device:

- Time-To-Live (TTL) threshold
- Forwarding of directed broadcasts
- Forwarding of source-routed packets
- Ones-based and zero-based broadcasts

All these parameters are global and thus affect all IP interfaces configured on the Extreme device.

To configure these parameters, use the procedures in the following sections.

## Changing the TTL threshold

The TTL threshold prevents routing loops by specifying the maximum number of router hops an IP packet originated by the Extreme device can travel through. Each device capable of forwarding IP that receives the packet decreases the packet's TTL by one. If a device receives a packet with a TTL of 1 and reduces the TTL to zero, the device drops the packet.

The default TTL is 64. You can change the TTL to a value from 1- 255.

To modify the TTL threshold to 25, enter the following commands.

```
device(config)# ip ttl 25
```

**Syntax: [no] ip ttl 1-255**

## Enabling forwarding of directed broadcasts

A directed broadcast is an IP broadcast to all devices within a single directly-attached network or subnet. A net-directed broadcast goes to all devices on a given network. A subnet-directed broadcast goes to all devices within a given subnet.

> NOTE
> A less common type, the all-subnets broadcast, goes to all directly-attached subnets. Forwarding for this broadcast type also is supported, but most networks use IP multicasting instead of all-subnet broadcasting.

Forwarding for all types of IP directed broadcasts is disabled by default. You can enable forwarding for all types if needed. You cannot enable forwarding for specific broadcast types.

To enable forwarding of IP directed broadcasts, enter the following command.

```
device(config)# ip directed-broadcast
```

**Syntax: [no] ip directed-broadcast**

The software makes the forwarding decision based on the device's knowledge of the destination network prefix. Routers cannot determine that a message is unicast or directed broadcast apart from the destination network prefix. The decision to forward or not forward the message is by definition only possible in the last hop router.

To disable the directed broadcasts, enter the following command in the CONFIG mode.

```
device(config)# no ip directed-broadcast
```

To enable directed broadcasts on an individual interface instead of globally for all interfaces, enter commands such as the following.

```
device(config)# interface ethernet 1/1
device(config-if-e10000-1/1)# ip directed-broadcast
```

**Syntax: [no] ip directed-broadcast**

# Disabling forwarding of IP source-routed packets

A source-routed packet specifies the exact router path for the packet. The packet specifies the path by listing the IP addresses of the router interfaces through which the packet must pass on its way to the destination. The Extreme device supports both types of IP source routing:

- **Strict source routing** - requires the packet to pass through only the listed routers. If the Extreme device receives a strict source-routed packet but cannot reach the next hop interface specified by the packet, the Extreme device discards the packet and sends an ICMP Source-Route-Failure message to the sender.

  NOTE
  The Extreme device allows you to disable sending of the Source-Route-Failure messages. Refer to Disabling ICMP messages on page 103.

- **Loose source routing** - requires that the packet pass through all of the listed routers but also allows the packet to travel through other routers, which are not listed in the packet.

The Extreme device forwards both types of source-routed packets by default. You cannot enable or disable strict or loose source routing separately.

To disable forwarding of IP source-routed packets, enter the following command.

```
device(config)# no ip source-route
```

**Syntax: [no] ip source-route**

To re-enable forwarding of source-routed packets, enter the following command.

```
device(config)# ip source-route
```

# Enabling support for zero-based IP subnet broadcasts

By default, the Extreme device treats IP packets with all ones in the host portion of the address as IP broadcast packets. For example, the Extreme device treats IP packets with 10.157.22.255/24 as the destination IP address as IP broadcast packets and forwards the packets to all IP hosts within the 10.157.22.x subnet (except the host that sent the broadcast packet to the Extreme device).

Most IP hosts are configured to receive IP subnet broadcast packets with all ones in the host portion of the address. However, some older IP hosts instead expect IP subnet broadcast packets that have all zeros instead of all ones in the host portion of the address. To accommodate this type of host, you can enable the Extreme device to treat IP packets with all zeros in the host portion of the destination IP address as broadcast packets.

NOTE
When you enable the Extreme device for zero-based subnet broadcasts, the Extreme device still treats IP packets with all ones the host portion as IP subnet broadcasts too. Thus, the Extreme device can be configured to support all ones only (the default) or all ones and all zeroes.

NOTE
This feature applies only to IP subnet broadcasts, not to local network broadcasts. The local network broadcast address is still expected to be all ones.

To enable the Extreme device for zero-based IP subnet broadcasts in addition to ones-based IP subnet broadcasts, enter the following command.

```
device(config)# ip broadcast-zero
```

**Syntax:** [ **no** ] **ip broadcast-zero**

# Allowing multicast addresses as source IP addresses

By default packets with multicast addresses as source IP address are dropped at the packet processor in the line card. You can now disable the dropping of packets with multicast addresses as source IP address.

Unicast or multicast destination IP address forwarding works as usual, regardless of whether you enable or disable this feature. You can allow multicast addresses as source IP address for all packets or switched traffic packets only. Packets with class D and E addresses as source IPv4 address and packets with prefixes beginning with 0xFF as source IPv6 addresses (for example FF01::11), are also allowed once you enable this feature.

NOTE
Unicast Reverse Path Forwarding is disabled once you allow multicast addresses as source IP addresses.

Perform the following steps to allow multicast addresses as source IP addresses.

1. Enter global configuration mode.
2. To allow multicast addresses as source IP addresses enter the **ip allow-src-multicast** command followed by the options *decimal* or **all**.
   The following example allows multicast addresses as source IP address for all traffic.

   ```
   device(config)# ip allow-src-multicast all
   ```

3. To allow multicast addresses as source IP address for only switched traffic, enter the **ip allow-src-multicast switched-traffic** command followed by the options *decimal* or **all**.
   The following example allows multicast addresses as source IP address for switched traffic on a specific slot.

   ```
   device(config)# ip allow-src-multicast switched-traffic 3
   ```

4. To view if the disable packet drop for multicast IPv4 or IPv6 as source IP is enabled or disabled for switched-traffic only, use the **show ip allow-src-multicast switched-only** command.
5. To view if the disable packet drop for multicast IPv4 or IPv6 as source IP is enabled or disabled for all traffic use the **show ip allow-src-multicast** command.

# Configuring the maximum ICMP error message rate

NOTE
The maximum ICMP error message rate configuration only supports IPv4 traffic.

The Extreme device configuration allows 200 ICMP error messages per second per IP interface. You can now configure the maximum ICMP error message rate on all Interface Modules. The maximum configured value is increased to 5000 error messages per second. The maximum ICMP error message rate configuration uses an ICMP error metering mechanism. The process for the ICMP error metering mechanism is as follows:

- There is a meter counter for each interface. There is one total meter counter per Interface Module.
- The interface counter and the total counter will increment every time an icmp error message is sent out.
- The timer will rest all counters to 0 every second.
- Before an error message is sent out, it check the interface meter counter against the user configured icmp error limit (5000 max). The total counter will check against 10000. The error message is dropped if one any counter is larger the checked value.

The total error rate for all IP interfaces on an Interface Module is 10,000 errors per second. The ICMP error metering mechanism is per IP interface; this includes VRF IP interfaces.

Since the ICMP error metering code implementation is similar between the Management Module and Interface Module code, this change will also affect the Management Module ICMP error rate.

To configure the maximum ICMP error rate, enter the following command.

```
device(config)# ip icmp max-err-msg-rate 600
```

**Syntax: [no] ip icmp max-err-msg-rate error per second**

The *error per second* variable specifies the maximum error rate in errors per second. The maximum configured value has a range from 0 (minimum) to 5000 (maximum) error message per second. The default value is 400.

# Disabling ICMP messages

The Extreme device is enabled to reply to ICMP echo messages and send ICMP Destination Unreachable messages by default.

You can selectively disable the following types of Internet Control Message Protocol (ICMP) messages:

- **Echo messages (ping messages)** - The Extreme device replies to IP pings from other IP devices.
- **Destination Unreachable messages** - If the Extreme device receives an IP packet that it cannot deliver to its destination, the Extreme device discards the packet and sends a message back to the device that sent the packet. The message informs the device that the destination cannot be reached by the Extreme device.

## *Disabling replies to broadcast ping requests*

By default, the Extreme device is enabled to respond to broadcast ICMP echo packets, which are ping requests.

To disable response to broadcast ICMP echo packets (ping requests), enter the following command.

```
device(config)# no ip icmp echo broadcast-request
```

**Syntax: [no] ip icmp echo broadcast-request**

If you need to re-enable response to ping requests, enter the following command.

```
device(config)# ip icmp echo broadcast-request
```

## *Disabling ICMP destination unreachable messages*

By default, when this Extreme device receives an IP packet that the device cannot deliver, the device sends an ICMP Unreachable message back to the host that sent the packet. You can selectively disable a device's response to the following types of ICMP Unreachable messages:

- **Administration** - The packet was dropped by the device due to a filter or ACL configured on the device.
- **Fragmentation-needed** - The packet has the Do not Fragment bit set in the IP Flag field, but the device cannot forward the packet without fragmenting it.
- **Host** - The destination network or subnet of the packet is directly connected to the device, but the host specified in the destination IP address of the packet is not on the network.
- **Network** - The device cannot reach the network specified in the destination IP address of the packet.
- **Port** - The destination host does not have the destination TCP or UDP port specified in the packet. In this case, the host sends the ICMP Port Unreachable message to the device, which in turn sends the message to the host that sent the packet.
- **Protocol** - The TCP or UDP protocol on the destination host is not running. This message is different from the Port Unreachable message, which indicates that the protocol is running on the host but the requested protocol port is unavailable.
- **Source-route-failure** - The device received a source-routed packet but cannot locate the next-hop IP address indicated in the packet's Source-Route option.

You can disable the device from sending these types of ICMP messages on an individual basis.

NOTE
Disabling an ICMP Unreachable message type does not change the device's ability to forward packets. Disabling ICMP Unreachable messages prevents the device from generating or forwarding the Unreachable messages.

To disable all ICMP Unreachable messages, enter the following command.

```
device(config)# no ip icmp unreachable
```

**Syntax: [no] ip icmp unreachable** [ **network** | **host** | **protocol** | **administration** | **fragmentation-needed** | **port** | **source-route-fail** ]

- If you enter the command without specifying a message type (as in the example above), all types of ICMP Unreachable messages listed above are disabled. If you want to disable only specific types of ICMP Unreachable messages, you can specify the message type. To disable more than one type of ICMP message, enter the **no ip icmp unreachable** command for each messages type.
- The **network** parameter disables ICMP Network Unreachable messages.
- The **host** parameter disables ICMP Host Unreachable messages.
- The **protocol** parameter disables ICMP Protocol Unreachable messages.
- The **administration** parameter disables ICMP Unreachable (caused by Administration action) messages.
- The **fragmentation-needed** parameter disables ICMP Fragmentation-Needed But Do not-Fragment Bit Set messages.
- The **port** parameter disables ICMP Port Unreachable messages.
- The **source-route-fail** parameter disables ICMP Unreachable (caused by Source-Route-Failure) messages.

To disable ICMP Host Unreachable messages and ICMP Network Unreachable messages but leave the other types of ICMP Unreachable messages enabled, enter the following commands instead of the command shown above.

```
device(config)# no ip icmp unreachable host
device(config)# no ip icmp unreachable network
```

If you have disabled all ICMP Unreachable message types but want to re-enable certain types, you can do so by entering commands such as the following.

```
device(config)# ip icmp unreachable host
device(config)# ip icmp unreachable network
```

These commands re-enable ICMP Unreachable Host messages and ICMP Network Unreachable messages.

## Disabling ICMP redirect messages

ICMP redirect messages can be disabled or re-enabled. By default, the Extreme device sends an ICMP redirect message to the source of a misdirected packet in addition to forwarding the packet to the appropriate router. You can disable ICMP redirect messages on a global basis or on an individual port basis.

> NOTE
> An unusually high receipt of multiple Internet Control Message Protocol (ICMP) Redirect packets that are used to change routing table entries in a short period of time may cause high CPU utilization. This can be avoided by configuring the maximum ICMP error message rate using **ip icmp max-err-msg-rate** command, 0 (minimum) to 5000 (maximum) error message per second. The default value is 400.The total error rate for all IP interfaces (SYSTEM) is 10,000 errors per second.

> NOTE
> The device forwards misdirected traffic to the appropriate router, even if you disable the redirect messages.

To disable ICMP redirect messages globally, enter the following command at the global CONFIG level of the CLI.

> NOTE
> The **ip icmp redirects** command is applicable to the MLX Series and XMR Series devices only.

```
device(config)# no ip icmp redirects
```

**Syntax: [no] ip icmp redirects**

To disable ICMP redirect messages on a specific interface, enter the following command at the configuration level for the interface.

```
device(config)# int e 3/11
device(config-if-e100-3/11)# no ip redirect
```

**Syntax: [no] ip redirect**

# Configuring IP load sharing

The IP route table can contain more than one path to a given destination. When this occurs, the Extreme device selects the path with the lowest cost as the path for forwarding traffic to the destination. If the IP route table contains more than one path to a destination and the paths each have the lowest cost, then the Extreme device uses IP load sharing to select a path to the destination.

IP load sharing is based on the destination address of the traffic. Extreme devices support load sharing based on individual host addresses or on network addresses.

You can enable a device to load balance across up to eight equal-cost paths. The default maximum number of equal-cost load sharing paths is four.

> NOTE
> IP load sharing is not based on source routing, only on next-hop routing.

NOTE
The term "path" refers to the next-hop router to a destination, not to the entire route to a destination. Thus, when the software compares multiple equal-cost paths, the software is comparing paths that use different next-hop routers, with equal costs, to the same destination.In many contexts, the terms "route" and "path" mean the same thing. Most of the user documentation uses the term "route" throughout. The term "path" is used in this section to refer to an individual next-hop router to a destination, while the term "route" refers collectively to the multiple paths to the destination. Load sharing applies when the IP route table contains multiple, equal-cost paths to a destination.

NOTE
The Extreme device also performs load sharing among the ports in aggregate links.

# How multiple equal-cost paths enter the IP route table

IP load sharing applies to equal-cost paths in the IP route table. Routes eligible for load sharing can enter the table from the following sources:

- IP static routes
- Routes learned through RIP, OSPF, and BGP4

## *Administrative distance*

The administrative distance is a unique value associated with each type (source) of IP route. Each path has an administrative distance. It is used when evaluating multiple equal-cost paths to the same destination from different sources, such as RIP, OSPF and so on, but not used when performing IP load sharing.

The value of the administrative distance is determined by the source of the route. The Extreme device is configured with a unique administrative distance value for each IP route source.

When the software receives paths from different sources to the same destination, the software compares their administrative distances, selects the one with the lowest distance, and puts it in the IP route table. For example, if the Extreme device has a path learned from OSPF and a path learned from RIP for a given destination, only the path with the lower administrative distance enters the IP route table.

Here are the default administrative distances on the Extreme device:

- Directly connected - 0 (this value is not configurable)
- Static IP route - 1 (applies to all static routes, including default routes and default network routes)
- Exterior Border Gateway Protocol (EBGP) - 20
- OSPF - 110
- RIP - 120
- Interior Gateway Protocol (IBGP) - 200
- Local BGP - 200
- Unknown - 255 (the device will not use this route)

Lower administrative distances are preferred over higher distances. For example, if the device receives routes for the same network from OSPF and from RIP, the device will prefer the OSPF route by default.

NOTE
You can change the administrative distances individually. Refer to the configuration chapter for the route source for information.

Since the software selects only the path with the lowest administrative distance, and the administrative distance is determined by the path's source, IP load sharing does not apply to paths from different route sources. IP load sharing applies only when the IP route table contains paths from the same IP route source to the same destination.

## Path cost

The cost parameter provides a basis of comparison for selecting among paths to a given destination. Each path in the IP route table has a cost. When the IP route table contains multiple paths to a destination, the Extreme device chooses the path with the lowest cost. When the IP route table contains more than one path with the lowest cost to a destination, the Extreme device uses IP load sharing to select one of the lowest-cost paths.

The source of a path's cost value depends on the source of the path:

- **IP static route** - The value you assign to the metric parameter when you configure the route. The default metric is 1.
- **RIP** - The number of next-hop routers to the destination.
- **OSPF** - The Path Cost associated with the path. The paths can come from any combination of inter-area, intra-area, and external Link State Advertisements (LSAs).
- **BGP4** - The path's Multi-Exit Discriminator (MED) value.

  NOTE
  If the path is redistributed between two or more of the above sources before entering the IP route table, the cost can increase during the redistribution due to settings in redistribution filters.

## Static route, OSPF, and BGP4 load sharing

IP load sharing and load sharing for static routes, OSPF routes, and BGP4 routes are individually configured. Multiple equal-cost paths for a destination can enter the IP route table only if the source of the paths is configured to support multiple equal-cost paths. For example, if BGP4 allows only one path with a given cost for a given destination, the BGP4 route table cannot contain equal-cost paths to the destination. Consequently, the IP route table will not receive multiple equal-cost paths from BGP4.

Table 7 lists the default and configurable maximum numbers of paths for each IP route source that can provide equal-cost paths to the IP route table. The table also lists where to find configuration information for the route source's load sharing parameters.

The load sharing state for all the route sources is based on the state of IP load sharing. Since IP load sharing is enabled by default on the Extreme device, load sharing for static IP routes, RIP routes, OSPF routes, and BGP4 routes also is enabled by default.

**TABLE 7** Default load sharing parameters for route sources

| Route source | Default maximum number of paths | Maximum number of paths |
|---|---|---|
| Static IP route | 4<br>NOTE<br>This value depends on the value for IP load sharing, and is not separately configurable. | 32<br>NOTE<br>This value depends on the value for IP load sharing, and is not separately configurable. |
| RIP | 4<br>NOTE<br>This value depends on the value for IP load sharing, and is not separately configurable. | 8<br>NOTE<br>This value depends on the value for IP load sharing, and is not separately configurable. |
| OSPF | 4 | 32 |
| BGP4 | 1 | 32 |

NOTE
Suppose you have a route that points to an ECMP next hop and the route paths consist of more than one type, then only the first path is programmed in the hardware for forwarding. The number of paths for ECMP is set to 1.

# Options for IP load sharing and LAGs

The following options have been added to refine the hash calculations used for IP load sharing and LAGs. These include the following:

- **Speculate UDP or TCP Headers** - This option is applied to ECMP and LAG index hash calculations.
- **Mask Layer-3 and Layer-4 Information** - This option is applied to ECMP and LAG index hash calculations.
- **Mask Layer-2 Information** - This option is applied to ECMP and LAG index hash calculations.
- **Mask MPLS label information** - This option is applied to ECMP and LAG index hash calculations.
- **Diversification** - This option is applied to ECMP and LAG index hash calculations.
- **Hash Rotate** - This option is applied to ECMP hash calculations and to LAG index calculations.
- **Symmetric** - This option is applied to trunk hash calculations.

NOTE
The CES 2000 Series devices do not support the same options as the XMR Series and MLX Series devices. Refer to the CES 2000 Series and CER 2000 Series Link Aggregation chapter for additional information on hash calculations used for IP load sharing and LAGs on the CES 2000 Series devices.

## *Speculate UDP or TCP packet headers*

With this option set, the packet headers following IPv4 headers are used for the ECMP and LAG index hash calculations even if the packet is not a TCP or UDP packet. If the packet is a non-fragmented, no-IP options, TCP or UDP packet, the TCP or UDP ports are used for hash calculations unless the **load-balance mask ip** or **load-balance mask ipv6** commands are used. This behavior is disabled by default and can be enabled using the following command.

```
device(config)# load-balance force-l4-hashing all
```

**Syntax: [no] load-balance force-l4-hashing** [ **all** | **slot-number** | **slot-number np-id** ]

The **all** option applies the command to all ports within the device.

Specifying a slot number using the *slot-number* variable limits the command to an individual module.

Specifying a slot number and a network processor ID using the *slot-number* and *np-id* variables limits the command to the ports supported by the specified network processor on the specified interface module.

NOTE
Problems can occur with the **Ping** and **Traceroute** functions when this option is enabled.

## *Masking Layer 3 and Layer 4 information*

Masking in networking means that a specific header field is used for hashing. With the Layer 3 and Layer 4 masking option set, the following values can be masked during ECMP and LAG index hash calculations: TCP or UDP source and destination port information, source and destination IP address, IPv4 protocol ID, and IPv6 next header.

When used with the **load-balance force-l4-hashing** command, the **load-balance mask ip** command takes precedence. The masking option can be set using the following commands for IPv4 addresses.

```
device(config)# load-balance mask ip src-l4-port all
```

**Syntax: [no] load-balance mask ip** [ **dst-ip** [ *slot number* | **all** | **pre-symmetriclb** ] | **src-ip** [ *slot number* | **all** | **pre-symmetriclb** ] | **dst-l4-port** |[ *slot number* | **all** ] | **src-l4-port** [ *slot number* | **all** ] | **protocol** [ *slot number* | **all** ] ]

Use the **src-l4-port** option when you want to mask the Layer 4 source port.

Use the **dst-l4-port** option when you want to mask the Layer 4 destination port.

Use the **src-ip** option when you want to mask the source IP address. The **src-ip** keyword contains the **pre-symmetriclb** option that masks the source IP address before symmetric load balancing can occur.

Use the **dst-ip** option when you want to mask the destination IP address. The **dst-ip** keyword contains the **pre-symmetriclb** option that masks the destination IP address before symmetric load balancing can occur.

Use the **protocol** option when you want to mask the IPv4 protocol ID.

The **all** option applies the command to all ports within the device.

Specifying a slot number using the *slot-number* variable limits the command to an individual module.

Specifying a slot number and a network processor ID using the *slot-number* and *np-id* variables limits the command to the ports supported by the specified network processor on the specified interface module.

The masking option can be set using the following commands for IPv6 addresses.

```
device(config)# load-balance mask ipv6 src-l4-port all
```

**Syntax: [no] load-balance mask ipv6** [ **dst-ip** [ *slot number* | **all** | **pre-symmetriclb** ] | **src-ip** [ *slot number* | **all** | **pre-symmetriclb** ] | **dst-l4-port** |[ *slot number* | **all** ] | **src-l4-port** [ *slot number* | **all** ] | **next-hdr** [ *slot number* | **all** ] ]

Except for the **next-hdr** option, the command options described for the **load-balance mask ip** command are valid for the **load-balance mask ipv6**.

Use the **next-hdr** option when you want to mask the IPv6 next header.

Use the **src-ip** option when you want to mask the source IPv6 address. The **src-ip** keyword contains the **pre-symmetriclb** option that masks the source IPv6 address before symmetric load balancing can occur. The symmetric load balancing can be either static or dynamic LAG load balancing.

Use the **dst-ip** option when you want to mask the destination IPv6 address. The **dst-ip** keyword contains the **pre-symmetriclb** option that masks the destination IPv6 address before symmetric load balancing can occur.

The **[no] load-balance mask ip** and **[no] load-balance mask ipv6** commands are disabled by default.

NOTE
The *Masking Layer 3 and Layer 4 information* feature supports both static and dynamic LAG load balancing.

## *Masking Layer 2 information*

With the **load-balance mask ethernet** command set, the following Layer 2 values can be masked during ECMP and LAG index hash calculations: source and destination MAC address, VLAN, Ethertype, and Inner VLAN. To mask Layer 2 information, use the **load-balance mask ethernet** command, as shown in the following.

```
device(config)# load-balance mask ethernet sa-mac all
```

**Syntax: [no] load-balance mask ethernet** [ **sa-mac** | **da-mac** | **vlan** | **etype** | **inner-vlan** ] [ **all** | **slot-number** | **slot-number np-id** ]

Use the **sa-mac** option when you want to mask the source MAC address.

Use the **da-mac** option when you want to mask the destination MAC address.

Use the **vlan** option when you want to mask the VLAN ID.

Use the **etype** option when you want to mask the Ethertype

Use the **inner-vlan** option when you want to mask the inner VLAN ID.

The **all** option applies the command to all ports within the device.

Specifying a slot number using the *slot-number* variable limits the command to an individual module.

Specifying a slot number and a network processor ID using the *slot-number* and *np-id* variables limits the command to the ports supported by the specified network processor on the specified interface module.

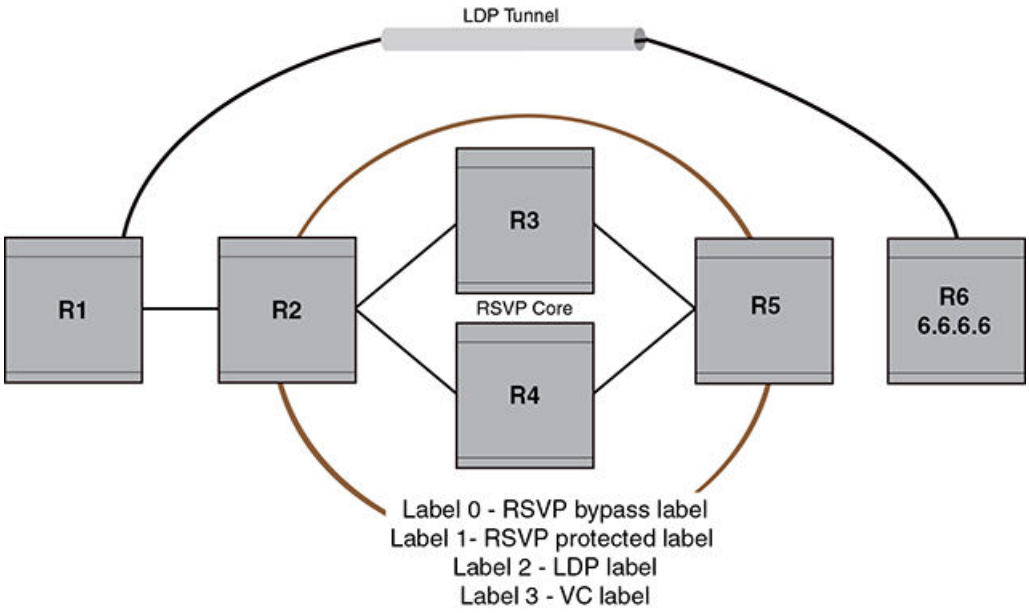
## *Configuring mask option for load balancing*

In an MPLS network, when the L2VPN is configured using a LDP tunnel, which in turn is using a RSVP bypass tunnel, then the packets will include four labels. The four labels are:

- RSVP bypass label - Label 0 which is the outermost MPLS label
- RSVP protected label - Label 1
- LDP label - Label 2
- VC label - Label 3 which is the innermost MPLS label

In the Figure 13, all the packets routed between the routers, R2 and R5 include four MPLS labels which are masked for calculating the ECMP and LAG index hash value.

**FIGURE 13** L2VPN packets over a LDP tunnel




To mask the MPLS labels, enter the following command.

```
device(config)# load-balance mask mpls label0 all
```

**Syntax: [no] load-balance mask mpls** [ **label0** | **label1** | **label2** | **label3** ] [ **all** | *slot-number* | *slot-number np-id* ]

Use the **label0** option to mask MPLS Label 0, which is the innermost MPLS label in a packet.

Use the **label1** option to mask MPLS Label 1, which is the next innermost MPLS label in a packet from MPLS Label 2.

Use the **label2** option to mask MPLS Label 2, which is the next innermost MPLS label in a packet with four labels or the outermost MPLS label in a packet with three labels.

Use the**label3** option to mask MPLS Label 3, which is the outermost MPLS label in a packet with four labels.

The **all** option applies the command to all ports within the router.

Specifying a slot number using the *slot-number* variable limits the command to an individual module.

Specifying a slot number and a network processor ID using the *slot-number* and *np-id* variables limits the command to the ports supported by the specified network processor on the specified interface module.

## *Displaying MPLS masking information*

To display the masking information, enter the following command.

```
device# show load-balance mask mpls
Mask MPLS options -
  Mask MPLS Label0 is enabled on -
  No Slots
  Mask MPLS Label1 is enabled on -
  No Slots
  Mask MPLS Label2 is enabled on -
  No Slots
  Mask MPLS Label3 is enabled on -
  All Slots
```

Table 8 describes the output parameters of the **show load-balance mask mpls** command.

**TABLE 8** Output parameters of the show load-balance mask mplscommand

| Field | Description |
|---|---|
| Slot | Shows the slot of the interface on which the MPLS masking is enabled. |
| Mask MPLS Label | Shows whether or not the following labels are masked.<br>• Label0 - Shows if the Label 0 is masked on the interface.<br>• Label1 - Shows if the Label 1 is masked on the interface.<br>• Label2 - Shows if the Label 2 is masked on the interface.<br>• Label3 - Shows if the Label 3 is masked on the interface. |

To display current running configuration, enter the following command.

```
device# show running-config
!
load-balance mask mpls label3 all
```

!

## *Hash diversification for LAGs and IP load balancing*

In a multi-stage network a traffic flow will normally use the same LAG port or same path (for IP load balancing) at each stage. The Hash Diversification feature works within an earlier stage of the hash calculation than the hash rotate feature. Using the**load-balance hash-diversify** command, you can provide a unique hash diversify value to a device, or a sub-set of ports on a device. This unique value is used in calculation of the ECMP and LAG index hash. Consequently, instead of a traffic flow always following the same port group or path, it will be distributed over different LAG or ECMP members. To apply hash diversification, use the following command.

```
device(config)# load-balance hash-diversify random all
```

**Syntax: [no] load-balance hash-diversify** [ **number** | **random** | **slot** ] [ **all** | **slot-number** | **slot-number np-id** ]

You can set the unique hash diversify value using one of the following options:

The *number* option allows you to specify a value from 0 - 255.

The **random** option directs the CPU to generate a random number for each packet processor and program it as the hash diversification value.

The **slot** option specifies the slot ID as the has diversification number.

The default value for the diversification number is 0 and the **no** version of the command resets the value to 0 regardless of any value previously set. Also, the most recent command added overrides any previous instances of the command. For example, if the **random** option is entered first and is then followed by the **slot** option, the value of the slot ID for the specified slot will be used.

The **all** option applies the command to all ports within the device.

Specifying a slot number using the *slot-number* variable limits the command to an individual module.

Specifying a slot number and a network processor ID using the *slot-number* and *np-id* variables limits the command to the ports supported by the specified network processor on the specified interface module.

This option can also be used in a multi-stage network to avoid the same traffic flow to always use one path of an ECMP or the same LAG member index at each stage. Using the hash rotate function the same set of traffic flows forwarded out of one LAG member or ECMP path to the next router can be distributed across different paths of the LAG member or ECMP path to the next router.

### *Hash rotate for LAGs and IP load balancing*

The hash rotate function provides another option (in addition to hash diversification) for diversifying traffic flow in a multi-stage network. Using this feature, the ECMP hash index can be rotated by a specified number of bits after it has been calculated. This allows path selection within IP load balancing to be more diverse.

To configure hash rotate to LAG index calculations, enter a command such as the following.

```
device(config)# load-balance hash-rotate 3 all
```

**Syntax: [no] load-balance hash-rotate rotate-number** [ **all** | **slot-number** | **slot-number np-id** ]

The *rotate-number* value specifies number of bits between 0 and 7 that you want to rotate the ECMP hash index value.

The **all** option applies the command to all ports within the device.

Specifying a slot number using the *slot-number* variable limits the command to an individual module.

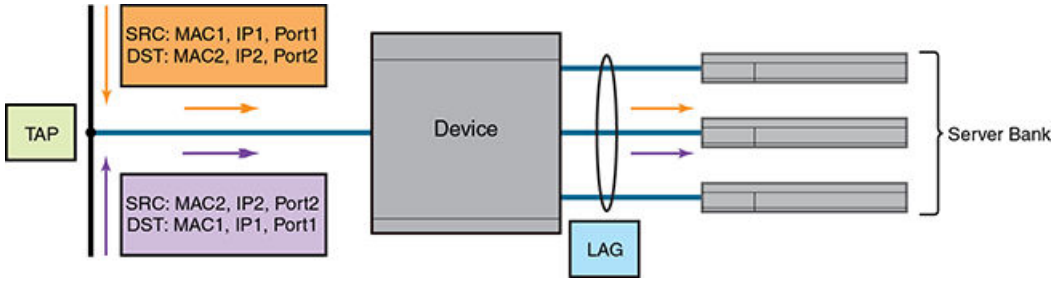
Specifying a slot number and a network processor ID using the *slot-number* and *np-id* variables limits the command to the ports supported by the specified network processor on the specified interface module.

NOTE
The hash diversification and hash rotate features can be applied separately or together. Depending on your network configuration, either or both of these features may need to be configured.

## Symmetric load balancing for LAGs

For many monitoring and security applications, bidirectional conversations flowing through the system must be carried on the same port of a LAG. For Network Telemetry applications, network traffic is tapped and sent to the Extreme devices, which can load balance selected traffic to the application servers downstream. Each server analyzes the bidirectional conversations. Therefore, the Extreme devices must enable symmetric load balancing to accomplish bidirectional conversations. In addition, firewalls between the Extreme devices can be configured to allow the bidirectional conversations per link of the LAG. These applications also require symmetric load balancing on the LAGs between the Extreme devices. Figure 14 depicts the symmetric load balancing for LAGs feature.

**FIGURE 14** Symmetric load balancing for LAGs




NOTE

The symmetric load balancing option is applicable only for MLX Series and XMR Series devices. The CER 2000 Series and CES 2000 Series devices load balance all traffic on the LAGs symmetrically. Therefore, the CER 2000 Series and CES 2000 Series devices do not support the symmetric load balancing commands.

With the symmetric load balancing option set, the trunk hash calculation is determined using all or a combination of the following parameters: MAC source and destination addresses, IPv4 source and destination addresses, IPv6 source and destination addresses, TCP or UDP source and destination port information, inner MAC source and destination addresses, inner IPv4 source and destination addresses, and inner IPv6 source and destination addresses.

To enable the symmetric load balancing option on an interface, enter commands such as the following.

```
device(config)# load-balance symmetric ethernet 2
device(config)# load-balance symmetric ip all
device(config)# load-balance symmetric ipv6 2
device(config)# load-balance symmetric l4_ip 2
device(config)# load-balance symmetric l4_ipv6 2
device(config)# load-balance symmetric inner_ethernet 2
device(config)# load-balance symmetric inner_ip 2
device(config)# load-balance symmetric inner_ipv6 2
```

**Syntax: [no] load-balance symmetric ethernet** | **ip** | **ipv6** | **l4_ip** | **l4_ipv6** | **inner_ethernet** | **inner_ip** | **inner_ipv6** | **packet** [ **all** | *slot-number* | *slot-number np-id* ]

The **ethernet** option specifies the Ethernet header fields.

The **ip** option specifies the IP header fields.

The **ipv6** option specifies the IPv6 header fields.

The **l4_ip** option specifies the Layer 4 IP fields

The **l4_ipv6** option specifies the Layer 4 IPv6 fields.

The**inner_ethernet** option specifies the inner Ethernet fields.

The **inner_ip** option specifies the inner IP fields.

The **inner_ipv6** option specifies the inner IPv6 fields.

The **packet** option specifies all the packet fields.

The **all** option applies the command to all ports within the router.

Specifying a slot number using the *slot-number* variable limits the command to an individual module.

Specifying a slot number and a network processor ID using the *slot-number* and *np-id* variables limits the command to the ports supported by the specified network processor on the specified interface module.

The **no** option is used to turn off the previously enabled symmetric load balancing option.

## *Displaying symmetric load balancing information*

To display the symmetric load balancing information for the interface, enter the following command.

```
device# show load-balance symmetric-options
Symmetric Ethernet options -
  Symmetric Ethernet is enabled on -
  Slot 2
  Slot 3
Symmetric IP options -
  Symmetric IP is enabled on -
  All Slots
Symmetric IPv6 options -
  Symmetric IPV6 is enabled on -
  Slot 1
  Slot 2
Symmetric IP Layer 4 IP options -
  Symmetric Layer 4 IP is enabled on -
  Slot 2
Symmetric IPv6 Layer 4 IPV6 options -
  Symmetric Layer 4 IPV6 is enabled on -
  Slot 2
Symmetric INNER Ethernet options -
  Symmetric INNER Ethernet is enabled on -
  Slot 2
Symmetric INNER IP options -
  Symmetric INNER IP is enabled on -
  Slot 2
Symmetric INNER IPV6 options -
  Symmetric INNER IPV6 is enabled on -
  Slot 2
```

**Syntax: show load-balance symmetric-options ethernet | ip | ipv6 | l4_ip | l4_ipv6 | inner_ethernet | inner_ip | inner_ipv6 | packet**

Table 9 describes the output parameters of the **show load-balance symmetric-options** command.

**TABLE 9** Output parameters of the show load-balance symmetric-options command

| Field | Description |
|---|---|
| Slot | Shows the slot number of the interface on which the symmetric load balancing option is enabled. |
| Symmetric options | Shows whether or not the symmetric option is enabled on the following interfaces:<br>• Symmetric Ethernet options - Shows if the symmetric option is enabled on the Ethernet interface.<br>• Symmetric IP options - Shows if the symmetric option is enabled on the IP interface.<br>• Symmetric IPv6 options - Shows if the symmetric option is enabled on the IPv6 interface.<br>• Symmetric Layer 4 IP options - Shows if the symmetric option is enabled on the Layer 4 IP interface.<br>• Symmetric Layer 4 IPv6 options - Shows if the symmetric option is enabled on the Layer 4 IPv6 interface.<br>• Symmetric INNER Ethernet options - Shows if the symmetric option is enabled on the inner Ethernet interface.<br>• Symmetric INNER IP options - Shows if the symmetric option is enabled on the inner IP interface.<br>• Symmetric INNER IPv6 options - Shows if the symmetric option is enabled on the inner IPv6 interface.<br>• Symmetric packet options - Shows if the symmetric option is enabled on all the interfaces. |

# How IP load sharing works

On the Extreme device, IP load sharing is done by the hardware. If there is more than one path to a given destination, a hash is calculated based on the source MAC address, destination MAC address, source IP address, destination IP address, VLAN-ID (if applicable), IPv4 protocol number, IPv6 next header and TCP/UDP source port and destination port if the packet is also a TCP/UDP packet. This hash is used to select one of the paths.

## *Changing the maximum number of load sharing paths*

By default, IP load sharing allows IP traffic to be balanced across up to four equal path. You can change the maximum number of paths that the Extreme device supports to a value between 2 and 32.

> NOTE
> The maximum number of paths supported by the BR-MLX-10Gx24-DM module is 16.

For optimal results, set the maximum number of paths to a value equal to or greater than the maximum number of equal-cost paths that your network typically contains. For example, if the Extreme device has six next-hop routers, set the maximum paths value to six.

> NOTE
> If the setting for the maximum number of paths is lower than the actual number of equal-cost paths, the software does not use all the paths for load sharing.

To change the maximum number of load sharing paths, enter the following command:

```
device(config)# ip load-sharing 32
```

**Syntax: [no] ip load-sharing number**

The *number* parameter specifies the number of ECMP load sharing paths. Enter a value between 2 and 32 for *number* to set the maximum number of paths. The default value is 4.

> NOTE
> A new **maximum-paths use-load-sharing** command was introduced under the BGP configuration that allows support for BGP routes in IP load sharing but does not enable BGP multipath load sharing.

## *Response to path state changes*

If one of the load-balanced paths becomes unavailable, the IP route table in hardware is modified to stop using the unavailable path. The traffic is load balanced between the available paths using the same hashing mechanism described above. (Refer to How IP load sharing works on page 115.)

# Configuring IRDP

The Extreme device uses ICMP Router Discovery Protocol (IRDP) to advertise the IP addresses of its device interfaces to directly attached hosts. IRDP is disabled by default. You can enable it globally or on individual ports.

Consider the following when you enable or disable IRDP globally:

- If you enable IRDP globally, all ports use the default values for the IRDP parameters.
- If you leave IRDP disabled globally but enable it on individual ports, you also can configure the IRDP parameters on an individual port basis.

NOTE
You can configure IRDP parameters only an individual port basis. To do so, IRDP must be disabled globally and enabled only on individual ports. You cannot configure IRDP parameters if the feature is globally enabled.

When IRDP is enabled, the Extreme device periodically sends Router Advertisement messages out the IP interfaces on which the feature is enabled. The messages advertise the Extreme device's IP addresses to directly attached hosts who listen for the messages. In addition, hosts can be configured to query the Extreme device for the information by sending Router Solicitation messages.

Some types of hosts use the Router Solicitation messages to discover their default gateway. When IRDP is enabled, the Extreme device responds to the Router Solicitation messages. Some clients interpret this response to mean that the Extreme device is the default gateway. If another router is actually the default gateway for these clients, leave IRDP disabled on the Extreme device.

IRDP uses the following parameters. If you enable IRDP on individual ports rather than globally, you can configure these parameters on an individual port basis. The IRDP parameters are as follows:

- **Packet type** - The Extreme device can send Router Advertisement messages as IP broadcasts or as IP multicasts addressed to IP multicast group 224.0.0.1. The packet type is IP broadcast.
- **Maximum message interval and minimum message interval** - When IRDP is enabled, the Extreme device sends the Router Advertisement messages every 450 - 600 seconds by default. The time within this interval that the Extreme device selects is random for each message and is not affected by traffic loads or other network factors. The random interval minimizes the probability that a host will receive Router Advertisement messages from other routers at the same time. The interval on each IRDP-enabled Extreme device interface is independent of the interval on other IRDP-enabled interfaces. The default maximum message interval is 600 seconds. The default minimum message interval is 450 seconds.
- **Hold time** - Each Router Advertisement message contains a hold time value. This value specifies the maximum amount of time the host should consider an advertisement to be valid until a newer advertisement arrives. When a new advertisement arrives, the hold time is reset. The hold time is always longer than the maximum advertisement interval. Therefore, if the hold time for an advertisement expires, the host can reasonably conclude that the router interface that sent the advertisement is no longer available. The default hold time is three times the maximum message interval.
- **Preference** - If a host receives multiple Router Advertisement messages from different routers, the host selects the router that sent the message with the highest preference as the default gateway. The preference can be a number from 4294967296 to 4294967295. The default is 0.

## *Enabling IRDP globally*

To globally enable IRDP, enter the following command.

```
device(config)# ip irdp
```

This command enables IRDP on the IP interfaces on all ports. Each port uses the default values for the IRDP parameters. The parameters are not configurable when IRDP is globally enabled.

## *Enabling IRDP on an individual port*

To enable IRDP on an individual interface and change IRDP parameters, enter commands such as the following.

```
device(config)# interface ethernet 1/3
device(config-if-e10000-1/3)# ip irdp maxadvertinterval 400
```

This example shows how to enable IRDP on a specific port and change the maximum advertisement interval for Router Advertisement messages to 400 seconds.

NOTE
To enable IRDP on individual ports, you must leave the feature globally disabled.

**Syntax: [no] ip irdp** [ **broadcast** | **multicast** ] [ **holdtime seconds** ] [ **maxadvertinterval seconds** ] [ **minadvertinterval seconds** ] [ **preference number** ]

The **broadcast and multicast** parameter specifies the packet type the Extreme device uses to send Router Advertisement.

- **broadcast** - The Extreme device sends Router Advertisement as IP broadcasts. This is the default.
- **multicast** - The Extreme device sends Router Advertisement as multicast packets addressed to IP multicast group 224.0.0.1.

The **holdtime***seconds* parameter specifies how long a host that receives a Router Advertisement from the Extreme device should consider the advertisement to be valid. When a host receives a new Router Advertisement message from the Extreme device, the host resets the hold time for the Extreme device to the hold time specified in the new advertisement. If the hold time of an advertisement expires, the host discards the advertisement, concluding that the router interface that sent the advertisement is no longer available. The value must be greater than the value of the **maxadvertinterval** parameter and cannot be greater than 9000. The default is three times the value of the **maxadvertinterval** parameter.

The **maxadvertinterval** parameter specifies the maximum amount of time the Extreme device waits between sending Router Advertisements. You can specify a value from 1 to the current value of the **holdtime** parameter. The default is 600 seconds.

The **minadvertinterval** parameter specifies the minimum amount of time the Extreme device can wait between sending Router Advertisements. The default is three-fourths (0.75) the value of the **maxadvertinterval** parameter. If you change the **maxadvertinterval** parameter, the software automatically adjusts the **minadvertinterval** parameter to be three-fourths the new value of the **maxadvertinterval** parameter. If you want to override the automatically configured value, you can specify an interval from 1 to the current value of the **maxadvertinterval** parameter.

The **preference***number* parameter specifies the IRDP preference level of the Extreme device. If a host receives Router Advertisements from multiple routers, the host selects the router interface that sent the message with the highest interval as the host's default gateway. The valid range is 4294967296 to 4294967295. The default is 0.

# Configuring UDP broadcast and IP helper parameters

Some applications rely on client requests sent as limited IP broadcasts addressed to the UDP's application port. If a server for the application receives such a broadcast, the server can reply to the client. Routers do not forward subnet directed broadcasts, so the client and server must be on the same network for the broadcast to reach the server. If the client and server are on different networks (on opposite sides of a router), the client's request cannot reach the server.

To configure the Extreme device to forward client requests to UDP application servers:

- Enable forwarding support for the UDP application port, if forwarding support is not already enabled.
- Configure a helper adders on the interface connected to the clients. Specify the helper address to be the IP address of the application server or the subnet directed broadcast address for the IP subnet the server is in. A helper address is associated with a specific interface and applies only to client requests received on that interface. The Extreme device forwards client requests for any of the application ports the Extreme device is enabled to forward to the helper address.

Forwarding support for the following application ports is enabled by default:

- bootps (port 67)
- dns (port 53)
- tftp (port 69)
- time (port 37)
- netbios-ns (port 137)
- netbios-dgm (port 138)
- tacacs (port 65)

NOTE
The application names are the names for these applications that the Extreme device recognizes, and might not match the names for these applications on some third-party devices. The numbers listed in parentheses are the UDP port numbers for the applications. The numbers come from RFC 1340.

NOTE
As shown above, forwarding support for BootP or DHCP is enabled by default. If you are configuring the Extreme device to forward BootP or DHCP requests, refer to Configuring BootP or DHCP forwarding parameters on page 119.

You can enable forwarding for other applications by specifying the application port number.

You also can disable forwarding for an application.

NOTE
If you disable forwarding for a UDP application, forwarding of client requests received as broadcasts to helper addresses is disabled. Disabling forwarding of an application does not disable other support for the application. For example, if you disable forwarding of Telnet requests to helper addresses, other Telnet support on the Extreme device is not also disabled.

## *Enabling forwarding for a UDP application*

If you want the Extreme device to forward client requests for UDP applications that the Extreme device does not forward by default, you can enable forwarding support for the port. To enable forwarding support for a UDP application, use either of the following methods. You also can disable forwarding for an application using these methods.

NOTE
You also must configure a helper address on the interface that is connected to the clients for the application. The Extreme device cannot forward the requests unless you configure the helper address. Refer to Configuring an IP helper address on page 120.

To enable the forwarding of specific UDP application broadcasts, enter the following command.

```
device(config)# ip forward-protocol udp bootpc
```

**Syntax: [no] ip forward-protocol udp udp-port-name | udp-port-num**

The *udp-port-name* parameter can have one of the following values. For reference, the corresponding port numbers from RFC 1340 are shown in parentheses. If you specify an application name, enter the name only, not the parentheses or the port number shown here:

- bootpc (port 68)
- bootps (port 67)
- discard (port 9)
- dns (port 53)
- echo (port 7)
- mobile-ip (port 434)
- netbios-dgm (port 138)
- netbios-ns (port 137)
- ntp (port 123)
- tacacs (port 65)
- talk (port 517)
- time (port 37)
- tftp (port 69)

In addition, you can specify any UDP application by using the application's UDP port number.

The *udp-port-num* parameter specifies the UDP application port number. If the application you want to enable is not listed above, enter the application port number. You also can list the port number for any of the applications listed above.

To disable forwarding for an application, enter a command such as the following.

```
device(config)# no ip forward-protocol udp well known application port number
```

This command disables forwarding of specific UDP application requests to the helper addresses configured on Extreme device interfaces.

### *Configuring an IP helper address*

To forward a client's broadcast request for a UDP application when the client and server are on different networks, you must configure a helper address on the interface connected to the client. Specify the server's IP address or the subnet directed broadcast address of the IP subnet the server is in as the helper address.

You can configure up to 16 helper addresses on each interface. You can configure a helper address on an Ethernet port or a virtual interface.

To configure a helper address on interface 2 on chassis module 1, enter the following commands.

```
device(config)# interface e 1/2
device(config-if-e1000-1/2)# ip helper-address 10.95.7.6
```

The commands in this example change the CLI to the configuration level for port 1/2, then add a helper address for server 10.95.7.6 to the port. If the port receives a client request for any of the applications that the Extreme device is enabled to forward, the Extreme device forwards the client's request to the server.

**Syntax: [no] ip helper-address ip-addr**

The *ip-addr* command specifies the server's IP address or the subnet directed broadcast address of the IP subnet the server is in.

## Configuring BootP or DHCP forwarding parameters

A host on an IP network can use BootP or DHCP to obtain its IP address from a BootP or DHCP server. To obtain the address, the client sends a BootP or DHCP request. The request is a subnet directed broadcast and is addressed to UDP port 67. A limited IP broadcast is addressed to IP address 255.255.255.255 and is not forwarded by the Extreme device or other IP routers.

When the BootP or DHCP client and server are on the same network, the server receives the broadcast request and replies to the client. However, when the client and server are on different networks, the server does not receive the client's request, because the Extreme device does not forward the request.

You can configure the Extreme device to forward BootP or DHCP requests. To do so, configure a helper address on the interface that receives the client requests, and specify the BootP or DHCP server's IP address as the address you are helping the BootP or DHCP requests to reach. Instead of the server's IP address, you can specify the subnet directed broadcast address of the IP subnet the server is in.

> NOTE
> The IP subnet configured on the port which is directly connected to the device sending a BootP or DHCP request, does not have to match the subnet of the IP address given by the DHCP server.

## BootP or DHCP forwarding parameters

The following parameters control the Extreme device's forwarding of BootP or DHCP requests:

- **Helper address** - The BootP or DHCP server's IP address. You must configure the helper address on the interface that receives the BootP or DHCP requests from the client. The Extreme device cannot forward a request to the server unless you configure a helper address for the server.
- **Gateway address** - The Extreme device places the IP address of the interface that received the BootP or DHCP request in the request packet's Gateway Address field (sometimes called the Router ID field). When the server responds to the request, the server sends the response as a unicast packet to the IP address in the Gateway Address field. (If the client and server are directly attached, the Gateway ID field is empty and the server replies to the client using a unicast or broadcast packet, depending on the server.) By default, the Extreme device uses the lowest-numbered IP address on the interface that receives the request as the Gateway address. You can override the default by specifying the IP address you want the Extreme device to use.
- **Hop Count** - Each router that forwards a BootP or DHCP packet increments the hop count by 1. Routers also discard a forwarded BootP or DHCP request instead of forwarding the request if the hop count is greater than the maximum number of BootP or DHCP hops allowed by the router. By default, the Extreme device forwards a BootP or DHCP request if its hop count is four or less, but discards the request if the hop count is greater than four. You can change the maximum number of hops the Extreme device will allow to a value from 1 - 15.

  NOTE
  The BootP or DHCP hop count is not the TTL parameter.

## Configuring an IP helper address

The procedure for configuring a helper address for BootP or DHCP requests is the same as the procedure for configuring a helper address for other types of UDP broadcasts. Refer to Configuring an IP helper address on page 119.

## Changing the IP address used for stamping BootP or DHCP requests

When the Extreme device forwards a BootP or DHCP request, the Extreme device "stamps" the Gateway Address field. The default value the Extreme device uses to stamp the packet is the lowest-numbered IP address configured on the interface that received the request.

The BootP or DHCP stamp address is an interface parameter. Change the parameter on the interface that is connected to the BootP or DHCP client.

To change the IP address used for stamping BootP or DHCP requests received on interface 1/1, enter commands such as the following.

```
device(config)# int e 1/1
device(config-if-e1000-1/1)# ip bootp-gateway 10.157.22.26
```

These commands change the CLI to the configuration level for port 1/1, then change the BootP or DHCP stamp address for requests received on port 1/1 to 10.157.22.26. The Extreme device will place this IP address in the Gateway Address field of BootP or DHCP requests that the Extreme device receives on port 1/1 and forwards to the BootP or DHCP server.

**Syntax: [no] ip bootp-gateway ip-addr**

If the **ip bootp-source-address** command is configured on the interface where the BootP or DHCP request is received, then the configured address will be used as the source IP address for the forwarded packets.

```
device(config-if-e1000-1/1)# ip bootp-source-address 10.157.22.26
```

**Syntax: [no] ip bootp-source-address ip-addr**

### *Changing the maximum number of hops to a BootP relay server*

Each BootP or DHCP request includes a field Hop Count field. The Hop Count field indicates how many routers the request has passed through. When the Extreme device receives a BootP or DHCP request, the Extreme device looks at the value in the Hop Count field:

- If the hop count value is equal to or less than the maximum hop count the Extreme device allows, the Extreme device increments the hop count by one and forwards the request.
- If the hop count is greater than the maximum hop count the Extreme device allows, the Extreme device discards the request.

NOTE
The BootP or DHCP hop count is not the TTL parameter.

To modify the maximum number of BootP or DHCP hops, enter the following command.

```
device(config)# bootp-relay-max-hops 10
```

This command allows the Extreme device to forward BootP or DHCP requests that have passed through up to ten previous hops before reaching the Extreme device.

**Syntax: [no] bootp-relay-max-hops 1-15**

Default: 4

# Filtering Martian addresses

Martian addresses are obviously invalid host or network addresses. They commonly are sent by improperly configured systems on the network. Martian address filtering allows the system to automatically filter out those invalid addresses. When Martian address filtering is enabled, the BGP protocol applies the Martian address filters to all in-bound routes as received from all neighbors. Unlike BGP protocol, IGP protocols will rely on the RTM (routing table manager) to do the route filtering.

If no match is found, the route is accepted. This will be the case for almost all routes. If a match is found, the route is discarded (default action - deny), unless the action is set to permit. Martian address filtering is in addition to normal BGP in-bound route policies.

To enable Martian address filtering, enter the following command.

```
device(config)#  ip martian filtering-on
```

**Syntax: [no] ip martian** [ **vrf name** ] **filtering-on**

The **vrf** *name* option applies martian filtering to a specified VRF.

NOTE
Martian address filtering is disabled by default.

When Martian address filtering is first enabled, the device will automatically load the following default Martian addresses:

```
* 0.0.0.0/8
* 10.0.0.0/8
* 127.0.0.0/8
* 172.16.0.0/12
* 192.168.0.0/16
* 224.0.0.0/4
```

```
* 240.0.0.0/4
```

## Adding, deleting or modifying Martian addresses

As described previously, there are a set number of Martian addresses that are loaded by default when Martian addressing is enabled. You can add, subtract or modify addresses that are filtered by martian addressing. Although there is no limit of the number of martian address can be configured, it's expected the size of martian address list should be small, generally less than 100. If the user adds a new martian address after routes are already learnt, they will be taken out of the routing table. Likewise if the user removes a martian address after routes are deleted from the routing table, they should be put back into the routing table.

To add an address to the Martian filtering list, use a command such as the following.

```
device(config)# ip martian 192.168.0.0/16
```

**Syntax: [no] ip martian** [ **vrf name** ] **destination-prefix/prefix-length** [ **permit** ]

The *destination-prefix/prefix-length* variable specifies the address and the prefix range to apply the martian filtering to. The matching rule is for prefix range match. It includes exact match, or with a longer prefix length match. For example, if the Martian address rule is 192.168.0.0/16, then routes 192.168.0.0/16, and 192.168.1.0/24 are matches. However route 192.0.0.0/8 is not a match.

The **vrf** *name* option applies the modification to the martian filtering list to a specified VRF.

The **no command** removes an address from the martian filtering list.

The **[permit]** option changes the default action of a martian address filter to permit. In this case, a route matches the "permit" martian address is accepted by the routing table manager. This option is only used if a user wants to allow a prefix "hole" in an otherwise denied martian address.

The default Martian addresses are described in: Filtering Martian addresses on page 121

### *Examples*

To remove a user defined Martian address or a system default Martian address, use the "no" form of the command.

```
device(config)# no ip martian 0.0.0.0/8
```

The following example configuration, creates a "hole" for 192.168.1.0/24 in the martian address 192.168.0.0/16.

```
device(config)# ip martian 192.168.1.0/24 permit
device(config)# ip martian 192.168.0.0/16
```

To display the currently configured Martian addresses refer to Displaying martian addressing information on page 136.

# Displaying IP information

You can display the following IP configuration information statistics:

- **Global IP parameter settings** - refer to Displaying global IP configuration information on page 123.
- **IP interfaces** - refer to Displaying IP interface information on page 124.
- **ARP entries** - refer to Displaying ARP entries on page 32.
- **Static ARP entries** - refer to Displaying ARP entries on page 32.
- **IP forwarding cache** - refer to Displaying the forwarding cache on page 126.
- **IP route table** - refer to Displaying the IP route table on page 128.
- **IP traffic statistics** - refer to Displaying IP traffic statistics on page 132.

The sections below describe how to display this information.

In addition to the information described below, you can display the following IP information:

- RIP information
- OSPF information
- BGP4 information
- PIM information

# Displaying global IP configuration information

To display IP configuration information, enter the following command at any CLI level.

```
device> show ip
Global Settings
  IP CAM Mode: dynamic IPVPN CAM Mode: static
  ttl: 64, arp-age: 10, bootp-relay-max-hops: 4, icmp-error-rate: 400
  IP Router-Id: 10.5.5.5
  enabled : UDP-Broadcast-Forwarding ICMP-Redirect Source-Route Load-Sharing
  RARP BGP4 OSPF
  disabled: Directed-Broadcast-Forwarding drop-arp-pending-packets IRDP Proxy
  -ARP RPF-Check RPF-Exclude-Default RIP IS-IS VRRP VRRP-Extended VSRP
Configured Static Routes: 31
Configured Static Mroutes: 30
```

**Syntax: show ip**

> **NOTE**
> This command has additional options, which are explained in other sections in this guide, including the sections below this one.

This display shows the following information.

**TABLE 10** CLI display of global IP configuration information

| This field... | Displays... |
|---|---|
| **Global settings** | |
| ttl | The Time-To-Live (TTL) for IP packets. The TTL specifies the maximum number of router hops a packet can travel before reaching the Extreme device. If the packet's TTL value is higher than the value specified in this field, the device drops the packet.<br>To change the maximum TTL, refer to Changing the TTL threshold on page 100. |
| arp-age | The ARP aging period. This parameter specifies how many minutes an inactive ARP entry remains in the ARP cache before the device ages out the entry.<br>To change the ARP aging period, refer to Changing the ARP aging period on page 29. |
| bootp-relay-max-hops | The maximum number of hops away a BootP server can be located from the device and still be used by the device's clients for network booting.<br>To change this value, refer to Changing the maximum number of hops to a BootP relay server on page 121. |
| router-id | The 32-bit number that uniquely identifies the device.<br>By default, the router ID is the numerically lowest IP interface configured on the device. To change the router ID, refer to Changing the router ID on page 95. |
| enabled | The IP-related protocols that are enabled on the device. |

**TABLE 10** CLI display of global IP configuration information (continued)

| This field... | Displays... |
|---|---|
| disabled | The IP-related protocols that are disabled on the device. |

# Displaying IP interface information

To display IP interface information, enter the following command at any CLI level.

```
Interface       IP-Address      OK?  Method Status     Protocol VRF
eth 3/10        10.25.25.3      YES  NVRAM  down       down     default-vrf
eth 3/19        10.11.11.3      YES  NVRAM  up         up       default-vrf
eth 3/20        10.33.32.1      YES  NVRAM  up         up       default-vrf
mgmt 1          10.25.106.12    YES  NVRAM  up         up       default-vrf
loopback 1      10.5.5.5        YES  NVRAM  up         up       default-vrf
```

**Syntax: show ip interface** [ **ethernet slot/port** ] | [ **loopback num** ] | [ **ve num** ]

This display shows the following information.

**TABLE 11** CLI display of interface IP configuration information

| This field... | Displays... |
|---|---|
| Interface | The type and the slot and port number of the interface. |
| IP-Address | The IP address of the interface.<br><br>NOTE<br>If an "s" is listed following the address, this is a secondary address. When the address was configured, the interface already had an IP address in the same subnet, so the software required the "secondary" option before the software could add the interface. |
| OK? | Whether the IP address has been configured on the interface. |
| Method | Whether the IP address has been saved in NVRAM. If you have set the IP address for the interface in the CLI, but have not saved the configuration, the entry for the interface in the Method field is "manual". |
| Status | The link status of the interface. If you have disabled the interface with the **disable** command, the entry in the Status field will be "administratively down". Otherwise, the entry in the Status field will be either "up" or "down". |
| Protocol | Whether the interface can provide two-way communication. If the IP address is configured, and the link status of the interface is up, the entry in the protocol field will be "up". Otherwise the entry in the protocol field will be "down". |
| VRF | Specifies the VRF type applied to the interface. |

## *Displaying IP interface information for a specified interface*

To display detailed IP information for a specific interface, enter a command such as the following.

```
device# show ip interface ethernet e 3/1
Interface Ethernet 3/1 (80)
  port enabled
  port state: UP
  ip address: 10.1.1.2/24
  Port belongs to VRF: default
  encapsulation: ETHERNET, mtu: 1500
  MAC Address 0004.80a0.4050
  directed-broadcast-forwarding: disabled
  No inbound ip access-list is set
```

```
No outbound ip access-list is set
No Helper Addresses are configured.
RPF mode: None  RPF Log: Disabled
0 unicast RPF drop    0 unicast RPF suppressed drop
RxPkts:  1200  TxPkts:  1200
RxBytes: 60000 TxBytes: 60000
```

NOTE

Interface counters (received packets and received bytes) are not supported on the CES 2000 Series or the CER 2000 Series devices. These values will always be 0.

The Extreme device software supports IPv4 and IPv6 packet and byte counters. The contents of these counters is displayed for a defined port as the result of the show ip interface ethernet command. In the above example, the fields in bold text display this content. Table 12 describes each of the fields that display interface counter statistics.

**TABLE 12** Interface counter display statistics

| This field... | Displays... |
|---|---|
| Interface | The interface that counter statistics are being displayed for. |
| RxPkts | The number of packets received at the specified port. |
| TxPkts | The number of packets transmitted from the specified port. |
| RxBytes | The number of bytes received at the specified port. |
| TxBytes | The number of bytes transmitted from the specified port. |

## *Displaying interface counters for all ports*

The Extreme device supports IPv4 and IPv6 packet and byte counters. The contents of these counters can be displayed for all ports on a device or per-port.

Commands have been added under IPv4 and IPv6 to display the interface counters for all ports on a device. The following example uses the **show ip interface counters** command to display to packet and byte counter information for all ports.

```
device# show ip interface counters
Interface      RxPkts     TxPkts      RxBytes    TxBytes
eth 3/1         1200       1200        600000     60000
eth 3/2         500        500         25000      25000
```

**Syntax: show ip interface counters**

Default byte counters include the 20-byte per-packet Ethernet overhead. You can configure an Extreme device to exclude the 20-byte per-packet Ethernet overhead from byte accounting by configuring the **vlan-counter exclude-overhead** command. Displaying IP interface information for a specified interface on page 124 describes each of the fields that display interface counter statistics.

**TABLE 13** Interface counter display statistics

| This field... | Displays... |
|---|---|
| Interface | The interface that counter statistics are being displayed for. |
| RxPkts | The number of packets received at the specified port. |
| TxPkts | The number of packets transmitted from the specified port. |
| RxBytes | The number of bytes received at the specified port. |
| TxBytes | The number of bytes transmitted from the specified port. |

## *Clearing the interface counters*

Use the following command to clear all interface counters on a device.

NOTE
The **clear ip interface counters** command is available for the CES 2000 Series and the CER 2000 Series devices; however, the counters are not supported and the values will always be 0.

```
device# clear ip interface counters
```

**Syntax: clear ip interface counters**

Use the following command to clear the interface counters for a specified port.

```
device# clear ip interface ethernet 3/2
```

**Syntax: clear ip interface ethernet port-number**

The *port-number*variable specifies the slot and port number that you want to clear the interface counters for.

# Displaying interface name in Syslog

By default an interface's slot number (if applicable) and port number are displayed when you display Syslog messages. You can display the name of the interface instead of its number by entering a command such as the following.

```
device(config)# ip show-portname
```

This command is applied globally to all interfaces on the Extreme device.

**Syntax: [no] Ip show-portname**

When you display the messages in the Syslog, you see the interface name under the Dynamic Log Buffer section. The actual interface number is appended to the interface name. For example, if the interface name is "lab" and its port number is "2", you see "lab2" displayed as in the example below.

```
device# show logging
Syslog logging: enabled (0 messages dropped, 0 flushes, 0 overruns)
    Buffer logging: level ACDMEINW, 3 messages logged
    level code: A=alert C=critical D=debugging M=emergency E=error
          I=informational N=notification W=warning
Static Log Buffer:
Dec 15 19:04:14:A:Fan 1, fan on right connector, failed
Dynamic Log Buffer (50 entries):
Dec 15 18:46:17:I:Interface ethernet Lab2
, state up
Dec 15 18:45:15:I:Warm start
```

# Displaying the forwarding cache

To display the IP Forwarding Cache for directly connected hosts, enter the following command.

```
device> show ip cache
Cache Entry Usage on LPs:
Module    Host   Network      Free       Total
15         6      6           204788     204800
```

**Syntax: show ip cache** [ **ip-addr** ] [ | **begin expression** | **exclude expression** | **include expression** ]

The *ip-addr* parameter displays the cache entry for the specified IP address.

The **show ip cache** command shows the forwarding cache usage on each interface module CPU. The CPU on each interface module builds its own forwarding cache, depending on the traffic. To see the forwarding cache of a particular interface module, use the **rconsole**.

```
device>rconsole 15
Connecting to slave CPU 15/1... (Press CTRL-Shift-6 X to exit)
rconsole-15/1@LP>show ip cache
```

```
Total number of host cache entries 3
D: Dynamic P:Permanent, F:Forward U:Us C:Conected Network
W:Wait ARP  I:ICMP Deny  K:Drop R:Frament S:Snap Encap  N:CAMInvalid
    IP Address    Next Hop     MAC              Type    Port    VLAN     Pri
1   10.1.0.0      DIRECT       0000.0000.0000    PU      2/5     n/a      0
2   10.2.0.0      DIRECT       0125.0a57.1c02    D       3/5     n/a      0
3   10.7.7.3       DIRECT       0000.0000.0000    PU      4/2     12       1
```

You also use the **rconsole** to display the IP Forwarding Cache for network entries.

```
device>rconsole 15
Connecting to slave CPU 15/1... (Press CTRL-Shift-6 X to exit)
rconsole-15/1@LP>show ip network
Total number of host cache entries 3
D: Dynamic P:Permanent, F:Forward U:Us C:Conected Network
W:Wait ARP  I:ICMP Deny  K:Drop R:Frament S:Snap Encap  N:CAMInvalid
    IP Address       Next Hop     MAC              Type    Port    VLAN     Pri
1   0.0.0.0/0        DIRECT       0000.0000.0000    PK              n/a      0
2   10.1.1.0/24      DIRECT       0000.0000.0000    PC              n/a      0
3   10.40.40.0/24    10.2.1.10    0000.0000.0033    PF      15/14   154      1
```

The **show ip cache** and **show ip network** commands entered on the rconsole display the following information.

**TABLE 14** CLI display of IP forwarding cache

| This field... | Displays... |
|---|---|
| IP Address | The IP address of the destination. |
| Next Hop | The IP address of the next-hop router to the destination. This field contains either an IP address or the value DIRECT. DIRECT means the destination is either directly attached or the destination is an address on this device. For example, the next hop for loopback addresses and broadcast addresses is shown as DIRECT. |
| MAC | The MAC address of the destination.<br><br>**NOTE**<br>If the entry is type U (indicating that the destination is this device), the address consists of zeroes. |
| Type | The type of host entry, which can be one or more of the following:<br>• D - Dynamic<br>• P - Permanent<br>• F - Forward<br>• U - Us<br>• C - Complex Filter<br>• W - Wait ARP<br>• I - ICMP Deny<br>• K - Drop<br>• R - Fragment<br>• S - Snap Encap |
| Port | The port through which this device reaches the destination. For destinations that are located on this device, the port number is shown as "n/a". |
| VLAN | Indicates the VLANs the listed port is in. |
| Pri | The QoS priority of the port or VLAN. |

# Dual Active Console

The Dual Active Console command enables the standby terminal console to mirror the features of the active console, such that the standby console appears as active console itself. Hence, you can manage the system from either active or standby console and it will not be necessary to switch the console cable after the active-standby management module switchover.

To enable this feature, enter the following command,

```
device(config)#dual-active-console
device(config)#wr mem
Write startup-config done.
device(config)#
```

To disable this feature, enter the following command,

```
device(config)#no dual-active-console
device(config)#wr mem
Write startup-config done.
device(config)#
```

# Displaying the IP route table

To display the IP route table, enter the **show ip route** command at any CLI level.

```
device# show ip route
Total number of IP routes: 4
Type Codes - B:BGP D:Connected I:ISIS S:Static R:RIP O:OSPF; Cost - Dist/Metric
        Destination        Gateway         Port        Cost       Type Uptime
1       10.0.0.0/24        DIRECT          eth 1/1     0/0        D    45m18s
2       10.10.0.0/24        DIRECT          eth 1/2     0/0        D    1h0m
3       10.20.0.0/24        10.0.0.2        eth 1/1     1/1        S    13m18s
4       10.30.0.0/24       10.0.0.2        eth 1/1     1/1        S    2m42s
```

**Syntax: show ip route num** | [ **ip-addr** [ **ip-mask** ] [ **debug** | **detail** | **longer** ] ] | **connected** | **bgp** | **isis** | **ospf** | **rip** | **static** | [ **summary** ] | **nexthop** [ **nexthop_id** [ **ref-routes** ] ] | [ **begin expression** | **exclude expression** | **include expression** ]

The *num* option display the route table entry whose row number corresponds to the number you specify. For example, if you want to display the tenth row in the table, enter "10".

The *ip-addr* parameter displays the route to the specified IP address.

The *ip-mask* parameter lets you specify a network mask or, if you prefer CIDR format, the number of bits in the network mask. If you use CIDR format, enter a forward slash immediately after the IP address, then enter the number of mask bits (for example: 10.157.22.0/24 for 10.157.22.0 255.255.255.0).

The **longer, detail, and debug** parameter applies only when you specify an IP address and mask. This option displays only the routes for the specified IP address and mask.

The **bgp** option displays the BGP4 routes.

The **connected** option displays only the IP routes that are directly attached to the Extreme device.

The **ospf** option displays the OSPF routes.

The **rip** option displays the RIP routes.

The **isis** option displays the RIP routes.

The **static** option displays only the static IP routes.

The **nexthop** option displays next-hop information for all next hops in the routing table or for a specific entry.

## Showing route details by IP address

You can display detailed information about a route by providing the IP address and using the **detail** option, as the following example illustrates.

```
device>show ip route 10.1.1.2 detail
Type Codes - B:BGP D:Connected I:ISIS O:OSPF R:RIP S:Static; Cost - Dist/Metric
ISIS Codes - L1:Level-1 L2:Level-2
OSPF Codes - i:Inter Area 1:External Type 1 2:External Type 2 s:Sham Link
        Destination        Gateway         Port        Cost          Type Uptime
1       10.1.1.0/24        DIRECT          eth 1/15    0/0           D    7h11m
        Nexthop Entry ID:14, Paths: 1, Ref_Count:1/1
1       10.1.1.0/24        10.1.1.2        eth 1/15    115/20        IL2  7h11m
        10.1.1.0/24        10.0.0.18      eth 4/11    115/20        IL2  7h11m
        10.1.1.0/24        10.0.0.30      eth 4/7     115/20        IL2  7h11m
        10.1.1.0/24        10.0.0.34      eth 4/14    115/20        IL2  7h11m
        Nexthop Entry ID:68343, Paths: 4, Ref_Count:8/21
D:Dynamic  P:Permanent  F:Forward  U:Us  C:Connected Network
W:Wait ARP  I:ICMP Deny  K:Drop  R:Fragment  S:Snap Encap N:CamInvalid
Module S1:
      IP Address         Next Hop        MAC              Type  Port  Vlan  Pri
      10.1.1.0/24          DIRECT           0000.0000.0000     PC    n/a           0
      OutgoingIf  ArpIndex PPCR_ID   CamLevel   Parent  DontAge Index
       eth 1/15     65535    1:2        1              0    69203192   38
      U_flags   Entry_flags  Age   Cam:Index           Trunk_fid      Ecmp_count
       0000e220                0     0x1a8fc (L3, right)   0x00000(     0) 0
        CAM Entry Flag: 00000003H
        PPCR : 1:2 CIDX: 0x1a8fc (L3, right) (IP_NETWORK: 0x68703)
         PPCR : 1:1 CIDX: 0x1a8fc (L3, right) (IP_NETWORK: 0x68703)
```

**Syntax: show ip route ip_addr detail**

The IP address can be just the IP address but can also include shorthand for the mask: ip-address/prefix-length.

## Using the summary option

The **summary** option displays a summary of the information in the IP route table. After the **summary** keyword, the pipe symbol (|) points to three options for modifying the presentation of the summary information, as follows:

- **begin** lets you start the display with the first matching line.
- **exclude** lets you exclude matching lines from the display.
- **include** lets you include matching lines in the display.

The default routes are displayed first.

## Using the connected option

Here is an example of how to use the **connected** option. To display only the IP routes that go to devices directly attached to the Extreme device.

```
device(config)# show ip route connected
Type Codes - B:BGP D:Connected I:ISIS S:Static R:RIP O:OSPF; Cost - Dist/Metric
      Destination       Gateway           Port   Cost    Type   Uptime
1     10.157.22.0/24     0.0.0.0           4/11    1       D      1h0m
```

Notice that the route displayed in this example has "D" in the Type field, indicating the route is for a directly connected device.

## Using the static option

Here is an example of how to use the **static** option. To display only the static IP routes.

```
device(config)# show ip route static
Type Codes - B:BGP D:Connected I:ISIS S:Static R:RIP O:OSPF; Cost - Dist/Metric
      Destination       Gateway          Port   Cost   Type  Uptime
1     10.144.33.11/32  10.157.22.12     1/1    2      S     1h0m
```

Notice that the route displayed in this example has "S" in the Type field, indicating the route is static.

## Using the longer option

Here is an example of how to use the **longer** option. To display only the routes for a specified IP address and mask, enter a command such as the following.

```
device(config)# show ip route
10.159.0.0/16 longer
Type Codes - B:BGP D:Connected I:ISIS S:Static R:RIP O:OSPF; Cost - Dist/Metric
   Destination       Gateway          Port   Cost   Type  Uptime
52 10.159.38.0/24   10.95.6.101      1/1    1      S     45m18s
53 10.159.39.0/24   10.95.6.101      1/1    1      S     1h0m
54 10.159.40.0/24   10.95.6.101      1/1    1      S     45m18s
55 10.159.41.0/24   10.95.6.101      1/1    1      S     1h0m
56 10.159.42.0/24   10.95.6.101      1/1    1      S     13m18s
```

This example shows all the routes for networks beginning with 209.159. The mask value and **longer** parameter specify the range of network addresses to be displayed. In this example, all routes within the range 209.159.0.0 - 209.159.255.255 are listed.

## Using the summary option

The **summary** option displays a summary of the information in the IP route table. The following is an example of the output from this command.

```
device# show ip route summary
IP Routing Table - 35 entries:
  6 connected, 28 static, 0 RIP, 1 OSPF, 0 BGP, 0 ISIS, 0 MPLS
  Number of prefixes:
  /0: 1 /16: 27 /22: 1 /24: 5 /32: 1
```

**Syntax: show ip route summary**

In this example, the IP route table contains 35 entries. Of these entries, 6 are directly connected devices, 28 are static routes, and 1 route was calculated through OSPF. One of the routes has a zero-bit mask (this is the default route), 27 have a 22-bit mask, 5 have a 24-bit mask, and 1 has a 32-bit mask.

## Using the nexthop option

You can display next-hop information for all next hops in the routing table or for a specific entry. For the first example, use the **show ip route nexthop** command to display all the next-hop entries, and then use the option to display the next hop for a specific table entry.

```
device#show ip route nexthop
Total number of IP nexthop entries: 30; Forwarding Use: 24
       NextHopIp        Port            RefCount    ID      Age
1      0.0.0.0         mgmt 1          0/1         1536    80682
2      0.0.0.0         eth 1/15        1/1         14      80632
3      0.0.0.0         eth 1/16        1/1         15      16626
4      0.0.0.0         eth 1/18        1/1         17      16626
5      0.0.0.0         eth 1/43        1/1         42      35923
6      0.0.0.0         eth 1/47        1/1         46      80641
7      0.0.0.0         eth 2/2         1/1         49      16630
8      0.0.0.0         eth 2/4         1/1         51      16630
```

```
9       10.1.1.2         eth 1/15       0/2         68347   16620
        10.1.2.2         eth 1/18
        10.0.0.18       eth 4/11
        10.0.0.25       eth 4/9
10      10.1.1.2         eth 1/15       0/3         68352   16615
        10.0.0.6        eth 4/4
        10.0.0.10       eth 2/2
        10.0.0.21       eth 4/1
11      0.0.0.0          eth 4/1        1/1         144     16624
12      0.0.0.0          eth 4/3        1/1         146     16641
13      0.0.0.0          eth 4/4        1/1         147     16624
14      0.0.0.0          eth 4/6        1/1         149     16624
15      0.0.0.0          eth 4/7        1/1         150     16641
```

**Syntax: show ip route nexthop** [ **nexthop_id** ]

The *nexthop_id* is under the column labeled ID in the output of the **show ip route nexthop** command. For example, use nexthop ID 1536 from the first row of the preceding example to show only that entry.

```
device#show ip route nexthop 1536
       NextHopIp        Port             RefCount      ID      Age
1      0.0.0.0          mgmt 1           0/1           1536    80685
```

## Displaying IP routes with nexthop ID

By using the **nexthop** option with the **ref-routes** keyword, you can display IP routes in the forwarding table that refer to the specified nexthop entry, as the following example illustrates (using nexthop ID 65575).

```
device#show ip route nexthop 65537 ref-routes
Type Codes - B:BGP D:Connected I:ISIS O:OSPF R:RIP S:Static; Cost - Dist/Metric
ISIS Codes - L1:Level-1 L2:Level-2
OSPF Codes - i:Inter Area 1:External Type 1 2:External Type 2 s:Sham Link
        Destination        Gateway         Port        Cost          Type Uptime
1       10.1.1.1/32        10.2.1.1        eth 1/11    115/10        IL2  7h51m
2       10.1.1.0/24        10.2.1.1        eth 1/11    115/10        IL2  7h51m
3       10.1.1.1/32        10.2.1.1        eth 1/11    115/40        IL2  7h51m
```

**Syntax: show ip route nexthop** [ **nexthop_id** [ **ref-routes** ] ]

## Description of command output fields

The following table lists the information in the **show ip route** output when you use no optional arguments.

**TABLE 15** CLI display of IP route table

| This field... | Displays... |
|---|---|
| Destination | The destination network of the route. |
| NetMask | The network mask of the destination address. |
| Gateway | The next-hop router. |
| Port | The port through which this device sends packets to reach the route's destination. |
| Cost | The route's cost. |
| Type | The route type, which can be one of the following:<br>• B - The route was learned from BGP.<br>• D - The destination is directly connected to this Extreme device.<br>• R - The route was learned from RIP.<br>• S - The route is a static route.<br>• * - The route is a candidate default route. |

**TABLE 15** CLI display of IP route table (continued)

| This field... | Displays... |
|---|---|
| | • O - The route is an OSPF route. Unless you use the ospf option to display the route table, "O" is used for all OSPF routes. If you do use the ospf option, the following type codes are used:<br>• O - OSPF intra area route (within the same area).<br>• IA - The route is an OSPF inter area route (a route that passes from one area into another).<br>• E1 - The route is an OSPF external type 1 route.<br>• E2 - The route is an OSPF external type 2 route. |
| Uptime | The amount of time since the route was last modified. The format of this display parameter may change depending upon the age of the route to include the seconds (s), minutes (m), hours (h), and days (d), as described in the following:<br>400d - Only days (d) displayed<br>20d23h - days (d) and hours (h) displayed<br>14h33m - hours (h) and minutes (m) displayed<br>10m59s - minutes (m) and seconds (s) displayed |

# Clearing IP routes

If needed, you can clear the entire route table or specific individual routes.

To clear all routes from the IP route table.

```
device# clear ip route
```

To clear route 10.157.22.0/24 from the IP routing table.

```
device# clear ip route 10.157.22.0/24
```

**Syntax: clear ip route** [ **ip-addr ip-mask** | **ip-addr/mask-bits** ]

# Displaying IP traffic statistics

To display IP traffic statistics, enter the following command at any CLI level.

> **NOTE**
> In the Extreme device, only those packets that are forwarded or generated by the CPU are included in the IP traffic statistics. Hardware forwarded packets are not included.

```
device# show ip traffic
IP Statistics
  1265602 total received, 690204 mp received, 225395 sent, 0 forwarded
  0 filtered, 0 fragmented, 0 bad header
  0 failed reassembly, 0 reassembled, 0 reassembly required
  2951 no route, 0 unknown proto, 0 no buffer, 0 other errors
ARP Statistics
  489279 total recv, 488154 req recv, 1125 rep recv, 1159 req sent, 3960 rep sent
  0 pending drop, 0 invalid source, 0 invalid dest
ICMP Statistics
Received:
  0 total, 0 errors, 0 unreachable, 0 time exceed
  0 parameter, 0 source quench, 0 redirect, 0 echo, 0 echo reply
  0 timestamp, 0 timestamp reply, 0 address mask, 0 address mask reply
  0 irdp advertisement, 0 irdp solicitation
```

```
Sent:
  2146 total, 0 errors, 2146 unreachable, 0 time exceed (0 mpls-response)
  0 parameter, 0 source quench, 0 redirect, 0 echo, 0 echo reply
  0 timestamp, 0 timestamp reply, 0 address mask, 0 address mask reply
  0 irdp advertisement, 0 irdp solicitation
UDP Statistics
  184784 received, 75473 sent, 110196 no port, 0 input errors
TCP Statistics
  86199 in segments, 84392 out segments, 909 retransmission, 0 input errors
ip packet list pool
pool: 237598e3, unit_size: 9362, initial_number:32, upper_limit:128
  total_number:32, allocated_number:0, alloc_failure 0
  flag: 0, pool_index:1, avail_data:27100000
ip reassembly list pool
pool: 23759783, unit_size: 23, initial_number:16, upper_limit:64
  total_number:16, allocated_number:0, alloc_failure 0
  flag: 0, pool_index:1, avail_data:270df800
ip fragments list pool
pool: 23759833, unit_size: 20, initial_number:32, upper_limit:128
  total_number:32, allocated_number:0, alloc_failure 0
  flag: 0, pool_index:1, avail_data:270e0800
```

### Syntax: show ip traffic

The **show ip traffic** command displays the following information.

**TABLE 16** CLI display of IP traffic statistics

| This field... | Displays... |
|---|---|
| **IP statistics** | |
| received | The total number of IP packets received by the device. |
| sent | The total number of IP packets originated and sent by the device. |
| forwarded | The total number of IP packets received by the device and forwarded to other devices. |
| filtered | The total number of IP packets filtered by the device. |
| fragmented | The total number of IP packets fragmented by this device to accommodate the IP MTU of this device or of another device. |
| reassembled | The total number of fragmented IP packets that this device re-assembled. |
| bad header | The number of IP packets dropped by the device due to a bad packet header. |
| no route | The number of packets dropped by the device because there was no route. |
| unknown proto | The number of packets dropped by the device because the value in the Protocol field of the packet header is unrecognized by this device. |
| no buffer | This information is used by Extreme customer support. |
| other errors | The number of packets that this device dropped due to error types other than the types listed above. |
| **ICMP statistics**<br>The ICMP statistics are derived from RFC 792, "Internet Control Message Protocol", RFC 950, "Internet Standard Subnetting Procedure", and RFC 1256, "ICMP Router Discovery Messages". Statistics are organized into Sent and Received. The field descriptions below apply to each. | |
| total | The total number of ICMP messages sent or received by the device. |
| errors | This information is used by Extreme customer support. |
| unreachable | The number of Destination Unreachable messages sent or received by the device. |
| time exceed | The number of Time Exceeded messages sent or received by the device. |
| parameter | The number of Parameter Problem messages sent or received by the device. |

**TABLE 16** CLI display of IP traffic statistics (continued)

| This field... | Displays... |
|---|---|
| source quench | The number of Source Quench messages sent or received by the device. |
| redirect | The number of Redirect messages sent or received by the device. |
| echo | The number of Echo messages sent or received by the device. |
| echo reply | The number of Echo Reply messages sent or received by the device. |
| timestamp | The number of Timestamp messages sent or received by the device. |
| timestamp reply | The number of Timestamp Reply messages sent or received by the device. |
| addr mask | The number of Address Mask Request messages sent or received by the device. |
| addr mask reply | The number of Address Mask Replies messages sent or received by the device. |
| irdp advertisement | The number of ICMP Router Discovery Protocol (IRDP) Advertisement messages sent or received by the device. |
| irdp solicitation | The number of IRDP Solicitation messages sent or received by the device. |
| **UDP statistics** | |
| received | The number of UDP packets received by the device. |
| sent | The number of UDP packets sent by the device. |
| no port | The number of UDP packets dropped because the packet did not contain a valid UDP port number. |
| input errors | This information is used by Extreme customer support. |
| **TCP statistics**<br>The TCP statistics are derived from RFC 793, "Transmission Control Protocol". | |
| active opens | The number of TCP connections opened by this device by sending a TCP SYN to another device. |
| passive opens | The number of TCP connections opened by this device in response to connection requests (TCP SYNs) received from other devices. |
| failed attempts | This information is used by Extreme customer support. |
| active resets | The number of TCP connections this device reset by sending a TCP RESET message to the device at the other end of the connection. |
| passive resets | The number of TCP connections this device reset because the device at the other end of the connection sent a TCP RESET message. |
| input errors | This information is used by Extreme customer support. |
| in segments | The number of TCP segments received by the device. |
| out segments | The number of TCP segments sent by the device. |
| retransmission | The number of segments that this device retransmitted because the retransmission timer for the segment had expired before the device at the other end of the connection had acknowledged receipt of the segment. |

# Displaying GRE tunnel information and statistics

Several show commands display information about configured GRE tunnels.

You must enable GRE statistics gathering using the **accounting-enable** and the **gre-session-enforce-check** commands under IP Tunnel Policy configuration mode.

These commands are optional and do not have to be entered in any specific order. Checking the output is recommended to verify the configuration and operation of the GRE tunnels.

NOTE

When reviewing the keepalive packet statistics in the output of the show interface tunnel command for a GRE tunnel, note that the transmitted keepalive packets are hardware generated and are not counted in the "Rcv-from-tnnl" and "Xmit-to-tnnl" statistics.

1. To display information about all GRE tunnels configured on a device, enter the following command.

```
device# show gre
Total Valid GRE Tunnels : 1, GRE Session Check Enforce: FALSE
GRE tnnl 1 UP : src_ip 10.25.25.4, dst_ip 10.15.15.3
   TTL 255, TOS 0, NHT 0, MTU 1476
```

2. Use the **show statistics tunnel** command with a specific *tunnel-id* to display statistics for a single tunnel. In this example, the tunnel type is GRE.

```
device# show statistics tunnel 1
                                              Packets
Tunnel Id  Tunnel Type  In-Port(s)    [Rcv-from-tnnl   Xmit-to-tnnl]
1          GRE          e2/1 - e2/2   586046           287497
                        e2/3 - e2/4   100340           150034
```

3. Use the **show statistics brief tunnel** command to display the aggregate statistics for a specific tunnel or for all tunnels. The feature combines both unicast and multicast statistics into one counter.

```
device# show statistics brief tunnel

                                    Packets
Tunnel Id  Tunnel Type   [Rcv-from-tnnl   Xmit-to-tnnl]
1          GRE           586046           287497
2          GRE           0                0
3          IPV6-Manual   0                0
```

4. Use the **show interface tunnel** command to display information about one or all of the tunnels.

```
device# show interface tunnel 1

Tunnel 1 is up, line protocol is up
  Hardware is Tunnel
  Tunnel source  10.30.30.1
  Tunnel destination is 10.20.20.1
  Tunnel mode gre ip
  No port name
  Internet address is: 10.50.50.4/24
  Tunnel TOS 0, Tunnel TTL 255, Tunnel MTU 1476 bytes
  Keepalive is not Enabled
  Tunnel Packet Statistics:
                      Unicast Packets                Multicast Packets
    In-Port(s)     [Rcv-from-tnnl  Xmit-to-tnnl]   [Rcv-from-tnnl  Xmit-to-tnnl]
    e5/1 - e5/20   0               16511754        0               0
    e6/1 - e6/20   0               14147748        0               20195730
    e7/1 - e7/24   21493545        0               40696309        0
    e16/1 - e16/2  0               3916998         0               0
    e16/3 - e16/4  0               13476342        0               0
```

# Displaying martian addressing information

To display Martian Addressing information, use the following command.

```
device# show ip martian
ip martian filtering on
0.0.0.0/8 deny
10.0.0.0/8 deny
127.0.0.0/8 deny
191.255.0.0/16 deny
192.0.0.0/24 deny
223.255.255.0/24 deny
240.0.0.0/4 deny
```

**Syntax: show** [ **vrf name** ] **ip martian**

You can use the **vrf** option to display martian addresses for a specific VRF.

# IPv6 Addressing



# IPv6 addressing overview

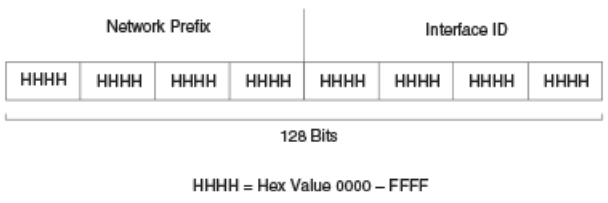
This chapter includes overview information about the following topics:

- IPv6 addressing.
- The IPv6 stateless auto-configuration feature, which enables a host on a local link to automatically configure its interfaces with new and globally unique IPv6 addresses associated with its location.

A limitation of IPv4 is its 32-bit addressing format, which is unable to satisfy potential increases in the number of users, geographical needs, and emerging applications. To address this limitation, IPv6 introduces a new 128-bit addressing format.

An IPv6 address is composed of 8 fields of 16-bit hexadecimal values separated by colons (:). Figure 15 shows the IPv6 address format.

**FIGURE 15** IPv6 address format



As shown in Figure 15, HHHH is a 16-bit hexadecimal value, while H is a 4-bit hexadecimal value. The following is an example of an IPv6 address:

2001:DB8:0000:0000:002D:D0FF:FE48:4672

Note that the sample IPv6 address includes hexadecimal fields of zeros. To make the address less cumbersome, you can do the following:

- Omit the leading zeros; for example, 2001:DB8:0:0:2D:D0FF:FE48:4672.
- Compress the successive groups of zeros at the beginning, middle, or end of an IPv6 address to two colons (::) once per address; for example, 2001:DB8::2D:D0FF:FE48:4672.

When specifying an IPv6 address in a command syntax, keep the following in mind:

- You can use the two colons (::) once in the address to represent the longest successive hexadecimal fields of zeros.
- The hexadecimal letters in the IPv6 addresses are not case-sensitive.

As shown in Figure 15, the IPv6 network prefix is composed of the left-most bits of the address. As with an IPv4 address, you can specify the IPv6 prefix using the *prefix* or *prefix-length* format, where the following applies:

The *prefix* parameter is specified as 16-bit hexadecimal values separated by a colon.

The *prefix-length* parameter is specified as a decimal value that indicates the left-most bits of the IPv6 address.

The following is an example of an IPv6 prefix:

2001:DB8:49EA:D088::/64

# IPv6 address types

As with IPv4 addresses, you can assign multiple IPv6 addresses to a device interface. Table 17 presents the three major types of IPv6 addresses that you can assign to a device interface.

A major difference between IPv4 and IPv6 addresses is that IPv6 addresses support **scope**, which describes the topology in which the address may be used as a unique identifier for an interface or set of interfaces.

Unicast and multicast addresses support scoping as follows:

- Unicast addresses support two types of scope: global scope and local scope. In turn, local scope supports link-local addresses. Table 17 describes global and link-local addresses and the topologies in which they are used.
- Multicast addresses support a scope field, which Table 17 describes.

**TABLE 17** IPv6 address types

| Address type | Description | Address structure |
|---|---|---|
| Unicast | An address for a single interface. A packet sent to a unicast address is delivered to the interface identified by the address. | Depends on the type of the unicast address:<br>• Aggregatable global address -- An address equivalent to a global or public IPv4 address. The address structure is as follows: a fixed prefix of 2000::/3 (001), a 45-bit global routing prefix, a 16-bit subnet ID, and a 64-bit interface ID.<br>• Unique local address -- An address used within a site or intranet. For more information on ULAs, refer to RFC 4193.<br>• Link-local address -- An address used between directly connected nodes on |

**TABLE 17** IPv6 address types (continued)

| Address type | Description | Address structure |
|---|---|---|
| | | a single network link. The address structure is as follows: a fixed prefix of FE80::/10 (1111 1110 10) and a 64-bit interface ID.<br>• IPv4-compatible address -- An address used in IPv6 transition mechanisms that tunnel IPv6 packets dynamically over IPv4 infrastructures. The address embeds an IPv4 address in the low-order 32 bits and the high-order 96 bits are zeros. The address structure is as follows: 0:0:0:0:0:0:A.B.C.D.<br>• Loopback address -- An address (0:0:0:0:0:0:0:1 or ::1) that a device can use to send an IPv6 packet to itself. You cannot assign a loopback address to a physical interface.<br>• Unspecified address -- An address (0:0:0:0:0:0:0:0 or ::) that a node can use as a source address only until the node has its own address that is auto-configured. |
| Multicast | An address for a set of interfaces belonging to different nodes. Sending a packet to a multicast address results in the delivery of the packet to all interfaces in the set. | A multicast address has a fixed prefix of FF00::/8 (1111 1111). The next 4 bits define the address as a permanent or temporary address. The next 4 bits define the scope of the address (node, link, site, organization, global). |
| Anycast | An address for a set of interfaces belonging to different nodes. Sending a packet to an anycast address results in the delivery of the packet to the closest interface identified by the address. | An anycast address looks similar to a unicast address, because it is allocated from the unicast address space. If you assign a unicast address to multiple interfaces, it is an anycast address. An interface assigned an anycast address must be configured to recognize the address as an anycast address.<br>An anycast address can be assigned to a router only.<br>An anycast address must not be used as the source address of an IPv6 packet. |

A device automatically configures a link-local unicast address for an interface by using the prefix of FE80::/10 (1111 1110 10) and a 64-bit interface ID. The 128-bit IPv6 address is then subjected to duplicate address detection to ensure that the address is unique on the link. If desired, you can override this automatically configured address by explicitly configuring an address.

# IPv6 stateless auto-configuration

Extreme devices use the IPv6 stateless auto-configuration feature to enable a host on a local link to automatically configure its interfaces with new and globally unique IPv6 addresses associated with its location. The automatic configuration of a host interface is performed without the use of a server, such as a Dynamic Host Configuration Protocol (DHCP) server, or manual configuration.

The automatic configuration of a host interface works in the following way: a router on a local link periodically sends router advertisement messages containing network-type information, such as the 64-bit prefix of the local link and the default route, to all nodes on the link.

When a host on the link receives the message, it takes the local link prefix from the message and appends a 64-bit interface ID, thereby automatically configuring its interface. (The 64-bit interface ID is derived from the MAC address of the host's NIC.) The 128-bit IPv6 address is then subjected to duplicate address detection to ensure that the address is unique on the link.

The duplicate address detection feature verifies that a unicast IPv6 address is unique before it is assigned to a host interface by the stateless auto configuration feature. Duplicate address detection uses neighbor solicitation messages to verify that a unicast IPv6 address is unique.

> NOTE
> For the stateless auto configuration feature to work properly, the advertised prefix length in router advertisement messages must always be 64 bits.

The IPv6 stateless auto-configuration feature can also automatically reconfigure a host's interfaces if you change the ISP for the host's network. (The host's interfaces must be renumbered with the IPv6 prefix of the new ISP.)

The renumbering occurs in the following way: a router on a local link periodically sends advertisements updated with the prefix of the new ISP to all nodes on the link. (The advertisements still contain the prefix of the old ISP.) A host can use the addresses created from the new prefix and the existing addresses created from the old prefix on the link. When you are ready for the host to use the new addresses only, you can configure the lifetime parameters appropriately using the **ipv6 nd prefix-advertisement** command. During this transition, the old prefix is removed from the router advertisements. At this point, only addresses that contain the new prefix are used on the link.

# Enabling IPv6 routing

By default, IPv6 routing is enabled. If forwarding of IPv6 traffic globally on the device has been disabled, you can enable it by entering the following command.

```
device(config)# ipv6 unicast-routing
```

**Syntax: [no] ipv6 unicast-routing**

To disable the forwarding of IPv6 traffic globally on the device, enter the **no** form of this command.

> NOTE
> Downgrading from release 04.1.00 to an earlier release of the software can impact IPv6 routing. In earlier versions of the NetIron software, IPv6 routing was disabled by default. As of release 04.1.00, IPv6 routing is enabled by default and therefore does not appear in the configuration. If you are downgrading from 04.1.00 to an earlier version of the software and want IPv6 routing to be enabled, you must add the line "ipv6 unicast-routing" to the configuration.

# Configuring IPv6 on each interface

To forward IPv6 traffic on an interface, the interface must have an IPv6 address, or IPv6 must be explicitly enabled. By default, an IPv6 address is not configured on an interface.

If you choose to configure a global or unique local IPv6 unicast address (ULA) for an interface, IPv6 is also enabled on the interface. Further, when you configure a global or unique local IPv6 unicast address, you must decide on one of the following in the low-order 64 bits:

- A manually configured interface ID.
- An automatically computed EUI-64 interface ID.

If you prefer to assign a link-local IPv6 address to the interface, you must explicitly enable IPv6, which causes a link-local address to be automatically computed for the interface. If preferred, you can override the automatically configured link-local address with an address that you manually configure.

This section provides the following information:

- Configuring a global or unique local IPv6 unicast address with a manually configured or automatically computed interface ID for an interface.
- Automatically or manually configuring a link-local address for an interface.
- Configuring IPv6 anycast addresses

NOTE
On XMR Series and MLX Series devices, the IPv6 packet received with the DA MAC as the router's MAC is subjected to an IPv6 route lookup irrespective of IPv6 routing enabled on the interface.

# Configuring a global or unique local IPv6 unicast address

Configuring a global or unique local IPv6 unicast address on an interface does the following:

- Automatically configures an interface ID (a link-local address), if specified.
- Enables IPv6 on that interface.

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

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

The neighbor discovery feature sends messages to these multicast groups. For more information, refer to Configuring IPv6 neighbor discovery on page 176.

## *Configuring a global or unique loal IPv6 unicast address with a manually configured interface ID*

To configure a global or unique local IPv6 unicast address, including a manually configured interface ID, for an interface, enter commands such as the following.

```
device(config)# interface ethernet 3/1
device(config-if-e100-3/1)# ipv6 address 2001:DB8:12D:1300:240:D0FF:
FE48:4672/64
```

These commands configure the global prefix 2001:DB8:12d:1300::/64 and the interface ID::240:D0FF:FE48:4672, and enable IPv6 on Ethernet interface 3/1.

**Syntax: ipv6 address** *ipv6-prefix/prefix-length*

You must specify the *ipv6-prefix* parameter in hexadecimal using 16-bit values between colons as documented in RFC 2373.

You must specify the *prefix-length* parameter as a decimal value. A slash mark (/) must follow the *ipv6-prefix* parameter and precede the *prefix-length* parameter.

### *Configuring a global or unique local IPv6 unicast address with an automatically computed EUI-64 interface ID*

To configure a global or unique local IPv6 unicast address with an automatically computed EUI-64 interface ID in the low-order 64-bits, enter commands such as the following.

```
device(config)# interface ethernet 3/1
device(config-if-e100-3/1)# ipv6 address 2001:DB8:12D:1300::/64 eui-64
```

These commands configure the global prefix 2001:DB8:12d:1300::/64 and an interface ID, and enable IPv6 on Ethernet interface 3/1.

**Syntax: [no] ipv6 address** *ipv6-prefix/prefix-length* **eui-64**

You must specify the *ipv6-prefix* parameter in hexadecimal using 16-bit values between colons as documented in RFC 2373.

You must specify the *prefix-length* parameter as a decimal value. A slash mark (/) must follow the *ipv6-prefix* parameter and precede the *prefix-length* parameter.

The **eui-64** keyword configures the global or unique local unicast address with an EUI-64 interface ID in the low-order 64 bits.The interface ID is automatically constructed in IEEE EUI-64 format using the interface's MAC address.

## Configuring a link-local IPv6 address

To explicitly enable IPv6 on an interface without configuring a global or unique local unicast address for the interface, enter commands such as the following.

```
device(config)# interface ethernet 3/1
device(config-if-e100-3/1)# ipv6 enable
```

These commands enable IPv6 on Ethernet interface 3/1 and specify that the interface is assigned an automatically computed link-local address.

**Syntax: [no] ipv6 enable**

> NOTE
> When configuring VLANs that share a common tagged interface with a Virtual Ethernet (VE) interface, it is recommended that you override the automatically computed link-local address with a manually configured unique address for the interface. If the interface uses the automatically computed address, which in the case of VE interfaces is derived from a global MAC address, all VE interfaces will have the same MAC address.

To override a link-local address that is automatically computed for an interface with a manually configured address, enter commands such as the following.

```
device(config)# interface ethernet 3/1
device(config-if-e100-3/1)# ipv6 address FE80::240:D0FF:FE48:4672 link-local
```

These commands explicitly configure the link-local address FE80::240:D0FF:FE48:4672 for Ethernet interface 3/1.

**Syntax: [no] ipv6 address** *ipv6-address* **link-local**

You must specify the *ipv6-address* parameter in hexadecimal using 16-bit values between colons as documented in RFC 2373.

The **link-local** keyword indicates that the Extreme device interface should use the manually configured link-local address instead of the automatically computed link-local address.

# Configuring IPv6 anycast addresses

In IPv6, an **anycast** address is an address for a set of interfaces that belong to different nodes. Sending a packet to an anycast address results in the delivery of the packet to the closest interface that has an anycast address.

An anycast address looks similar to a unicast address, because it is allocated from the unicast address space. If you assign an IPv6 unicast address to multiple interfaces, it is an anycast address. On the device, you configure an interface assigned an anycast address to recognize the address as an anycast address.

For example, the following commands configure an anycast address on interface 2/1.

```
device(config)# int e 2/1
device(config-if-e100-2/1)# ipv6 address 2001:db8::6/64 anycast
```

**Syntax: [no] ipv6 address** *ipv6-prefix* | *prefix-length* [ **anycast** ]

IPv6 anycast addresses are described in detail in RFC 1884. See RFC 2461 for a description of how the IPv6 Neighbor Discovery mechanism handles anycast addresses.

**IPv6 anycast filtering**

By default all IPv6 packets with anycast address as destination will be processed. The following command provides options to selectively enable protocols or disable all protocols.

```
Extreme(config)# ipv6 anycast-no-response allow tcp
```

**Syntax:[no] ipv6 anycast-no-response [allow tcp|udp|icmp]**

The **allow tcp | udp | icmp** specifies the protocol to allow for processing.

NOTE

1. The **allow** options can also be used as standalone commands. If **ipv6 anycast-no-response** is already configured, it is modified based on the specified filters.
2. User can enable generation of TCP resets for incoming TCP packets with destination set as anycast address, by configuring **ip tcp enable-reset** command. However, if **ipv6 anycast-no-response** command is also enabled, this command becomes void since all anycast packets are blocked. If user requires reset to be sent for incoming anycast TCP packets, user has to configure **ipv6 anycast-no-response allow tcp** to unblock the incoming TCP packets.

# Configuring IPv6 127 bit mask address

With 127 bit mask we will have 127 bits in the network part of the address, and 1 bit in the host part of the address. With 1 bit in the host part, we can have only two IPv6 addresses, one for each host. With 127 bit mask we consider 0 and 1 as host address and eliminates subnet-anycast for the configured network from that link.

NOTE

The 127 bit mask address supports only inter-router Point-to-Point links.

# Benefits of using 127 bit mask:

- Eliminates the Ping-pong issue
- Reduces the impact of Denial of Service (DOS) attacks
- Saves IPv6 address space

For example, the following commands configure an 127 bit mask IPv6 address:

```
device(config)# interface ethernet 1/1
device(config-if-e10000-1/1)#enable
device(config-if-e10000-1/1)#vrf forwarding green
device(config-if-e10000-1/1)#ipv6 address 10:1:1::1/127
device(config-if-e10000-1/1)#ipv6 enable
device(config-if-e10000-1/1)#ipv6 ospf area 0
device(config)# interface ethernet 1/2
device(config-if-e10000-1/2)#enable
device(config-if-e10000-1/2)#ip address 8.8.8.1/24
device(config-if-e10000-1/2)#ipv6 address 1:1:1:1:1::/127
device(config-if-e10000-1/2)#ipv6 enable
```

# Configuring the management port for an IPv6 automatic address configuration

You can configure the management port to automatically obtain an IPv6 address. The process is the same for all ports and is described in detail in the Configuring a global or unique local IPv6 unicast address with an automatically computed EUI-64 interface ID on page 142

# IPv6 host support

You can configure the device to be an IPv6 host. An IPv6 host has interfaces with IPv6 addresses, but does not have IPv6 routing enabled.

This section lists supported and unsupported IPv6 host features.

## IPv6 host supported features

The following IPv6 host features are supported:

- Automatic address configuration

  NOTE
  Automatic IPv6 address configuration is supported, however, automatic configuration of an IPv6 *global* address is supported only if there is an IPv6 router present on the network. Manual IPv6 address configuration is not supported.

- HTTP/HTTPS over IPv6
- IPv6 ping
- Telnet using an IPv6 address
- TFTP using an IPv6 address
- Trace route using an IPv6 address
- Name to IPv6 address resolution using IPv6 DNS Server
- IPv6 access lists
- IPv6 debugging
- SSH version 1 over IPv6
- SNMP over IPv6
- Logging (Syslog) over IPv6

- MLD version 1 and version 2

See IPv6 Addressing on page 137 for additional support information

# Restricting SNMP access to an IPv6 node

You can restrict SNMP access (which includes Extreme Network Advisor access) to a specified IPv6 host. Enter a command such as the following.

```
device(config)# snmp-client ipv6 2001:DB8:efff:89::23
```

**Syntax: [no] snmp-client ipv6** *ipv6-address*

The *ipv6-address* must be in hexadecimal format using 16-bit values between colons, as documented in RFC 2373.

> NOTE
> You cannot use the following IPv6 addresses with the **snmp-client ipv6** *ipv6-address* command: :: (unspecified address), ff02::01 (all nodes address), and ff02:02 (all routers address)).

# Specifying an IPv6 SNMP trap receiver

You can specify an IPv6 host to be a trap receiver so that all SNMP traps are sent to the same SNMP trap receiver or set of receivers, typically one or more host devices on the network. Enter a command such as the following.

```
device(config)# snmp-server host ipv6 2001:DB8:89::13
```

**Syntax: [no] snmp-server host ipv6** *ipv6-address*

The *ipv6-address* must be in hexadecimal format using 16-bit values between colons, as documented in RFC 2373.

# Restricting Telnet access by specifying an IPv6 ACL

You can specify an IPv6 ACL to restrict Telnet access to management functions on the device. Enter commands similar to the following.

```
(config)# ipv6 access-list acl1
(config-ipv6-access-list acl1)# deny ipv6 host 2000:2382::e0bb:2 any
(config-ipv6-access-list acl1)# deny ipv6 2001:DB8::ff89/128 any
(config-ipv6-access-list acl1)# permit ipv6 any any
(config-ipv6-access-list acl1)# exit
(config)# telnet access-group ipv6 acl1
```

This example configures and applies an IPv6 ACL named "acl1", which denies Telnet access to the device from the specified IPv6 addresses, but allows access from any other IPv6 address.

```
(config)# ipv6 access-list acl2
(config-ipv6-access-list acl2)# permit ipv6 host 2000:2382::e0bb:2 any
(config-ipv6-access-list acl2)# deny ipv6 any any
(config-ipv6-access-list acl2)# exit
```

This example configures and applies an IPv6 ACL named "acl2", which allows Telnet access to the device only from the specified IPv6 address, and denies access from any other IPv6 address.

**Syntax: telnet access-group ipv6** *ipv6-acl-name*

The *ipv6-acl-name* is a valid IPv6 ACL.

# Restricting SSH access by specifying an IPv6 ACL

You can configure an IPv6 ACL to restrict SSH access to management functions on the device. Enter commands such as the following.

```
(config)# ipv6 access-list acl1
(config-ipv6-access-list acl1)# deny ipv6 host 2000:2382::e0bb:2 any
(config-ipv6-access-list acl1)# deny ipv6 2001:DB8::ff89/128 any
(config-ipv6-access-list acl1)# permit ipv6 any any
(config-ipv6-access-list acl1)# exit
(config)# ssh access-group ipv6 acl1
```

This example configures and applies an IPv6 ACL named "acl1", which denies SSH access to the device from the specified IPv6 addresses, but allows access from any other IPv6 address.

```
(config)# ipv6 access-list acl2
(config-ipv6-access-list acl2)# permit ipv6 host 2000:2382::e0bb:2 any
(config-ipv6-access-list acl2)# deny ipv6 any any
(config-ipv6-access-list acl2)# exit
(config)# ssh access-group ipv6 acl2
```

This example configures and applies an IPv6 ACL named "acl2", which allows SSH access to the device only from the specified IPv6 address, and denies access from any other IPv6 address.

**Syntax: [no] ssh access-group ipv6** *ipv6-acl-name*

The *ipv6-acl-name* is a valid IPv6 ACL.

# Restricting Web management access by specifying an IPv6 ACL

You can configure an IPv6 ACL to restrict Web management access to management functions on the device. Enter commands such as the following.

```
(config)# ipv6 access-list acl1
(config-ipv6-access-list acl1)# deny ipv6 host 2000:2382::e0bb:2 any
(config-ipv6-access-list acl1)# deny ipv6 2001:DB8::ff89/128 any
(config-ipv6-access-list acl1)# permit ipv6 any any
(config-ipv6-access-list acl1)# exit
(config)# web access-group ipv6 acl1
```

This example configures and applies an IPv6 ACL named "acl1", which denies Web management access to the device from the specified IPv6 addresses, but allows access from any other IPv6 address.

```
(config)# ipv6 access-list acl2
(config-ipv6-access-list acl2)# permit ipv6 host 2000:2382::e0bb:2 any
(config-ipv6-access-list acl2)# deny ipv6 any any
(config-ipv6-access-list acl2)# exit
```

This example configures and applies an IPv6 ACL named "acl2", which allows Web management access to the device only from the specified IPv6 address, and denies access from any other IPv6 address.

**Syntax: web access-group ipv6** *ipv6-acl-name*

The *ipv6-acl-name* variable is a valid IPv6 ACL.

# Restricting SNMP access by specifying an IPv6 ACL

You can configure an IPv6 ACL to restrict Web management access to management functions on the device.

NOTE
The syntax for configuring ACLs for SNMP access differs from the syntax for controlling Telnet, SSH, and Web management access using ACLs.

```
device(config)# ipv6 access-list aclro
device(config-ipv6-access-list aclro)# deny ipv6 host 2000:2382::e0bb:2 any
device(config-ipv6-access-list aclro)# deny ipv6 2001:DB8::ff89/128 any
device(config-ipv6-access-list aclro)# permit ipv6 any any
device(config-ipv6-access-list aclro)# exit
device(config)# ipv6 access-list aclrw
device(config-ipv6-access-list aclrw)# permit ipv6 host 2000:2382::e0bb:2 any
device(config-ipv6-access-list aclrw)# deny ipv6 any any
device(config-ipv6-access-list aclrw)# exit
device(config)# snmp-server community public ro ipv6 aclro
device(config)# snmp-server community private rw ipv6 aclrw
device(config)# write memory
```

These commands configure IPv6 ACLs *aclro* and *aclrw*, then apply these ACLs to community strings. ACL *aclro* controls read-only access using the "public" community string. ACL *aclrw* controls read-write access using the "private" community string.

**Syntax: [no] snmp-server community** *string* { **ro** | **rw** } **ipv6** *ipv6-acl-name*

The *string* specifies the SNMP community string you must enter for SNMP access.

The **ro** parameter indicates that the community string is for read-only ("get") access. The **rw** parameter indicates the community string is for read-write ("set") access.

The **ipv6** parameter indicates that you are applying an IPv6 access list.

The *ipv6-acl-name* variable specifies the IPv6 access list name.

NOTE
When **snmp-server community** is configured, all incoming SNMP packets are validated first by their community strings and then by their bound ACLs. Packets are permitted if no filters are configured for an ACL.

# Restricting Web management access to your device to a specific IPv6 host

You can restrict Web management access to your device to a specific IPv6 host only. Enter commands such as the following.

```
device(config)# web client ipv6 2001:db8:e0bb::2
```

**Syntax: [no] web client ipv6** *ipv6-address*

The *ipv6-address* must be in hexadecimal format using 16-bit values between colons, as documented in RFC 2373.

# Specifying an IPv6 Syslog server

To specify an IPv6 Syslog server, enter a command such as the following.

```
device(config)# log host ipv6 2001:db8:e0bb::4
```

**Syntax: [no] log host ipv6 ipv6-address** [ **udp-port-num** ]

The *ipv6-address* must be in hexadecimal using 16-bit values between colons, as documented in RFC 2373.

The *udp-port-num* optional parameter specifies the UDP application port used for the Syslog facility.

## Viewing IPv6 SNMP server addresses

Many **show** commands display IPv6 addresses for IPv6 SNMP servers. This example shows output for the **show snmp server** command.

```
device# show snmp server

      Contact:
     Location:
Community(ro): .....

Traps
              Warm/Cold start: Enable
                      Link up: Enable
                    Link down: Enable
                Authentication: Enable
     Locked address violation: Enable
         Power supply failure: Enable
                  Fan failure: Enable
           Temperature warning: Enable
                 STP new root: Enable
           STP topology change: Enable
                         vsrp: Enable

 Total Trap-Receiver Entries: 4

Trap-Receiver IP-Address                Port-Number Community

     1        10.147.201.100
               162     .....

     2        2001:db8:4000::200
            162     .....

     3        10.147.202.100
               162     .....

     4        2001:db8:3000::200
            162     .....
```

## Disabling router advertisement and solicitation messages

Router advertisement and solicitation messages enable a device to discover other devices on the same link. By default, router advertisement and solicitation message generation is enabled. To disable this feature, configure an IPv6 access list that denies them. Enter commands such as the following.

```
device(config)# ipv6 access-list rtradvert
device(config-ipv6-access-list rtradvert)# deny icmp any any router-advertisement
device(config-ipv6-access-list rtradvert)# deny icmp any any router-solicitation
device(config-ipv6-access-list rtradvert)# permit ipv6 any any
```

# IPv6 Non stop routing and graceful restart

At times, routers may need to restart or may undergo failover. Traditionally during a restart or failover, sessions with the restarting devices are tore down and re-established. Traffic is disrupted due to route deletion and addition in the forwarding plane. Graceful Restart (GR) and Non Stop Routing (NSR) are two different mechanisms to prevent routing protocol re-convergence during a processor switchover.

When Graceful Restart is used, peer networking devices are informed, via protocol extensions that the router is undergoing a restart condition. Peer devices, known as "helper" devices, will continue to forward to the restarting router until a "grace period", within which the adjacency is re-established.

When Non Stop Routing is used, peer networking devices have no knowledge of any event on the router that is switching over. All information needed to continue the routing protocol peering state is transferred to the standby processor so it can continue immediately upon a switchover. Since NSR does not require the help of neighboring routers during restart, NSR capable routers can be deployed independently in an existing network.

# Limitations

- Configuration events that occur at the same time as the switchover may get lost are lost due to the CLI synchronization.
- Neighbor, interface, or NSSA translation state changes 'close' to and during the switchover will not be handled.
  - Due to the core-reset of the LP, dead-timers below 40 seconds are not supported.
  - Number of neighbors supported may be limited depending on how many packets LP can send upon completion of the core-reset, due to competition with LP-sync-updates to get OSPF neighbor packets sent out.
- Traffic counters will not be synced. Neighbor and LSA DB counters will be recalculated on Standby during sync.
- There may be a slowdown of LSA acking due to the wait for the ack from Standby before acking the received LSAs.
- OSPF Database Overflow condition for External LSAs - depending on the sequence of redistribution or new LSAs (from neighbors), the LSAs accepted within the limits of the database may change upon switchover.
- The NSR hitless failover event may not be completely transparent to the network as after switchover additional flooding related protocol traffic will be generated to the directly connected neighbors.
- OSPF Startup Timers will not be applied upon NSR switchover.

# Supported protocols

The following protocols support both failover and Hitless Operating system Switchover (HLOS) for each protocol.

**TABLE 18** IPv6 Supported protocols for non-stop routing and graceful restart

| Protocol | Mechanism |
|---|---|
| OSPFv3 | Non-stop routing, Graceful restart helper |
| IS-IS IPv6 | Non-stop routing |
| BGP IPv6 | Graceful restart |

# Restart global timers

Restart contains two global timers, **max-hold-timer** and the **protocols-converge-timer,** that:

- Limit the amount of time used for re-syncing routes between the backup Management module and Interface modules (LPs) within the same chassis
- Allow a buffer time for protocols to converge and solve dependencies among each other

If the protocol-based restart features are configured when a Management module (MP) performs a switchover to its backup, routes are maintained on the LPs through the protocol-based restart processes for a specified period of time while the new MP learns the network routes. Once the MP learns all of its routes, the routes from the MP are synced with the routes on the LPs.

## *Graceful-restart IPv6 max-hold-timer*

The **graceful-restart ipv6 max-hold-timer** command defines the time that a device waits before sync up forwarding information is sent to the LP.

Use the **graceful-restart ipv6 max-hold-timer** command to set the max-hold-timer value.

```
device(config)# graceful-restart ipv6 max-hold-timer 300
```

**Syntax: [no] graceful-restart ipv6 max-hold-timer** *hold-interval*

The *hold-time* variable is the maximum hold time in seconds before sync up forwarding information is sent to the LP. The acceptable range is 30 to 3600 seconds. The default is 300 seconds.

### *Graceful-restart IPv6 protocols-converge-timer*

The **graceful-restart ipv6 protocols-converge-timer** command defines the time that a device waits for restarting protocols to converge at the final step in the restart process. In a heavily loaded system where BGP/OSPF/GRE/Static protocols can have a dependency on each other, their restart procedures may also depend on each other. This timer allows protocols to solve inter-dependencies after individual restart processes and before routing modules sync up new forwarding information to the interface module. The default value of 5 seconds will work in most cases, but if a system is heavily loaded and has protocols that depend on each other, it is recommended to increase this value.

Use the **graceful-restart ipv6 protocols-converge-timer** command to set the timer value.

```
device(config)# graceful-restart ipv6 protocols-converge-timer 20
```

**Syntax: [no] graceful-restart ipv6 protocols-converge-timer convergence-interval**

The *hold-time* variable is the maximum hold time in seconds before management routing modules sync up new forwarding information to interface modules during restart. The range of permissible values is 0 to 1200 seconds. The default value is 5 seconds.

## Configuring NSR and graceful restart on OSPFv3

OSPFv3 supports nonstop routing and graceful-restart helper mode. Nonstop routing and graceful-restart helper mode can be configured both in legacy router mode or VRF mode. The following commands are used to configure NSR and graceful-restart helper mode.

```
device(config-ospf6-router)#nononstop-routing
```

**Syntax: [no] nonstop-routing**

The**nonstop routing** command enables nonstop routing in OSPFv3.

NSR OSPFv3 is only supported on MLX Series and XMR Series devices. Graceful restart helper mode is supported on MLX Series and XMR Series devices and CER 2000 Series and CES 2000 Series devices.

Use the**graceful-restart helper** command to configure or disable helper mode.

```
device(config-ospf6-router)#graceful-restart helper
```

**Syntax: [no] graceful-restart helper** [ **disable** | **strict-lsa-checking** ]

The **graceful-restart helper disable** command disables the graceful-restart helper capability. By default it is enabled.

The**strict-las-checking** command exits helper mode upon a change in topology during a graceful restart.

### *Show commands*

## Show running-configuration

This command shows the running configuration.

```
device#show running-config
...
ip router-id 10.1.1.1
!
ipv6 router ospf
 area 0
 nonstop-routing
!
...
!
ipv6 router ospf vrf red
 graceful-restart helper strict-lsa-checking
!
...
ipv6 router ospf vrf blue
 area 0
 graceful-restart helper disable
!
```

**Syntax: show running-config**

## Show ipv6 ospf

This command shows the IPv6 OSPF configuration.

```
device#show ipv6 ospf
OSPFv3 Process number 0 with Router ID 0x10010101(10.1.1.1)
 Running 0 days 3 hours 11 minutes 42 seconds
 Number of AS scoped LSAs is 9
 Sum of AS scoped LSAs Checksum is 00006cc6
 External LSA Limit is 250000
 Route calculation executed 1 times
 Pending outgoing LSA count 0
 Authentication key rollover interval 300 seconds
 Number of areas in this router is 1
 High Priority Message Queue Full count: 0
 BFD is disabled
 Graceful restart helper is enabled, strict lsa checking is disabled
 Nonstop Routing is enabled
```

**Syntax: show ipv6 ospf**

## Show ipv6 ospf vrf *vrf name*

This command shows the IPv6 OSPF configuration on a specific VRF.

```
device#show ipv6 ospf vrf red
 OSPFv3 Process number 0 with Router ID 0x10020202(10.2.2.2)
 Running 0 days 8 hours 32 minutes 14 seconds
 Number of AS scoped LSAs is 4
 Sum of AS scoped LSAs Checksum is 00007d93
 External LSA Limit is 250000
 Route calculation executed 1 times
 Pending outgoing LSA count 0
 Authentication key rollover interval 300 seconds
 Number of areas in this router is 1
 High Priority Message Queue Full count: 0
 BFD is disabled
 Graceful restart helper is enabled, strict lsa checking is enabled
 Nonstop Routing is disabled
device#show ipv6 ospf vrf blue
 OSPFv3 Process number 0 with Router ID 0x10020202(10.2.2.2)
 Running 0 days 8 hours 32 minutes 14 seconds
```

```
 Number of AS scoped LSAs is 4
 Sum of AS scoped LSAs Checksum is 00007d93
 External LSA Limit is 250000
 Route calculation executed 1 times
 Pending outgoing LSA count 0
 Authentication key rollover interval 300 seconds
 Number of areas in this router is 1
 High Priority Message Queue Full count: 0
 BFD is disabled
 Graceful restart helper is disabled, strict lsa checking is disabled
 Nonstop Routing is disabled
```

**Syntax: show ipv6 ospf vrf** *vrfname*

## Show ipv6 ospf database

```
This command shows the IPv6 OSPF database configuration.
device#show ipv6 ospf database
 LSA Key - Rtr:Router Net:Network Inap:InterPrefix Inar:InterRouter
          Extn:ASExternal Grp:GroupMembership Typ7:Type7 Link:Link
          Iap:IntraPrefix Grc:Grace
Area ID         Type LSID       Adv Rtr         Seq(Hex) Age  Cksum Len   Sync
0               Rtr  0          10.1.1.1         800004cb 264  e06e  40    Yes
0               Iap  0          10.1.1.1         800004dc 264  9de4  52    Yes
0               Grc  1          10.2.2.2         80000001 17   a8a6  32    Yes
```

**Syntax: show ipv6 ospf database**

## Show ipv6 ospf data summary

```
This command displays the IPv6 OSPF data summary.
device(config-ospf6-router)#show ipv6 ospf data summary
 AS scope:
                      Active     MaxAge
 ASExternal                0          0
 Area 0 scope:
                      Active     MaxAge
 Router                    1          0
 Network                   0          0
 InterPrefix               0          0
 InterRouter               0          0
 Type7                     0          0
 IntraPrefix               1          0
 Other                     0          0
 Total                     0          0
  Interface scope (over 1 interfaces):
                       Active     MaxAge
  Link                      0          0
  Grace                     1          0
  Other                     0          0
  Total                     1          0
 Total: 3 LSAs, 3 Active LSAs, 0 MaxAge LSAs
```

**Syntax: show ipv6 ospf data summary**

## Show ipv6 ospf database grace

```
This command shows the IPv6 OSPF LSA timer grace period configuration.
device#show ipv6 ospf database grace
  LSA Key - Rtr:Router Net:Network Inap:InterPrefix Inar:InterRouter
          Extn:ASExternal Grp:GroupMembership Typ7:Type7 Link:Link
          Iap:IntraPrefix Grc: Grace
 Area ID         Type LSID       Adv Rtr         Seq(Hex) Age  Cksum Len
0               Grc  1          10.4.4.4         80000001 17   a8a6  36
    Restart duration: 150
    Restart Reason: Software Reload
```

**Syntax: show ipv6 ospf database grace**

# Configuring Non Stop Routing on IS-IS

NOTE
IPv6 IS-IS NSR is not supported on the CES 2000 Series and CER 2000 Series platforms.

IS-IS IPv6 supports nonstop routing. The following command is used to configure NSR. Further configuration details are available in Chapter 54.

```
device(config-isis-router)#nononstop-routing
```

**Syntax: [no] nonstop-routing**

The**nonstop routing** command enables nonstop routing in IS-IS IPv6.

## *Show commands*

### Show isis

This command shows the IS-IS configuration.

```
device#show isis
IS-IS Routing Protocol Operation State: Enabled
 IS-Type: Level-1-2
 ...
 Global Hello Padding For Point to Point Circuits: Enabled
 Ptpt Three Way HandShake Mechanism: Enabled
 BGP Ipv4 Converged: FALSE, Ipv6 Converged: FALSE
 IS-IS Traffic Engineering Support: Disabled
 No ISIS Shortcuts Configured
 BFD: Disabled
 NSR: Enabled
    NSR State: Normal
    Standby MP: Active
    Sync State: Enabled
 Interfaces with IPv4 IS-IS configured:
    None
 ...
```

# Configuring BGP graceful restart

BGP IPv6 supports graceful restart.

- BGP informs Graceful Restart capability to its peer.
- BGP peers retains BGP routing information and help Graceful Restart process.

The following command is used to configure graceful restart.

```
device(config-bgp-router)#graceful-restart
```

**Syntax: [no] graceful-restart** [ **purge-time** ] [ **restart-time** ] [ **stale-routes-time** ]

The**graceful-restart** command enables graceful restart for the address-family. The **purge-time** command is used to configure the maximum time in seconds before stale routes are purged. The **purge-time** cannot be less than the time set for the **stale-routes-time** .

The **restart-time** command is used to configure the maximum restart time advertised to neighbors in seconds. The**stale-routes-time** command is used to configure the maximum wait time in seconds for BGP EOR marker.

## *Show commands*

### Show running-configuration

```
This command shows the running configuration.
device#show running-config
Current BGP configuration:
router bgp
 local-as 200
 neighbor 2001:DB8:22::6 remote-as 100

 address-family ipv4 unicast
 graceful-restart stale-routes-time 100
 graceful-restart purge-time 100
 graceful-restart
 no neighbor 2001:DB8:22::6 activate
 exit-address-family

 address-family ipv4 multicast
 exit-address-family

 address-family ipv6 unicast
 graceful-restart restart-time 160
 graceful-restart stale-routes-time 120
 graceful-restart purge-time 120
 graceful-restart
 neighbor 2001:DB8:22::6 activate
 exit-address-family

 address-family ipv6 multicast
 exit-address-family

 address-family l2vpn vpls
 exit-address-family
end of BGP configuration
```

### Show ipv6 bgp neighbors *IP address*

```
This command shows the running configuration.
device#show ipv6 bgp neighbors 2001:DB8:22::6
1   IP Address: 2001:DB8:22::6, AS: 100 (EBGP), RouterID: 10.6.6.6, VRF: default-vrf
    State: ESTABLISHED, Time: 2h24m36s, KeepAliveTime: 60, HoldTime: 180
       KeepAliveTimer Expire in 16 seconds, HoldTimer Expire in 142 seconds
    Minimal Route Advertisement Interval: 0 seconds
       RefreshCapability: Received
       GracefulRestartCapability: Sent
           Restart Time 160 sec, Restart bit 0
           afi/safi 2/1, Forwarding bit 0
    Messages:    Open    Update  KeepAlive Notification Refresh-Req
       Sent    : 1       1       164       0            0
       Received: 1       0       164       0            0
    Last Update Time: NLRI       Withdraw          NLRI       Withdraw
                 Tx: ---         ---          Rx: ---        ---
    Last Connection Reset Reason:Unknown
    Notification Sent:     Unspecified
    Notification Received: Unspecified
    Neighbor NLRI Negotiation:
      Peer Negotiated IPV6 unicast  capability
      Peer configured for IPV6 unicast  Routes
    Neighbor ipv6 MPLS Label Capability Negotiation:
    Neighbor AS4 Capability Negotiation:
    Outbound Policy Group:
       ID: 2, Use Count: 1
    BFD:Disabled
    TCP Connection state: ESTABLISHED, flags:00000044 (0,0)
     Maximum segment size: 1440
```

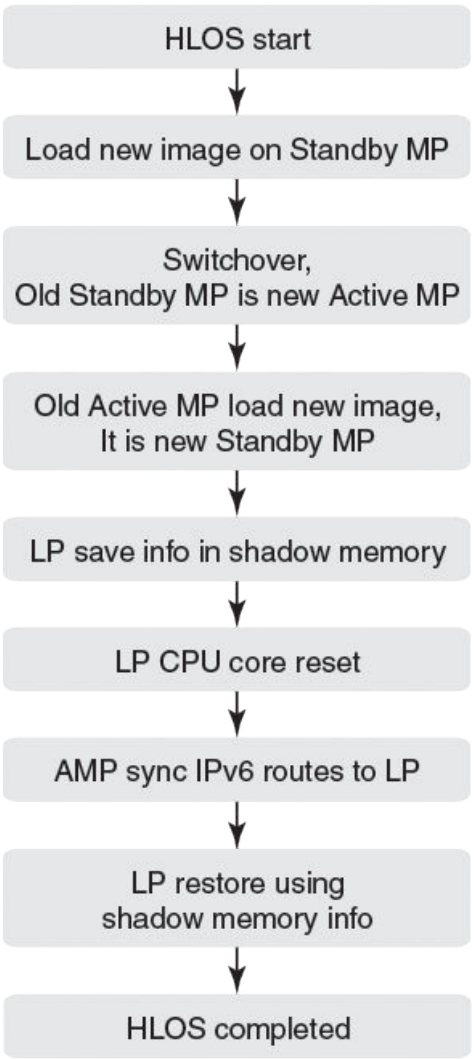
**Syntax: Show ipv6 bgp neighbors** *IPaddress*

# IPv6 Hitless OS upgrade

OSPFv3, IS-IS IPv6, and BGP IPv6 support both failover and Hitless Operating System Switchover (HLOS). HLOS provides a platform support mechanism to upgrade image without disrupting routing and forwarding service.

The process of syncing routes between a new MP and its LPs using the new timers are illustrated in Figure 16 and described in the following steps.

**FIGURE 16** IPv6 HLOS operation




1. HLOS starts and the Standby MP is rebooted with a new image.
2. System switches over and the Standby MP takes role of the Active MP.
3. The old Active MP is rebooted with the new image and it takes the role of the Standby MP.
4. Once the Active and Standby MP are in sync, the LP backs up the necessary IPv6 route information.

5. The LP CPU core resets, once the core reset is complete the LP receives IPv6 route information from Active MP.
6. The LP restores the complete IPv6 routes using the information synced from the Active MP to the LP and the backed up information on the LP.
7. HLOS complete.

# Configuring IPv4 and IPv6 protocol stacks

If a device is deployed as an endpoint for an IPv6 over IPv4 tunnel, you must configure the device to support IPv4 and IPv6 protocol stacks. Each interface that sends and receives IPv4 and IPv6 traffic must be configured with an IPv4 address and an IPv6 address. You can also explicitly enable IPv6 using the **ipv6 enable** command. Refer to Configuring a link-local IPv6 address on page 142.)

To configure an interface to support both IPv4 and IPv6 protocol stacks, enter commands such as the following.

```
device(config)# ipv6 unicast-routing
device(config)# interface ethernet 3/1
device(config-if-e100-3/1)# ip address 192.168.1.1 255.255.255.0
device(config-if-e100-3/1)# ipv6 address 2001:DB8:12d:1300::/64 eui-64
```

These commands globally enable IPv6 routing on the device, and configure an IPv4 address and an IPv6 address for Ethernet interface 3/1.

**Syntax: [no] ipv6 unicast-routing**

To disable IPv6 traffic globally on the Extreme device, enter the **no** form of this command.

**Syntax: [no] ip address** *ip-address sub-net-mask* [ **secondary** ]

You must specify the *ip-address* parameter using 8-bit values in dotted decimal notation.

You can specify the *sub-net-mask* parameter in either dotted decimal notation or as a decimal value preceded by a slash mark (/).

The **secondary** keyword specifies that the configured address is a secondary IPv4 address.

To remove the IPv4 address from the interface, enter the **no** form of this command.

**Syntax: [no] ipv6 address** *ipv6-prefix/prefix-length* [ **eui-64** ]

This syntax specifies a global or unique local IPv6 unicast address. For information about configuring a link-local IPv6 address, refer to Configuring a link-local IPv6 address on page 142.

You must specify the *ipv6-prefix* parameter in hexadecimal using 16-bit values between colons as documented in RFC 2373.

You must specify the *prefix-length* parameter as a decimal value. A slash mark (/) must follow the *ipv6-prefix* parameter and precede the *prefix-length* parameter.

The **eui-64** keyword configures the global or unique local unicast address with an EUI-64 interface ID in the low-order 64 bits. The interface ID is automatically constructed in IEEE EUI-64 format using the MAC address of the interface. If you do not specify the **eui-64** keyword, you must manually configure the 64-bit interface ID as well as the 64-bit network prefix. For more information about manually configuring an interface ID, refer to Configuring a global or unique local IPv6 unicast address on page 141.

# IPv6 Over IPv4 tunnels in hardware

To enable communication between the isolated IPv6 domains using the IPv4 infrastructure, you can configure IPv6 over IPv4 tunnels.

NOTE
The CES 2000 Series and CER 2000 Series currently do not support IPv6 over IPv4 tunneling.

NetIron OS devices support the following IPv6 over IPv4 tunneling in hardware mechanisms:

- Manually configured tunnels
- Automatic 6to4 tunnels

In general, a manually configured tunnel establishes a permanent link between routers in IPv6 domains, while the automatic tunnels establish a transient link that is created and taken down on an as-needed basis. (Although the feature name and description may imply otherwise, some configuration is necessary to set up an automatic tunnel.) Also, a manually configured tunnel has explicitly configured IPv4 addresses for the tunnel source and destination, while the automatic tunnels have an explicitly configured IPv4 address for the tunnel source and an automatically generated address for the tunnel destination.

These tunneling mechanisms require that the router at each end of the tunnel run both IPv4 and IPv6 protocol stacks. The routers running both protocol stacks, or dual-stack routers, can interoperate directly with both IPv4 and IPv6 end systems and routers.

The following features are not supported for IPv6 tunnel configuration at this time:

- Keep-alive
- Hitless upgrade
- Tunnels over MPLS or GRE

# Configuring a IPv6 IP tunnel

To configure a IPv6 IP Tunnel, configure the following parameters:

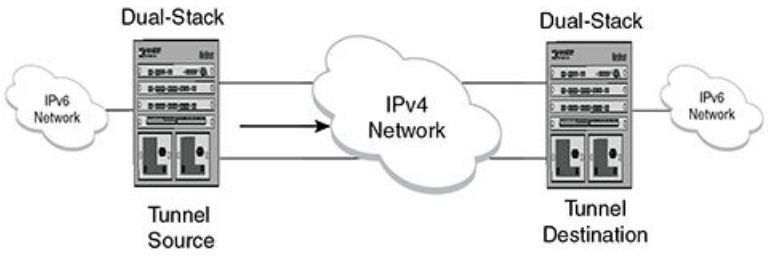
- CAM Restrictions
- Maximum Number of Tunnels (optional)
- Tunnel Interface
- Source Address or Source Interface for the Tunnel
- Destination address for the Tunnel
- IPv6 Encapsulation
- IP address for the Tunnel
- TTL Value (optional)
- TOS Value (optional)
- MTU Value (optional)

NOTE
Do not forward packets from one type of tunnel to another type of tunnel in XPP. Packets may not be routed properly.

# Configuring a manual IPv6 tunnel

You can use a manually configured tunnel to connect two isolated IPv6 domains. You should deploy this point-to-point tunnel mechanism if you need a permanent and stable connection.

**FIGURE 17** Manually configured tunnel




### *Configuration notes on manual tunnels:*

- The tunnel mode should be **ipv6ip** indicating that this is an IPv6 manual tunnel.
- Both source and destination addresses needs to be configured on the tunnel.
- On the remote side you need to have exactly opposite source/destination pair.
- The tunnel destination should be reachable through the IPv4 backbone.
- The ipv6 address on the tunnel needs to be configured for the tunnel to come up.
- The tunnel source can be an IP address or interface name.
- Manual tunnels provide static point-point connectivity.
- Static routing on top of the tunnel is supported.
- IPv6 routing protocols including OSPFv3 and RIPing on top of the tunnel are supported.

  NOTE
  IPv6 IS-IS is not supported on top of the tunnel.

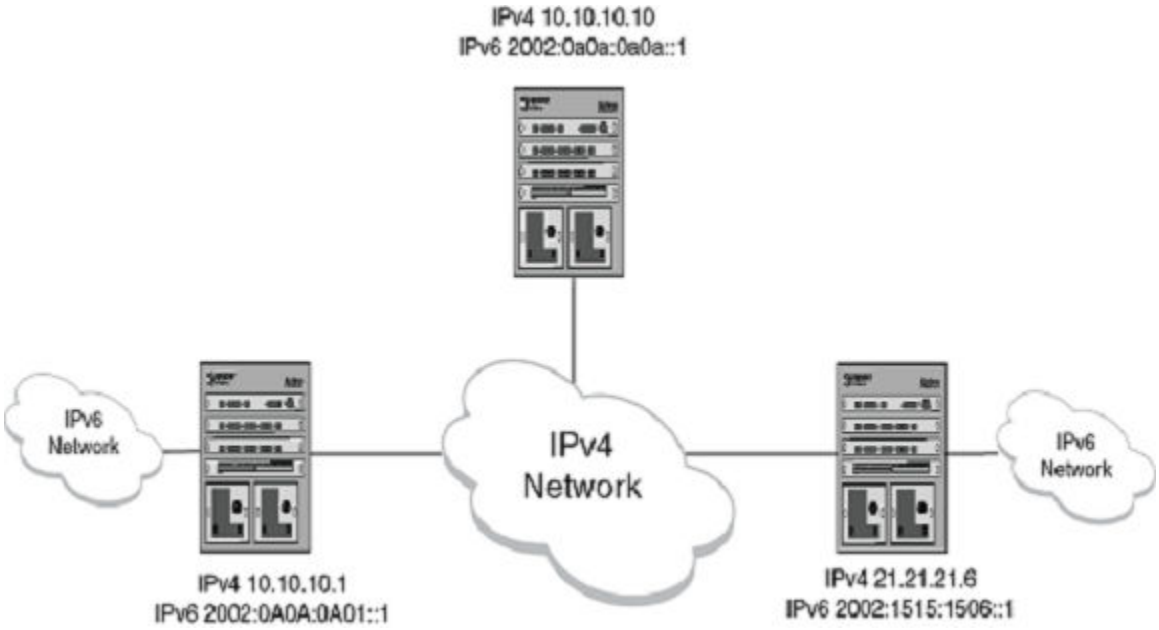
## Configuring an automatic 6to4 tunnel

An automatic 6to4 tunnel establishes a transient link between IPv6 domains, which are connected by an IPv4 backbone. When needed, a device on which an automatic 6to4 tunnel is configured in one domain can establish a tunnel with another similarly configured device in another domain. When no longer needed, the devices take down the tunnel.

Instead of a manually configured tunnel destination, an automatic 6to4 tunnel constructs a globally unique 6to4 prefix, which determines the tunnel destination. The 6to4 prefix has the following format:

2002:*ipv4-address* ::/48

When two domains need to communicate, a device creates a tunnel using the 6to4 prefix. The software automatically generates the 6to4 prefix by concatenating a configured static IPv6 prefix of 2002 with the destination device's globally unique IPv4 address. (Each device in an IPv6 domain that needs to communicate over an automatic 6to4 tunnel must have one globally unique IPv4 address, from which the globally unique 6to4 prefix is constructed.) After the communication ends, the tunnel is taken down.




## *Configuration notes on 6to4tunnels:*

- This tunnel treats the IPv4 infrastructure as a virtual non-broadcast link and support multipoint connectivity.
- Tunnel mode must be configured as **ipv6ip 6to4.**
- Tunnel source must be configured.
- Tunnel destination is not configured on 6to4 tunnel explicitly, as the destination is specified as part of static nexthop or BGP nexthop.
- Static route with **2002::/16** MUST be configured.
- IPv6 address with **2002:A.B.C.D::/48** must be configured for the tunnel to come up (A.B.C.D is the tunnel source IP address).
- You can have 6to4 tunnel with multiple nexthops depending on the IPv6 nexthop used to forward the packets.
- With 6to4 tunnels, you can only use routing protocols (that is BGP+) that specify the nexthop in the configuration.
- OSPFv3, IPv6 IS-IS and RIPng are not supported on the 6to4 tunnels.
- Static routes can be used with 6to4 tunnels. If you use a static route to configure the nexthop, you MUST enable nexthop recursion in the system (ipv6 route next-hop-recursion).
- The 6to4 tunnel tries to resolve all the nexthops and programs the cam and pram entries needed. The IPv4 address in the nexthop should be reachable through the IPv4 network.

In the below configuration:

- – **10.10.10.1** is the tunnel source IP address
  – **10.10.10.10** is the static nexthop
  – **21.21.21.6** is I-BGP nexthop
  – **22.22.22.6** is E-BGP nexthop

## *Static route Nexthop example:*

- Create a static route pointing to the tunnel.

```
device(config) #ipv6 route 2002::/16 tunnel 2 // Mandatory for 6to4 Configuration
device(config) #ipv6 route next-hop-recursion // Mandatory with static nexthop
device(config)# ipv6 route 3001::/64 2002:0a0a:0a0a::1 // Static Nexthop: 10.10.10.10
```

- Create a Source Interface - The remote node needs to have a similar route pointing to this node.

```
device(config)# interface ethernet 1/1
device(config-if-e10000-1 /1)ip address 10.10.10.1 255.255.255.0
```

- Create a 6to4 Tunnel configuration.

```
device(config) interface tunnel 2
device(config-tnif-2) tunnel mode ipv6ip 6to4
device(config-tnif-1) tunnel source 10.10.10.1
device(config-tnif-1) ipv6 address 2002:0a0a:0a01::1/64
```

: I-BGP Nexthop.

```
device(config) router bgp
device(config-bgp) local-as 100
device(config-bgp) neighbor 2002:1515:1506::1 remote-as 100 // BGP Nexthop: 21.21.21.6
device(config-bgp)# address-family ipv4 unicast
device(config-bgp)# no neighbor 2001:DB8:1506::1 activate
device(config-bgp)# exit-address-family
device(config-bgp)# address-family ipv4 multicast
device(config-bgp)# exit-address-family
device(config-bgp)# address-family ipv6 unicast
device(config-bgp)# neighbor 2002:1515:1506::1 activate
device(config-bgp)# exit-address-family
```

: E-BGP Nexthop.

```
device(config)# router bgp
device(config-bgp)# local-as 100
device(config-bgp)# neighbor 2002:1616:1606::1 remote-as 101 // BGP Nexthop: 22.22.22.6
device(config-bgp)# neighbor 2002:1616:1606::1 ebgp-multihop
device(config-bgp)# address-family ipv4 unicast
device(config-bgp)# no neighbor 2002:1515:1506::1 activate
device(config-bgp)# exit-address-family
device(config-bgp)# address-family ipv4 multicast
device(config-bgp)# exit-address-family
device(config-bgp)# address-family ipv6 unicast
device(config-bgp)# neighbor 2002:1616:1606::1 activate
device(config-bgp)# exit-address-family
```

## *Configuring the maximum number of tunnels supported*

You can configure the device to support a specified number of tunnels using the following command.

```
device(config)# system-max ip-tunnels 512
device(config)# write memory
```

**Syntax: [no] system-max ip-tunnels number**

The *number* variable specifies the number of IPv6 tunnels that can be supported on the Extreme device. The permissible range is 1 - 512. The default value is 256.

> **NOTE**
> You must write this command to memory and perform a system reload for this command to take effect.

## Configuring a tunnel interface

To configure a tunnel interface, use a the following command.

```
device(config)# interface tunnel 1
device(config-tnif-1)
```

**Syntax: [no] interface tunnel tunnel id**

The *tunnel-id* variable is numerical value that identifies the tunnel being configured.Possible range is from 1 to the maximum configured tunnels in the system.

## Configuring a source address or source interface for a tunnel interface

To configure a source address for a specific tunnel interface, enter the following command.

```
device(config)# interface tunnel 1
device(config-tnif-1)tunnel source 10.0.8.108
```

To configure a source interface for a specific tunnel interface, enter the following command.

```
device(config)# interface tunnel 100
device(config-tnif-100)tunnel source ethernet 3/1
```

**Syntax: [no] tunnel source ip-address | port-no**

You can specify either of the following:

The *ip-address* variable is the source IP address being configured for the specified tunnel.The *port-no* variable is the source slot/port of the interface being configured for the specified tunnel. When you configure a source interface, there must be at least one IP address configured on that interface. Otherwise, the interface will not be added to the tunnel configuration and an error message like the following will be displayed: "Error - Tunnel source interface 3/1 has no configured ip address.

It can be a physical or virtual interface (ve).

## Configuring a destination address for a tunnel interface

To configure a destination address for a specific tunnel interface, enter the following command.

```
device(config)# interface tunnel 1
device(config-tnif-1)tunnel destination 10.108.5.2
```

**Syntax: [no] tunnel destination ip-address**

The *ip-address* variable is destination IP address being configured for the specified tunnel.

## Configuring a tunnel interface for IPv6 encapsulation

To configure a specified tunnel interface for IPv6 encapsulation, enter the following command.

```
device(config)# interface tunnel 1
device(config-tnif-1)tunnel mode ipv6ip
```

**Syntax: [no] tunnel mode ipv6ip 6to4 | auto-tunnel**

The **6to4** parameter specifies automatic tunneling using 6 to 4.

The **auto-tunnel** parameter specifies automatic tunnel using IPv4 compatible ipv6 address.

## Configuring an IP address for a tunnel interface

To configure an IP address for a specified tunnel interface, enter the following command.

```
device(config)# interface tunnel 1
device(config-tnif-1)ipv6 address 2001:0a0a:0a01::1/64
```

**Syntax: [no] ipv6 address ipv6-address**

The *ipv6-address* variable is the IPv6 address being configured for the specified tunnel interface.

## Configuring a TTL value

This is an optional parameter that allows you to set the Time-to-Live value for the outer IP header of the IPv6 tunnel packets.

To configure the TTL value, enter the following command.

```
device(config)# interface tunnel 1
device(config-tnif-1)tunnel ttl 100
```

**Syntax: [no] tunnel ttl ttl-value**

The *ttl-value* variable specifies a TTL value for the outer IP header. Possible values are 1 - 255. The default value is 255.

## Configuring a TOS value

This is an optional parameter that allows you to set the TOS value for the outer IP header of the GRE tunnel packets.

To configure the TOS value, enter the following command.

```
device(config)# interface tunnel 1
device(config-tnif-1)tunnel tos 100
```

**Syntax: [no] tunnel ttl tos-value**

The *tos-value* variable specifies a TOS value for the outer IP header. Possible values are 1 - 255. The default value is 0.

## Configuring IPv6 session enforce check

You can enable the IPv6 session enforce check by using the **ipv6-session-enforce-check** command. When an IPv6 packet arrives and this feature is enabled, the system tries to match the IPv6 packet source and destination address pair with the tunnel configured destination and source pair. If the pairs do not match, the packet is dropped in hardware.

To configure the IPv6 session enforce check, go to the IP tunnel policy context and enter the **ipv6-session-enforce-check** command.

```
device(config)#ip-tunnel-policy
device(config-ip-tunnel-policy)#ipv6-session-enforce-check
```

**Syntax: [no] ipv6-session-enforce-check**

To disable the IPv6 session enforce check, use the **no** form of this command. This command is disabled by default. You might have to write the configuration to memory and reload the system when the configuration of this command is changed because a one-time creation of a source-ingress CAM partition is necessary. The system prompts you if the memory write and reload are required.

The first-time execution of certain commands necessitates the creation of a source-ingress CAM partition, after which you write to memory and reload. These commands are **gre-session-enforce-check** , **ipv6-session-enforce-check** , and **accounting-enable** . After this CAM partition is created, it is not necessary to follow either of the other two commands with a memory write and reload.

NOTE

The ipv6-sessions-enforce-check is not supported for 6to4 automatic tunnels.

### *Configuring a maximum MTU value for a tunnel interface*

This command allows you to set an MTU value for packets entering the tunnel. Packets that exceed either the default MTU value of 1480 bytes or the value that you set using this command are sent back to the source.

The following command allows you to change the MTU value for packets transiting "tunnel 1".

```
device(config)# interface tunnel 1
device(config-tnif-1)tunnel mtu 1500
```

**Syntax: [no] tunnel mtu packet-size**

The *packet-size* variable specifies the maximum MTU size in bytes for the packets transiting the tunnel.

NOTE
To prevent packet loss after the 20 byte IP header is added, make sure that any physical interface that is carrying IPv6 tunnel traffic has an IP MTU setting at least 24 bytes greater than the tunnel MTU setting.

## Bypassing ACLs in an IPv6-over-IPv4 tunnel

Use this procedure to disable IPv6 ACLs on the terminating node of an IPv6-over-IPv4 tunnel for internal traffic coming over the tunnel.

NOTE
Disabling ACL processing on an IPv6-over-IPv4 tunnel also disables support for the following features for internal traffic on that tunnel:

- All features employing IPv6 ACLs
- BFD over MPLS
- Multicast
- PBR
- OpenFlow

1. Access global configuration mode.

   ```
   device# configure terminal
   ```

2. Access ACL global policy configuration mode.

   ```
   device(config)# acl-policy
   ```

3. Enter the **disable-acl-for-6to4** command.

   ```
   device(config-acl-policy)# disable-acl-for-6to4
   ```

## Displaying IPv6 tunneling information

You can display IPv6 Tunneling Information using the **show ip-tunnels** , **show ipv6 interface** , **show ipv6 route** and **show interface tunnel** commands as shown in the following:

## Displaying tunnel information

For example, to tunnel information for tunnel 2, enter the following command at any level of the CLI.

```
device# show ip-tunnels 2
IPv6 tnnl 2 UP   : src_ip 10.211.2.1, dst_ip 10.212.2.1
   TTL 255, TOS 0, NHT 0, MTU 1480
```

**Syntax: show ip tunnels number**

The *number* parameter indicates the tunnel interface number for which you want to display information.

This display shows the following information.

**TABLE 19** Show IP tunnel display information

| This field... | Displays... |
|---|---|
| IPv6 tnnl *UP* \| *DOWN* | The status of the tunnel interface can be one of the following:<br>• up - The tunnel interface is functioning properly.<br>• down - The tunnel interface is not functioning and is down. |
| src_ip | The tunnel source can an IPv4 address. |
| dst_ip | The tunnel destination can an IPv4 address. |
| TTL | The TTL value configured for the outer IP header. Possible values are 1 - 255. |
| TOS | The TOS value configured for the outer IP header. Possible values are 1 - 255. |
| NHT | The nextHop Table index value. |
| MTU | The setting of the IPv6 maximum transmission unit (MTU). |

## Displaying tunnel interface information

For example, to display status and configuration information for tunnel interface 1, enter the following command at any level of the CLI.

```
device# show interfaces tunnel 1
Tunnel1 is up, line protocol is up
  Hardware is Tunnel
  Tunnel source ethernet 3/5
  Tunnel destination is not configured
  Tunnel mode ipv6ip auto-tunnel
  No port name
  MTU 1500 bytes
```

**Syntax: show interfaces tunnel number**

The *number* parameter indicates the tunnel interface number for which you want to display information.

This display shows the following information.

**TABLE 20** IPv6 tunnel interface information

| This field... | Displays... |
|---|---|
| Tunnel interface status | The status of the tunnel interface can be one of the following:<br>• up - The tunnel interface is functioning properly.<br>• down - The tunnel interface is not functioning and is down. |
| Line protocol status | The status of the line protocol can be one of the following:<br>• up - The line protocol is functioning properly.<br>• down - The line protocol is not functioning and is down. |
| Hardware is tunnel | The interface is a tunnel interface. |

**TABLE 20** IPv6 tunnel interface information (continued)

| This field... | Displays... |
|---|---|
| Tunnel source | The tunnel source can be one of the following:<br>• An IPv4 address<br>• The IPv4 address associated with an interface or port. |
| Tunnel destination | The tunnel destination can an IPv4 address. |
| Tunnel mode | The tunnel mode can be one the following:<br>• ipv6ip auto-tunnel - Indicates an automatic IPv4-compatible tunnel.<br>• ipv6ip 6to4 - Indicates an automatic 6to4 tunnel. |
| Port name | The port name configured for the tunnel interface. |
| MTU | The setting of the IPv6 maximum transmission unit (MTU). |

## *Displaying interface level IPv6 settings*

To display Interface level IPv6 settings for tunnel interface 1, enter the following command at any level of the CLI.

```
device# show ipv6 inter tunnel 1
Interface Tunnel 1 is up, line protocol is up
  IPv6 is enabled, link-local address is fe80::3:4:2 [Preferred]
  Global unicast address(es):
    1001::1 [Preferred],  subnet is 1001::/64
    1011::1 [Preferred],  subnet is 1011::/64
  Joined group address(es):
    ff02::1:ff04:2
    ff02::5
    ff02::1:ff00:1
    ff02::2
    ff02::1
  MTU is 1480 bytes
  ICMP redirects are enabled
  No Inbound Access List Set
  No Outbound Access List Set
  OSPF enabled
```

The display command above reflects the following configuration.

```
device# show running-config interface tunnel 1
!
interface tunnel 1
 port-name ManualTunnel1
 tunnel mode ipv6ip
 tunnel source loopback 1
 tunnel destination 10.1.1.1
 ipv6 address fe80::3:4:2 link-local
 ipv6 address 1011::1/64
 ipv6 address 1001::1/64
 ipv6 ospf area 0
```

# IPv6 over IPv4 GRE tunnel

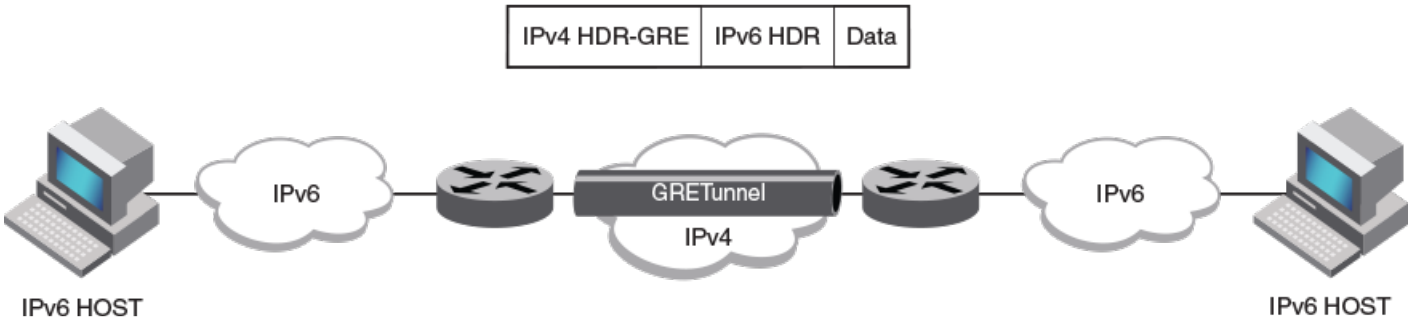
IPv6 data packets can be transported across an IPv4 network that does not support IPv6 using GRE tunnels.

GRE provides a way to encapsulate packets inside of a transport protocol and transmit them from one tunnel endpoint to another. To allow communication between isolated IPv6 networks across an existing IPv4 network, a GRE tunnel can be configured between end devices that have dual stack (IPv4/IPv6) support. Incoming packets from an IPv6 network are encapsulated within a GRE header and are transported across an IPv4 network to a host in another IPv6 network. This feature allows remote IPv6 network traffic to be transported without upgrading the IPv4 network over which the traffic is encapsulated.

NOTE
The number of IPv6 GRE tunnels supported is 512.

The following diagram shows a GRE tunnel over an IPv4 network with remote IPv6 hosts and some packet heading fields to note that an IPv6 header is included before the data.

**FIGURE 18** IPv6 over an IPv4 GRE tunnel network




The following IPv4 GRE tunnel configuration options are supported:

- Tunnel MTU
- Time-to-Live (TTL) value
- Type of Service (ToS)
- Keepalive
- VRF—The usual VRF limitations apply.
- GRE session enforce check—When the **gre-session-enforce-check** command is entered, and a GRE packet arrives at the device, the software tries to match the GRE packet source and destination address pair with the tunnel configured destination and source pair. If the pairs do not match, the packet is dropped.
- System max—Configures the devices to support a specified number of tunnels. Requires writing to memory and a system reload.
- Accounting—To start collecting the statistics for GRE and IPv6 tunnels use the **accounting-enable** command in IP tunnel policy configuration.

The following limitations apply to IPv6 over IPv4 GRE tunnels:

- IPv6 fragmentation is not supported before encapsulation.
- GRE header encapsulation is limited to 4 bytes.
- Multicast is not supported for IPv6 over IPv4 GRE tunnels.

NOTE
The IPv6 over IPv4 GRE Tunnels feature is only supported on MLXe and XMR devices. Not all interface cards on these devices are supported. See the Feature Support Matrix for more details.

# Configuring a GRE tunnel for IPv6 traffic

To allow IPv6 traffic to be transported across an IPv4 network, a GRE tunnel can be employed.

Use this task when there are isolated IPv6 networks that need to communicate across an IPv4-only network. the endpoints of the tunnel must support both IPv4 and IPv6. This tunnel source address on one device must be one of the device IPv4 addresses that is configured on a physical, loopback, or VE interface, through which the other end of the tunnel is reachable.

1. From privileged EXEC mode, enter global configuration mode.

```
device# configure terminal
```

2. Create a tunnel interface.

```
device(config)# interface tunnel 3
```

3. Assign a source IPv4 address for the tunnel.

```
device(config-tnif-3)# tunnel source 10.1.1.1
```

This source address will be entered as the destination address specified for the device on the other end of the tunnel.

4. Assign a destination IPv4 address for the tunnel.

```
device(config-tnif-3)# tunnel destination 10.1.1.2
```

This destination address should be the address of the IPv4 interface of the device on the other end of the tunnel.

5. Enable GRE encapsulation on the tunnel interface.

```
device(config-tnif-3)# tunnel mode gre ip
```

6. Specify the IPv6 address of the tunnel interface.

```
device(config-tnif-3)# ipv6 address 2005:a0a:a01::1/64
```

7. Change the MTU value for packets transiting the tunnel.

```
device(config-tnif-3)# ip mtu 1400
```

This step is optional.

8. Enable GRE link keepalive

```
device(config-tnif-3)# keepalive 12 4
```

In this example, the device waits for 4 consecutive lost keepalive packets before bringing the tunnel down. There will be a 12 second interval between each packet. This step is optional.

9. Exit to global configuration mode.

```
device(config-tnif-3)# exit
```

10. If a route to the tunnel destination does not already exist, create a static route and specify that the route is through the tunnel interface.

```
device(config)# ipv6 route 2002:a0a:a01::1/64 2005:a0a:a01::1
```

    In this example, an IPv6 static route is created on this device to specify that network 2002:a0a:a01::1/64 is reachable via tunnel address 2005:a0a:a01::1/64. There is an IPv4 network between the GRE tunnel endpoints.

On the device at one end of the tunnel, the following sample configures a GRE tunnel for IPv6 traffic.

```
device# configure terminal
device(config)# interface ethernet 4/1
device(config-int-e10000-4/1)# ip address 10.1.1.1/24
device(config-int-e10000-4/1)# exit

device(config)# interface tunnel 3
device(config-tnif-3)# tunnel source 10.1.1.1
device(config-tnif-3)# tunnel destination 10.1.1.2
device(config-tnif-3)# tunnel mode gre ip
device(config-tnif-3)# ipv6 address 2005:a0a:a01::1/64
device(config-tnif-3)# exit
device(config)# ipv6 route 2002:a0a:a01:1/64 2005:a0a:a01::2
device(config)# exit
device>
```

The following sample configures the device at the second end of the tunnel.

```
device# configure terminal
device(config)# interface ethernet 6/1
device(config-int-e10000-6/1)# ip address 10.1.1.2/24
device(config-int-e10000-6/1)# exit

device(config)# interface tunnel 3
device(config-tnif-3)# tunnel source 10.1.1.2
device(config-tnif-3)# tunnel destination 10.1.1.1
device(config-tnif-3)# tunnel mode gre ip
device(config-tnif-3)# ipv6 address 2005:a0a:a01::2/64
device(config-tnif-3)# exit
device(config)# ipv6 route 2001:a0a:a01:1/64 2005:a0a:a01::1
device(config)# exit
device>
```

# Verifying IPv6 over GRE Tunnel

The configuration of IPv6 over an IPv4 GRE tunnel can be verified using various show commands.

The following commands can be entered in any order.

NOTE

When reviewing the keepalive packet statistics in the output of the **show interface tunnel** command for a GRE tunnel, note that the transmitted keepalive packets are hardware generated and are not counted in the "Xmit-to-tnnl" and "Rcv-from-tnnl" statistics.

1. To view IPv6 GRE tunnel information.

```
device(config)# show ipv6 interface tunnel 3

Interface Gre_tnnl 3 is up, line protocol is up
  IPv6 is enabled, link-local address is fe80::224:38ff:fea4:1a00 [Preferred]
  Global unicast address(es):
    2005:a0a:a01::2 [Preferred],  subnet is 2005:a0a:a01::/64
    2005:a0a:a01:: [Anycast],  subnet is 2005:a0a:a01::/64
  Joined group address(es):
    ff02::1:ff00:2
    ff02::1:ff00:0
    ff02::1:ffa4:1a00
    ff02::2
    ff02::1
 Local Proxy disabled
  Port belongs to VRF: default-vrf
  MTU is 1400 bytes
  ICMP redirects are disabled
  No Inbound Access List Set
  No Outbound Access List Set
```

2. To view IPv4 information about the configured GRE tunnel, use the following command.

```
device(config)# show interface tunnel 3

Tunnel3 is up, line protocol is up
  Hardware is Tunnel
  Tunnel source  10.1.1.2
  Tunnel destination is 10.1.1.1
  Tunnel mode gre ip
  Configured BW is 0 kbps
  No port name
  Global unicast address(es):
    2005:a0a:a01::2,  subnet is 2005:a0a:a01::/64
  Tunnel TOS 0, Tunnel TTL 255, Tunnel MTU 1400 bytes
  Keepalive is Enabled
  VRF Forwarding: default-vrf
```

3. To view details of the IPv6 route where the GRE tunnels are shown under the Interface field.

```
device(config)# show ipv6 route

IPv6 Routing Table - 2 entries:
Type Codes - B:BGP C:Connected I:ISIS L:Local O:OSPF R:RIP S:Static
     BGP  Codes - i:iBGP e:eBGP
ISIS Codes - L1:Level-1 L2:Level-2
OSPF Codes - i:Inter Area 1:External Type 1 2:External Type 2
STATIC Codes - d:DHCPv6
   Type  IPv6 Prefix        Next Hop Router   Interface  Dis/Metric  Uptime src-vrf
1  S     2001:a0a:a01::/64  2005:a0a:a01::1   gre_tnl 2  1/1         0m7s   -
2  C     2005:a0a:a01::/64  ::                gre_tnl 2  0/0         2m59s  -
```

# Configuring IPv6 Domain Name Server (DNS) resolver

The Domain Name Server (DNS) resolver feature lets you use a host name to perform Telnet, ping, and traceroute commands. You can also define a DNS domain on a device to recognize all hosts within that domain. After you define a domain name, the device automatically appends the appropriate domain to the host and forwards it to the domain name server.

For example, if the domain "example.com" is defined on a device, and you want to initiate a ping to host "EXC01" on that domain, you only need to reference the host name instead of the host name and the domain name. For example, enter either of the following commands to initiate the ping.

```
device# ping exc01
device# ping exc01.example.com
```

## Defining a DNS entry

You can define up to four DNS servers for each DNS entry. The first entry serves as the primary default address. If a query to the primary address is not resolved after three attempts, the next gateway address is queried (up to three times). This process continues for each defined gateway address until the query is resolved. The order in which the default gateway addresses are polled is the same as the order in which you enter them.

To define the domain name *example.com* on a device and then define four possible default DNS gateway addresses, using IPv4 addressing, enter the following commands.

```
device(config)# ip dns domain-name example.com
device(config)# ip dns server-address 10.157.22.199 10.96.7.15 10.95.7.25 10.98.7.15
```

**Syntax: [no] ip dns server-address** *ip-addr* [ *ip-addr* ] [ *ip-addr* ] [ *ip-addr* ]

In this example, the first IP address in the command becomes the primary gateway address and all others are secondary addresses. Because IP address 10.98.7.15 is the last address listed, it is also the last address consulted to resolve a query.

### *Defining an IPv6 DNS entry*

IPv6 defines new DNS record types to resolve queries for domain names to IPv6 addresses, as well as IPv6 addresses to domain names. Devices running IPv6 software support AAAA DNS records, which are defined in RFC 1886.

AAAA DNS records are analogous to the A DNS records used with IPv4. A complete IPv6 address is stored in each record. AAAA records have a type value of 28.

To establish an IPv6 DNS entry for the device, enter the following command.

```
device(config)# ipv6 dns domain-name example.com
```

**Syntax: [no] ipv6 dns domain-name** *domainname*

To define an IPv6 DNS server address, enter the following command.

```
device(config)# ipv6 dns server-address 2001:DB8::1
```

**Syntax: [no] ipv6 dns server-address** *ipv6-addr* [ *ipv6-addr* ] [ *ipv6-addr* ] [ *ipv6-addr* ]

For example, in a configuration where *ftp6.example.com* is a server with an IPv6 protocol stack, when a user pings *ftp6.example.com* , the device attempts to resolve the AAAA DNS record. In addition, if the DNS server does not have an IPv6 address, as long as it is able to resolve AAAA records, it can still respond to DNS queries.

### *DNS queries of IPv4 and IPv6 DNS servers*

IPv4 and IPv6 DNS record queries search through IPv4 and IPv6 DNS servers are described here.

**For IPv4 DNS record queries:**

- Loop through all configured IPv4 DNS servers.
- If no IPv4 DNS servers are configured, then loop through all configured IPv6 DNS servers (if any).

**For IPv6 DNS record queries:**

- Loop through all configured IPv6 DNS servers.
- If no IPv6 DNS servers are configured, then loop through all configured IPv4 DNS servers (if any).

# IPv6 Non-Stop Routing support

When IPv6 Non-Stop-Routing (NSR) is used, peer networking devices do not have knowledge of any event on the switching over router. All information needed to continue the routing protocol peering state is transferred to the standby processor so it can pick up immediately upon a switchover. As NSR does not need the help of neighboring routers during restart, the NSR-capable routers can be deployed independently in an existing network.

This section describes support for IPv6 Non-Stop-Routing (NSR) on MLX Series and XMR Series devices. The scope of this section is for IPv6 unicast routing only.

## Limitations

- Configuration events that occur closer to switchover may get lost due to CLI synchronization issues.
- Neighbor, interface, or NSSA translation state changes that occur close to and during the switchover will not be handled.
- Counters - Traffic counters will not be synchronized. Neighbor and LSA DB counters will be recalculated on Standby during sync and thus not synchronized.
- OSPF Database Overflow condition for External LSAs - depending on the sequence of redistribution or new LSAs (from neighbors) the LSAs accepted within the limits of the database may change upon switchover.
- The NSR hitless failover event may not be completely transparent to the network as after switchover additional flooding related protocol traffic will be generated to the directly connected neighbors.
- OSPF Startup Timers - will not be applied upon NSR switchover.

## Configuring IPv6 NSR support

Use the following commands to configure IPv6 Non-Stop Routing support.

The **graceful-restart ipv6 max-hold-timer** sets the hold interval.

```
device(config)#graceful-restart ipv6 max-hold-timer 100
```

**Syntax: [no] graceful-restart ipv6 max-hold-timer** *hold-interval*

The acceptable range for the maximum hold time before sync up forwarding information is 30 to 3600 seconds. The default is 300 seconds.

The **graceful-restart ipv6 protocols-converge-timer** sets the convergence interval. The default setting is 5 seconds.

```
device(config)#no graceful-restart ipv6 protocol-convergence-timer 50
```

**Syntax: [no] graceful-restart ipv6 protocols-converge-timer** *convergence-interval*

The acceptable range for the maximum time for protocols to converge after a graceful restart is 0 to 1200 seconds. The default protocol convergence time is 5seconds.

# ECMP load sharing for IPv6

IPv6 ECMP load sharing is hardware-managed. If there is more than one path to a given destination, a hash is calculated based on the source MAC address, destination MAC address, source IPv6 address, destination IPv6 address, and TCP/UDP source port and destination port (if the packet is also a TCP and UDP packet). This hash is used to select one of the paths.

## Disabling or re-enabling ECMP load sharing for IPv6

ECMP load sharing for IPv6 is enabled by default. To disable the feature, enter the following command.

```
device(config)# no ipv6 load-sharing
```

To re-enable the feature after disabling it, enter the following command.

```
device(config)# ipv6 load-sharing 4
```

**Syntax: [no] ipv6 load-sharing** *number*

The *number* parameter specifies the number of ECMP load sharing paths. Enter a value between 2 and 32 for *number* to set the maximum number of paths. The default value is 4.

NOTE
The maximum number of paths supported by the BR-MLX-10Gx24-DM module is 16.

## Changing the maximum number of load sharing paths for IPv6

By default, IPv6 ECMP load sharing balances traffic across up to four equal paths. You can change the maximum number of paths to a value between 2 and 32.

To change the number of ECMP load sharing paths for IPv6, enter the following command:

```
device(config)# ipv6 load-sharing 8
```

**Syntax: [no] ipv6 load-sharing** *num* **ber**

The *number* parameter specifies the number of ECMP load sharing paths. Enter a value between 2 and 32 for *number* to set the maximum number of paths. The default value is 4.

NOTE
The maximum number of paths supported by the BR-MLX-10Gx24-DM module is 16.

# Configuring IPv6 ICMP

ICMP for IPv6 provides error and informational messages. The stateless auto-configuration, neighbor discovery, and path MTU discovery features use ICMP messages.

This section explains how to configure the following IPv6 ICMP options:

- ICMP rate limiting
- ICMP redirects
- ICMP unreachable address or route messages
- ICMP error messages for source-routed IPv6 packets
- ICMP error messages for an unreachable address
- ICMP messages for an unreachable route
- ICMP error messages for IPv6 packets with hop-limit 0
- ICMP error messages for CER 2000 Series and CES 2000 Series devices

# Configuring ICMP rate limiting

You can limit the rate at which IPv6 ICMP error messages are sent out on a network. For this rate-limiting implementation, IPv6 ICMP uses a token bucket algorithm.

The algorithm works using a *virtual bucket* that contains a number of tokens, where each token represents the ability to send one ICMP error message. Tokens are placed in the bucket at a specified interval until the maximum allowed number of tokens is reached. For each error message ICMP sends, a token is removed from the bucket. ICMP generates a series of error messages until the bucket is empty. When the bucket is empty, further error messages cannot be sent until a new token is placed in the bucket.

You can adjust the following elements related to the token bucket algorithm:

- The interval at which tokens are added to the bucket. The default is 100 milliseconds.
- The maximum number of tokens in the bucket. The default is 10 tokens.

For example, to adjust the interval to 1000 milliseconds and the number of tokens to 100 tokens, enter the following command.

```
device(config)# ipv6 icmp error-interval 1000 100
```

**Syntax: [no] ipv6 icmp error-interval** *interval* [ *number-of-tokens* ]

The interval at which tokens are placed in the bucket has a range of 0 - 2147483647 milliseconds.

> NOTE
> If you keep the default interval (100 milliseconds), output from the **show run** command does not show the setting of the **ipv6 icmp error-interval** command. In addition, if you configure the interval value to a number that does not evenly divide into 100000 (100 milliseconds), the system rounds the value up to the next higher value that does divide evenly. For example, if you specify an interval value of 150, the system rounds it to 200.

ICMP rate limiting is enabled by default. To disable ICMP rate limiting, set the interval to 0.

# Enabling ICMP redirect messages

To enable ICMP redirect messages, you need to configure icmp redirect at both global level and the interface level. You can enable or disable a device to transmit ICMP redirect messages from a global level and the interface level.

To enable the ICMP redirect messages from global level, enter the following commands.

**Syntax: [no] ipv6 icmp redirects**

By default, IPv6 redirect is disabled and the device does not send an ICMP redirect message to a neighboring host to inform it of a better first-hop device on a path to a destination. (For more information about how ICMP redirect messages are implemented for IPv6, refer to Configuring IPv6 neighbor discovery on page 176.)

To enable the sending of ICMP redirect messages on interface 3/1, enter the following commands.

```
device(config)# interface ethernet 3/1
device(config-if-e100-3/1)# ipv6 redirects
```

To disable the ICMP redirect messages from Ethernet interface 3/1, enter the following commands.

```
device(config)# interface ethernet 3/1
device(config-if-e100-3/1)# no ipv6 redirects
```

**Syntax: [no] ipv6 redirects**

Use the **show ipv6 interface***interface port-number* command to verify that the sending of ICMP redirect messages is enabled on a particular interface.

# Disabling or re-enabling ICMP redirect messages

You can disable or re-enable a device to transmit ICMP redirect messages from an interface. By default, a device sends an ICMP redirect message to a neighboring host to inform it of a better first-hop device on a path to a destination. No further configuration is required to enable the sending of ICMP redirect messages. (For more information about how ICMP redirect messages are implemented for IPv6, refer to Configuring IPv6 neighbor discovery on page 176.)

For example, to disable the ICMP redirect messages from Ethernet interface 3/1, enter the following commands.

```
device(config)# interface ethernet 3/1
device(config-if-e100-3/1)# no ipv6 redirects
```

**Syntax: [no] ipv6 redirects**

To re-enable the sending of ICMP redirect messages on Ethernet interface 3/1, enter the following commands.

```
device(config)# interface ethernet 3/1
device(config-if-e100-3/1)# ipv6 redirects
```

Use the **show ipv6 interface***interface port-number* command to verify that the sending of ICMP redirect messages is enabled on a particular interface.

# Disabling ICMP error messages for source-routed IPv6 packets

By default, ICMP error messages are transmitted to announce discarded IPv6 source-routed packets that were addressed to one of the IPv6 addresses of a device. By default, these packets are discarded in software, as described in Software filtering of IPv6 source-routed packets on page 185.

You can disable or re-enable the sending of ICMP error messages for discarded, IPv6 source-routed packets by using the **ipv6 icmp source-route** command. Use the **no** form of this command to disable the transmission of these error messages. The following example illustrates the disabling operation.

```
device(config)# no ipv6 icmp source-route
```

**Syntax: [no] ipv6 icmp source-route**

# Enabling ICMP error messages for an unreachable address

By default, the ICMPv6 destination unreachable messages with the code for an unreachable address are not sent for a discarded IPv6 packet. You can enable the sending of these messages by using the **ipv6 icmp unreachable address** command. This command applies globally.

For example, to enable ICMPv6 error messages for unreachable address on the current device, enter the following command.

```
device(config)# ipv6 icmp unreachable address
```

**Syntax: [no] ipv6 icmp unreachable address**

Use the **no** parameter in front of the **ipv6 icmp unreachable address** command to disable the sending of ICMPv6 destination unreachable messages with the code is address unreachable.

# Enabling ICMP messages for an unreachable route

By default, the ICMPv6 destination unreachable messages with the code for an unreachable route are not sent for a discarded IPv6 packet. You can enable the sending of these messages by using the **ipv6 icmp unreachable route** command.

For example, to enable ICMPv6 error messages for unreachable route on the current device, enter the following command.

```
device(config)# ipv6 icmp unreachable route
```

**Syntax: [no] ipv6 icmp unreachable route**

Use the **no** parameter in front of the **ipv6 icmp unreachable route** command to disable the sending of ICMPv6 destination unreachable messages with the code for destination unreachable.

# Enabling ICMP error messages for IPv6 packets with hop-limit 0

By default, an MLX Series and XMR Series series box does not respond to an IPv6 packet with hop-limit 0, and drops it at the hardware. You can enable or disable a device to respond to such packets with a proper ICMPv6 error message using the **ipv6 icmp hop-limit-zero** command from the global config mode.

> NOTE
> This command is available only on MLX Series and XMR Series routers.

For example, to enable ICMPv6 error messages for IPv6 routed packets with hop-limit 0, enter the following command:

```
device(config)#ipv6 icmp hop-limit-zero
```

**Syntax: [no] ipv6 icmp hop-limit-zero**

Use the **show running-configuration** command to see if this is enabled or disabled. Use the no parameter in front of the**ipv6 icmp hop-limit-zero** command to disable the sending of ICMP error messages for IPv6 Routed packet with hop-limit 0.

# Enabling ICMP error messages for multicast Too Big packets

By default, the device will not send an ICMPv6 Packet Too Big error message for the multicast packets. You can enable or disable (default behavior) a device to send the ICMPv6 Packet Too Big error messages for the IPv6 packets sent to multicast address destination using the **ipv6 icmp packet-too-big-for-multicast** command from the global config mode.

> NOTE
> This command is available on MLX Series, XMR Series, CES 2000 Series and CER 2000 Series devices.

For example, to enable a device to send the ICMPv6 Packet Too Big error messages for the IPv6 packets sent to multicast address destination, enter the following command:

```
device(config)#ipv6 icmp packet-too-big-for-multicast
```

**Syntax: [no] ipv6 icmp packet-too-big-for-multicast**

Use the **show running-configuration** command to see if this is enabled or disabled. Use the no parameter in front of the **ipv6 icmp packet-too-big-for-multicast** command to disable the sending of ICMPv6 Packet Too Big error message for the multicast packets.

## Enabling ICMP error messages for CES or CER 2000 Series devices

By default, the CES 2000 Series and CER 2000 Series devices do not generate error message for many of the ICMPv6 error cases. You can enable or disable a device to generate error message in all the error conditions using the **ipv6 icmp generate-error-message** command from the global config mode.

> NOTE
> Enabling this command enables all the IPv6 packets will be sent to the CPU. This command is available only on CES 2000 Series and CER 2000 Series devices.

For example, to enable a device to generate error message in all the error conditions, enter the following command:

```
device(config)#ipv6 icmp generate-error-message
```

**Syntax: [no] ipv6 icmp generate-error-message**

Use the **show running-configuration** command to see if this is enabled or disabled. Use the no parameter in front of the **ipv6 icmp generate-error-message** command to disable device to generate error message in all the error conditions.

# Configuring IPv6 neighbor discovery

The neighbor discovery feature for IPv6 uses IPv6 ICMP messages to do the following:

- Determine the link-layer address of a neighbor on the same link.
- Verify that a neighbor is reachable.
- Track neighbor devices.

An IPv6 host is required to listen for and recognize the following addresses, which identify this host:

- Link-local address.
- Assigned unicast address.
- Loopback address.
- All-nodes multicast address.
- Solicited-node multicast address.
- Multicast address to all other groups to which it belongs.

You can adjust the following IPv6 neighbor discovery features:

- Neighbor solicitation messages for duplicate address detection.
- Router advertisement messages:
  - Interval between router advertisement messages.
  - Value that indicates a device is advertised as a default device (for use by all nodes on a given link).
  - Prefixes advertised in router advertisement messages.
  - Flags for host stateful autoconfiguration.
- The time that an IPv6 node considers a remote node reachable (for use by all nodes on a given link).

The default maximum value for IPv6 neighbor discovery (ND) entries is 4096 for XMR Series and MLX Series devices.

The memory is allocated for IPv4 and IPv6 separately. The maximum IPv4 ARP and IPv6 ND entries can be supported together.

# Neighbor solicitation and advertisement messages

Neighbor solicitation and advertisement messages enable a node to determine the link-layer address of another node (neighbor) on the same link. (This function is similar to the function provided by the Address Resolution Protocol [ARP] in IPv4.) For example, node 1 on a link wants to determine the link-layer address of node 2 on the same link. To do so, node 1, the source node, multicasts a neighbor solicitation message. The neighbor solicitation message, which has a value of 135 in the Type field of the ICMP packet header, contains the following information:

- **Source address:** IPv6 address of node 1 interface that sends the message.
- **Destination address:** solicited-node multicast address (FF02:0:0:0:0:1:FF00::/104) that corresponds the IPv6 address of node 2.
- Link-layer address of node 1.
- A query for the link-layer address of node 2.

After receiving the neighbor solicitation message from node 1, node 2 replies by sending a neighbor advertisement message, which has a value of 136 in the Type field of the ICMP packet header. The neighbor solicitation message contains the following information:

- **Source address:** IPv6 address of the node 2 interface that sends the message.
- **Destination address:** IPv6 address of node 1.
- Link-layer address of node 2.

After node 1 receives the neighbor advertisement message from node 2, nodes 1 and 2 can now exchange packets on the link.

After the link-layer address of node 2 is determined, node 1 can send neighbor solicitation messages to node 2 to verify that it is reachable. Also, nodes 1, 2, or any other node on the same link can send a neighbor advertisement message to the all-nodes multicast address (FF02::1) if there is a change in their link-layer address.

# Router advertisement and solicitation messages

Router advertisement and solicitation messages enable a node on a link to discover the devices on the same link.

Each configured interface on a link sends out a router advertisement message, which has a value of 134 in the Type field of the ICMP packet header, periodically to the all-nodes link-local multicast address (FF02::1).

A configured interface can also send a router advertisement message in response to a router solicitation message from a node on the same link. This message is sent to the unicast IPv6 address of the node that sent the router solicitation message.

At system startup, a host on a link sends a router solicitation message to the all-routers multicast address (FF01). Sending a router solicitation message, which has a value of 133 in the Type field of the ICMP packet header, enables the host to automatically configure its IPv6 address immediately instead of awaiting the next periodic router advertisement message.

Because a host at system startup typically does not have a unicast IPv6 address, the source address in the router solicitation message is usually the unspecified IPv6 address (0:0:0:0:0:0:0:0). If the host has a unicast IPv6 address, the source address is the unicast IPv6 address of the host interface sending the router solicitation message.

Entering the **ipv6 unicast-routing** command automatically enables the sending of router advertisement messages on all configured interfaces. You can configure several router advertisement message parameters. For information about disabling router advertisement messages and the router advertisement parameters you can configure, refer to Configuring reachable time for remote IPv6 nodes on page 182 and Setting IPv6 router advertisement parameters on page 178.

# Neighbor redirect messages

After forwarding a packet, by default, a device can send a neighbor redirect message to a host to inform it of a better first-hop device. The host receiving the neighbor redirect message will then readdress the packet to the better device.

A device sends a neighbor redirect message only for unicast packets, only to the originating node, and to be processed by the node.

A neighbor redirect message has a value of 137 in the Type field of the ICMP packet header.

# Setting neighbor solicitation parameters for duplicate address detection

Although the stateless autoconfiguration feature assigns the 64-bit interface ID portion of an IPv6 address using the MAC address of the host's NIC, duplicate MAC addresses can occur. Therefore, the duplicate address detection feature verifies that a unicast IPv6 address is unique before it is assigned to a host interface by the stateless autoconfiguration feature. Duplicate address detection verifies that a unicast IPv6 address is unique.

If duplicate address detection identifies a duplicate unicast IPv6 address, the address is not used. If the duplicate address is the link-local address of the host interface, the interface stops processing IPv6 packets.

You can configure the following neighbor solicitation message parameters that affect duplicate address detection while it verifies that a tentative unicast IPv6 address is unique:

- The number of consecutive neighbor solicitation messages that duplicate address detection sends on an interface. By default, duplicate address detection sends three neighbor solicitation messages without any follow-up messages.
- The interval in seconds at which duplicate address detection sends a neighbor solicitation message on an interface. By default, duplicate address detection sends a neighbor solicitation message every 1 second.

  NOTE
  For the interval at which duplicate address detection sends a neighbor solicitation message on an interface, the device uses seconds as the unit of measure instead of milliseconds.

For example, to change the number of neighbor solicitation messages sent on Ethernet interface 3/1 to two and the interval between the transmission of the two messages to 9 seconds, enter the following commands.

```
device(config)# interface ethernet 3/1
device(config-if-e100-3/1)# ipv6 nd dad attempt 2
device(config-if-e100-3/1)# ipv6 nd ns-interval 9
```

**Syntax: [no] ipv6 nd dad attempt number**

**Syntax: [no] ipv6 nd ns-interval number**

For the number of neighbor solicitation messages, you can specify between 0-255 attempts. Configuring a value of 0 disables duplicate address detection processing on the specified interface. To restore the number of messages to the default value, use the **no** form of this command.

For the interval between neighbor solicitation messages, you can specify between 0 and 4294967 seconds. Not recommended for very short intervals in normal IPv6 operation. When a non-default value is configured, the configured time is both advertised and used by the device itself. To restore the default interval, use the **no** form of this command.

# Setting IPv6 router advertisement parameters

You can adjust the following parameters for router advertisement messages:

- The interval (in seconds) at which an interface sends router advertisement messages. By default, an interface sends a router advertisement message every 200 seconds.

- The "router lifetime" value, which is included in router advertisements sent from a particular interface. The value (in seconds) indicates if the device is advertised as a default device on this interface. If you set the value of this parameter to 0, the device is not advertised as a default device on an interface. If you set this parameter to a value that is not 0, the device is advertised as a default device on this interface. By default, the device lifetime value included in device advertisement messages sent from an interface is 1800 seconds.

When adjusting these parameter settings, it is recommended that the interval between device advertisement transmission be less than or equal to the device lifetime value if the device is advertised as a default device. For example, to adjust the interval of device advertisements to 300 seconds and the device lifetime value to 1900 seconds on Ethernet interface 3/1, enter the following commands.

```
device(config)# interface ethernet 3/1
device(config-if-e100-3/1)# ipv6 nd ra-interval 300
device(config-if-e100-3/1)# ipv6 nd ra-lifetime 1800
```

**Syntax: [no] ipv6 nd ra-interval number**

**Syntax: [no] ipv6 nd ra-lifetime number**

The *number* parameter in both commands indicates any numerical value.

Possible range value for **ipv6 nd ra-interval***number* **is 3 to 1800 seconds.**

Possible range value for **ipv6 nd ra-lifetime***number* **is 3 to 1800 seconds.**

To restore the default interval or device lifetime value, use the **no** form of the respective command.

# Controlling prefixes advertised in IPv6 router advertisement messages

By default, router advertisement messages include prefixes configured as addresses on interfaces using the **ipv6 address** command. You can use the **ipv6 nd prefix-advertisement** command to control exactly which prefixes are included in router advertisement messages. Along with which prefixes the router advertisement messages contain, you can also specify the following parameters:

- Valid lifetime -- (Mandatory) The time interval (in seconds) in which the specified prefix is advertised as valid. The default is 2592000 seconds (30 days). When the timer expires, the prefix is no longer considered to be valid.
- Preferred lifetime -- (Mandatory) The time interval (in seconds) in which the specified prefix is advertised as preferred. The default is 604800 seconds (7 days). When the timer expires, the prefix is no longer considered to be preferred.
- Onlink flag -- (Optional) If this flag is set, the specified prefix is assigned to the link upon which it is advertised. Nodes sending traffic to addresses that contain the specified prefix consider the destination to be reachable on the local link.
- Autoconfiguration flag -- (Optional) If this flag is set, the stateless auto configuration feature can use the specified prefix in the automatic configuration of 128-bit IPv6 addresses for hosts on the local link.

For example, to advertise the prefix 2001:DB8:a487:7365::/64 in router advertisement messages sent out on Ethernet interface 3/1 with a valid lifetime of 1000 seconds, a preferred lifetime of 800 seconds, and the Onlink and Autoconfig flags set, enter the following commands.

```
device(config)# interface ethernet 3/1
device(config-if-e100-3/1)# ipv6 nd prefix-advertisement 2001:DB8:a487:7365::/64 1000 800 onlink autoconfig
```

**Syntax: [no] ipv6 nd prefix-advertisement ipv6-prefix/prefix-length valid-lifetime preferred-lifetime** [ **autoconfig** ] [ **onlink** ]

You must specify the *ipv6-prefix* parameter in hexadecimal using 16-bit values between colons as documented in RFC 2373.

You must specify the *prefix-length* parameter as a decimal value. A slash mark (/) must follow the *ipv6-prefix* parameter and precede the *prefix-length* parameter.

The valid lifetime and preferred lifetime is a numerical value between 0 - 4294967295 seconds. The default valid lifetime is 2592000 seconds (30 days), while the default preferred lifetime is 604800 seconds (7 days).

To remove a prefix from the router advertisement messages sent from a particular interface, use the **no** form of this command.

# Configuring the Domain Name of DNS suffix

This section provides information about the IPv6 RA option that allows IPv6 routers to advertise domain names of DNS suffixes (the DNS name excluding the host) to IPv6 hosts in a local area network. This option to configure domain names is valid for any network that supports the use of ND6. The domain names that are advertised by routers are sent through RA messages to IPv6 hosts.

This option is supported only when IPv6 routing is active on the network. The newly configured domain name can be used as long as the RA router lifetime has not expired.

## *Configuration considerations*

- A maximum of 4 domain names and their corresponding lifetime-multiplier values can be configured at the global configuration level.
- A maximum of 4 domain names and their corresponding lifetime-multiplier values can be configured per interface.
- The domain name that is configured on the interface overrides all other domain name configurations at the system level for this interface.

By default, the domain name of the DNS suffix and the lifetime multiplier information is not configured. The following examples are used to configure the domain names of a DNS suffix for a lifetime-multiplier value of 200.

```
device(config)# ipv6 nd ra-domain-name extreme.com lifetime-multiplier 200
device(config-if-e10000-1/10)# ipv6 nd ra-domain-name extreme.com lifetime-multiplier 200
```

**Syntax: [no] ipv6 nd ra-domain-name string** [ **lifetime-multiplier decimal** ]

The *string* parameter specifies the domain name of the DNS suffix.

The **lifetime-multiplier is the percentage value of the maximum router advertisement interval. The maximum router advertisement interval is the maximum time that can be allowed between sending unsolicited RA messages for DNS name resolution for a DNS entry. The lifetime-multiplier value is calculated as twice the RA lifetime. The maximum router advertisement interval percentage range is 100 through 200%. The default value for maximum router advertisement interval is 200%.**

To disable the advertisement of the specified domain name of DNS suffix in the RA message, use the **no** form of the respective command.

# Configuring the recursive DNS server addresses and lifetime multiplier

This section provides information about the IPv6 RA attribute that allows IPv6 routers to advertise recursive DNS server addresses and lifetime multiplier values to IPv6 hosts in a local area network. This option to configure recursive DNS server addresses is valid for any network that supports the use of neighbor discovery (ND6). The recursive server addresses that are advertised by routers are sent through RA messages and are used to translate domain names to IP addresses.

This option is supported only when IPv6 routing is active on the network. The newly configured recursive DNS server address can be used as long as the RA router lifetime has not expired.

### *Configuration considerations*

- A maximum of 4 recursive DNS server addresses and their corresponding lifetime-multiplier values can be configured at the global configuration level.
- A maximum of 4 recursive DNS server addresses and their corresponding lifetime-multiplier values can be configured per interface.
- The recursive DNS server address that is configured on the interface overrides all other recursive DNS server configurations at the system level for this interface.

By default, the recursive DNS server address and the lifetime multiplier information is not configured. The following examples configure the recursive DNS address for a lifetime-multiplier value of 200.

```
device(config)# ipv6 nd ra-dns-server 2001:DC8:200::3 lifetime-multiplier 200
device(config-if-e100-3/1)# ipv6 nd ra-dns-server 2001:DC8:200::3 lifetime-multiplier 200
```

**Syntax: [no] ipv6 nd ra-dns-server ipv6-address** [ **lifetime-multiplier decimal** ]

The *ipv6-address* parameter specifies the global IPv6 address of the DNS server.

The **lifetime-multiplier is the percentage value of the maximum router advertisement interval. The maximum router advertisement interval is the maximum time that can be allowed between sending unsolicited RA messages for DNS name resolution for a DNS entry. The lifetime-multiplier decimal value is calculated as twice the RA lifetime. The percentage range is 100 through 200%. The default value for the maximum router advertisement interval is 200%.**

To disable the advertisement of the specified server address in the RA message, use the **no** form of the command.

## Setting flags in IPv6 router advertisement messages

An IPv6 router advertisement message can include the following flags:

- **Managed Address Configuration** -- This flag indicates to hosts on a local link if they should use the stateful autoconfiguration feature to get IPv6 addresses for their interfaces. If the flag is set, the hosts use stateful autoconfiguration to get addresses as well as non-IPv6-address information. If the flag is not set, the hosts do not use stateful autoconfiguration to get addresses and if the hosts can get non-IPv6-address information from stateful autoconfiguration is determined by the setting of the Other Stateful Configuration flag.
- **Other Stateful Configuration** -- This flag indicates to hosts on a local link if they can get non-IPv6 address autoconfiguration information. If the flag is set, the hosts can use stateful autoconfiguration to get non-IPv6-address information.

  NOTE
  When determining if hosts can use stateful autoconfiguration to get non-IPv6-address information, a set Managed Address Configuration flag overrides an unset Other Stateful Configuration flag. In this situation, the hosts can obtain non address information. However, if the Managed Address Configuration flag is not set and the Other Stateful Configuration flag is set, then the setting of the Other Stateful Configuration flag is used.

By default, the Managed Address Configuration and Other Stateful Configuration flags are not set in router advertisement messages. For example, to set these flags in router advertisement messages sent from Ethernet interface 3/1, enter the following commands.

```
device(config)# interface ethernet 3/1
device(config-if-e100-3/1)# ipv6 nd managed-config-flag
device(config-if-e100-3/1)# ipv6 nd other-config-flag
```

**Syntax: [no] ipv6 nd managed-config-flag**

**Syntax: [no] ipv6 nd other-config-flag**

To remove either flag from router advertisement messages sent on an interface, use the **no** form of the respective command.

## Configuring reachable time for remote IPv6 nodes

You can configure the duration (in seconds) that a device considers a remote IPv6 node reachable. By default, an interface uses the value of 30 seconds.

The router advertisement messages sent by an interface include the amount of time specified by the **ipv6 nd reachable-time** command so that nodes on a link use the same reachable time duration. By default, the messages include a default value of 0.

NOTE
The device uses seconds, instead of milliseconds, for the interval at which it sends router advertisement messages.

It is not recommended to configure a short reachable time duration, because a short duration causes the IPv6 network devices to process the information at a greater frequency.

For example, to configure the reachable time of 40 seconds for Ethernet interface 3/1, enter the following commands.

```
device(config)# interface ethernet 3/1
device(config-if-e100-3/1)# ipv6 nd reachable-time 40
```

**Syntax: [no] ipv6 nd reachable-time seconds**

For the *seconds* parameter, you can specify between 0-3600 seconds. To restore the default time, use the **no** form of this command.

# IPv6 ND Prefix Suppress

Extreme devices support IPv6 ND Prefix Suppress, which is useful in an LAN where multiple hosts are connected to router(s). Prefix Suppress performs these functions:

- – Advertisement of on-link prefix information is suppressed in router advertisement (RA) messages.
  – Hosts are prevented from auto configuring based on the prefix in the RA message.
  – DHCPv6 is used for security and accountability.
  – Advertisement of identical prefixes by multiple routers is suppressed.
  – Global Suppress option suppresses IPv6 addresses defined on an interface from getting advertised in the RA message.

  NOTE
  When the user configures the Global Suppress option, an RA is generated with all deprecated IPv6 address entries that are not advertised in subsequent RA messages.

- – Prefix advertisement entry in the RA message is advertised if a duplicate entry exists in the prefix advertisement list and IPv6 address list

Configuring the suppress option to specific IPv6 addresses defined on an interface generates deprecated IPv6 address entries (i.e. with `preferred lifetime = 0 hours` and `valid lifetime = 2 hours` ) in RA messages. When a host receives deprecated IPv6 address entries, the address is forbidden for new sessions although existing sessions can continue using the address.

Deprecated entries are advertised in the following scenarios:

- – Suppress option is configured for IPv6 address entries.
  – Prefix advertisement entries are un-configured.
  – IPv6 address entries are un-configured.

# Configuring IPv6 Prefix Suppress

Command syntax for configuring the suppress option for an IPv6 address entry:

```
(config-if-x)#[no] ipv6 nd address <ipv6-address> suppress
```

Command syntax for configuring the suppress option for all IPv6 address entries:

```
(config-if-x)#[no] ipv6 nd address suppress
```

Command syntax for Show IPv6 interface output is modified to display individual or globally suppressed entries:

```
Router-A# show ipv6 interface ethernet 1/2
  MTU is 1500 bytes
  ICMP redirects are disabled
  ND DAD is enabled, number of DAD attempts: 3
  ND reachable time is 30000 Milliseconds
  ND advertised reachable time is 0 seconds
  ND retransmit interval is 1000 milliseconds
  ND advertised retransmit interval is 0 milliseconds
  ND next router advertisement will be sent in 2 seconds
  ND router advertisements live for 1800 seconds
  ND suppress-ra disabled
  ND address-prefixes suppressed in router advertisement - all
  ND address-prefixes suppressed in router advertisement -
  300::1/64
Router-A#
```

Command syntax for debugging IPv6 Prefix Suppress:

```
Router-A# debug ipv6 ra
```

NOTE

No additional debug commands are added for this feature. Debug commands available for IPv6 ND can be used for this feature.

## *Configuration Considerations for IPv6 Prefix Suppress*

The following considerations should be considered prior to configuring IPv6 Prefix Suppress:

- – Suppress option is not configurable for a non-existent IPv6 address entry. As a result, the suppress option is not applicable to future references.
  – Suppress option is not supported for suppressing prefix advertisement entries.

  NOTE

  The user may un-configure the prefix advertisement entry so it is not advertised in the RA message.

- – Configuration of the suppress option is not allowed for a duplicate entry in any combination.
  – When multiple IPv6 addresses of the same subnet are defined on an interface, apply the suppress option on individual entries.

# IPv6 ND Router Advertisement Control

IPv6 ND Router Advertisement Control allows for disabling sending out router advertisements at the interface level. The **no ipv6 nd suppress-ra** command at the interface level allows the user to disable and enable the sending of the ND Router Advertisement on an interface. By default, the sending of ND Router Advertisement (RA) is enabled on all interfaces, except for the tunnel and loopback interfaces, providing that the IPv6 Unicast Routing is enabled and the interfaces are active for IPv6.

The IPv6 ND Router Advertisement Control gives the ability to quickly turn off the sending of IPv6 ND Router Advertisement message on an IPv6 enabled interfaces.

By default,

- The ND Router Advertisement is enabled.
- Interface is enabled to send ND Router Advertisements.
- The **ipv6 nd suppress-ra** and **ipv6 nd send-ra** interface commands, when configured, override the system and VRF global **ipv6 nd global-suppress-ra** command.

Users sometimes require the ability to quickly turn off the sending of IPv6 ND Router Advertisement message on an IPv6 enabled interfaces. This is achieved by providing the following additional configuration command at interface level:

```
device(config-if-e10000-1/1)#no ipv6 nd suppress-ra
```

The **ipv6 nd send-ra** command is a new interface level command added as part of this enhancement. This allows the user to configure the sending of RA messages on some selected interfaces when the **ipv6 nd global-suppress-ra** command is set to disable the sending of RA messages on all other interfaces.

**Syntax: [no]ipv6 nd suppress-ra**

# IPv6 source routing security enhancements

The IPv6 specification (RFC 2460) specifies support for IPv6 source-routed packets using a type 0 Routing extension header, requiring device and host to process the type 0 routing extension header. However, this requirement may leave a network open to a DoS attack.

A security enhancement disables sending IPv6 source-routed packets to IPv6 devices either completely or selectively as described in the following sections. (This enhancement conforms to RFC 5095.)

## Complete filtering of IPv6 source-routed packets

Extreme devices are configured to drop all IPv6 source-routed packets in hardware and software as described:

- **Hardware** - IPv6 source-routed packets that contain a type 0 routing extension header immediately after the IPv6 header are dropped in hardware by default.
- **Software** - IPv6 source-routed packets addressed to any IPv6 address on a device (regardless of where the routing extension header is located) are dropped in software by default.

Details of hardware and software filtering of IPv6 source-routed packets is provided in the following.

### *Hardware filtering of IPv6 source-routed packets*

All IPv6 source-routed packets that contain a type 0 routing extension header immediately after the IPv6 header are automatically dropped in hardware. This filtering is performed on both IPv6 packets that require forwarding and IPv6 packets addressed to one of the IPv6 addresses on the device without sending an ICMP error message. This filtering behavior is enabled by default. Consequently, if you want a the device to process IPv6 source-routed packets that contain a type 0 routing extension header immediately after the IPv6 header you must direct it to perform this action through use of the **ipv6 forward-source-route** command, as shown in the following.

```
device(config)# ipv6 forward-source-route
```

**Syntax: [no] ipv6 forward-source-route**

The default condition is for source-routed packets to be dropped. If you enable forwarding using this command, you can return to the default state by using the **no** option in front of the command.

NOTE
Source routed, IPv6 packets where the type 0 routing extension header does not follow directly after the IPv6 header are not automatically dropped in hardware.

### *Software filtering of IPv6 source-routed packets*

By default, all IPv6 source-routed packets addressed to any IPv6 address on a device are dropped by software (regardless of where the Routing Extension Header resides). You can enable the forwarding of these packets by using the **ipv6 source-route** command, as the following example shows.

```
device(config)# ipv6 source-route
```

**Syntax: [no] ipv6 source-route**

The default condition is to disallow the forwarding of source-routed packets to IPv6 addresses. If you enable forwarding by using this command, you can return to the default state by using the **no** option of the command.

The **ipv6 forward-source-route** command must be enabled for the **ipv6 source-route** command to operate.

By default, ICMP error messages are sent for packets dropped by software. You can use the **ipv6 icmp source-route** command to disable the generation of ICMPv6 parameter problem for software discarded IPv6 source-routed packets addressed to one of the IPv6 addresses of a device. This is described in Disabling ICMP error messages for source-routed IPv6 packets on page 174.

## Selective filtering of IPv6 source-routed packets using ACLs

You can selectively filter IPv6 source-routed packets using ACLs. This is accomplished by creating an IPv6 ACL that specifies a type 0 routing extension header. This is done using the **routing-header-type** option when configuring an IPv6 ACL. An example of an IPv6 ACL that selectively drops IPv6 source-routed packets is shown in the following.

```
device(config)# ipv6 access-list deny-access1
device(config-ipv6-access-list deny-access1)#deny ipv6 any any routing-header-type 0
```

As with complete filtering, selective filtering can be done in both hardware and software as described:

- **Hardware** - Inbound and outbound IPv6 source-routed packets that contain a type 0 routing extension header immediately after the IPv6 header can be selectively dropped in hardware through use of an IPv6 ACL and bound to an interface using the **ipv6 traffic-filter** command.
- **Software** - Inbound IPv6 source-routed packets that contain a routing extension header anywhere in a packet can be selectively dropped in software using an IPv6 ACL and bound to interfaces using the **ipv6 access-class** command.

Details about how to configure selective hardware and software filtering of IPv6 source-routed packets are provided in the following.

### *Selective hardware filtering of IPv6 source-routed packets*

Both inbound and outbound IPv6 source-routed packets that contain a type 0 routing extension header immediately after the IPv6 header can be selectively dropped in hardware using an IPv6 ACL. source-routed packets dropped in hardware are dropped without an ICMP error message being sent. To apply an IPv6 ACL with the **routing-header-type** option for hardware filtering, you must apply the IPv6 ACL to specific ports using the **ipv6 traffic-filter** command as shown in the following example.

```
device(config)# interface ethernet 3/1
device(config-if-e100-3/1)# ipv6 traffic-filter deny-access1 in
```

Additionally, you must also enable forwarding using the**ipv6 forward-source-route** command (as shown in the following) to allow any forwarding of IPv6 source-routed packets.

```
device(config)# ipv6 forward-source-route
```

### *Selective software filtering of IPv6 source-routed packets*

Inbound IPv6 source-routed packets that contain a routing extension header anywhere in a packet can be selectively dropped in software using an IPv6 ACL. source-routed packets dropped in software generate ICMP Destination Unreachable error messages.

> NOTE
> This filtering only applies to packets addressed to one of the IPv6 addresses of the device.

To apply an IPv6 ACL with the **routing-header-type** option for software filtering, you must apply the IPv6 ACL system wide using the**ipv6 access-class** command.

```
device(config)# # ipv6 access-class deny-access1 in
```

Additionally, you must also enable forwarding using the**ipv6 forward-source-route** and **ipv6 source-route** commands (as shown in the following) to allow any forwarding of IPv6 source-routed packets.

```
device(config)# ipv6 forward-source-route
device(config)# ipv6 source-route
```

## Complete and selective filtering combination and order of application

If the complete filtering of IPv6 source-routed packets is enabled (the default state) then selective filtering cannot be performed. Consequently, you must use the **ipv6 forward-source-route** and **ipv6 source-route** commands to allow IPv6 source-routed packets when you are selectively allowing some IPv6 source-routed packets.

The following configuration of complete hardware and software filtering can be used with selective filtering if the commands are configured in the correct order:

- When the **ipv6 forward-source-route** command is configured, IPv6 source-routed packets that contain a type 0 routing extension header immediately after the IPv6 header are not dropped by hardware.
- All IPv6 source-routed packets addressed to any IPv6 address on a device (regardless of where the Routing Extension Header is located) are dropped by software. This is the default configuration.

When using the **ipv6 forward-source-route** and **ipv6 source-route** commands as described, the filtering is performed in the order described below.

1. Inbound filtering is performed on the receiving interface using an ACL applied using the **ipv6 traffic-filter** command. This filtering is performed using hardware.
2. Complete filtering for IPv6 source route. This filtering is performed by the CPU.
3. Selective filtering using an IPv6 ACL applied on a system-wide basis using the **ipv6 access-class** command.
4. Selective filtering by hardware using an IPv6 ACL bound to an interface for outbound traffic using the**ipv6 traffic-filter** command.

## Configuration examples for complete and selective filtering of source routed packets

The following examples demonstrate how to use this feature for different purposes:

- Dropping all IPv6 Source Routed Packets on all Ports
- Dropping all IPv6 Source Routed Packets on a Specified Port
- Silently Dropping all IPv6 Source Routed Packets Addressed to IPv6 Addresses
- Dropping all IPv6 Source Routed Packets Addressed to IPv6 Addresses from a Specified Source

- Allowing IPv6 Source Routed Packets from a Specified Source on a Specified Interface

Each of these examples are described in detail in the following sections.

## Dropping all IPv6 source-routed packets on all ports

By default, all IPv6 source-routed packets received on all device ports are dropped.

## Dropping all IPv6 source-routed packets on a specified port

The following example shows a configuration that will drop all IPv6 source-routed packets received on port 1/1 of a device.

In this example, the IPv6 ACL is configured to drop any IPv6 packet with a type 0 routing header immediately after the IPv6 header.

```
device(config)# ipv6 access-list deny-access1
device(config-ipv6-access-list deny-access1)# deny any any ipv6 routing-header-type 0
device(config-ipv6-access-list deny-access1# permit ipv6 any any
device(config-ipv6-access-list deny-access1)# exit
```

The default is for the device to drop all IPv6 source-routed packets in hardware and software. Forwarding of these packets must be explicitly enabled using the **ipv6 forward-source-route** and**ipv6 source-route** commands as shown.

```
device(config)# ipv6 forward-source-route
device(config)# ipv6 source-route
```

The IPv6 ACL must then be bound to the interface it is intended to filter as shown in the following example for the Ethernet 1/1 interface.

```
device(config)# interface ethernet 1/1
device(config-if-e100-1/1)# ipv6 traffic-filter deny-access1 in
```

## Silently dropping all IPv6 source-routed packets sent to IPv6 addresses

The following configuration drops all IPv6 source-routed packets addressed to the IPv6 addresses on a device without sending an ICMP error message.

ICMPv6 parameter problem error messages are sent for dropped IPv6 source-routed packets addressed to the IPv6 addresses on the device. To disable these messages, use the **no** option with the **ipv6 icmp source-route** command.

```
device(config)# no ipv6 icmp source-route
```

By default, the device drops all IPv6 source-routed packets in hardware and software. Use the **ipv6 forward-source- route** command to enable the forwarding of IPv6 source-routed packets with a type 0 routing extension header immediately after the IPv6 header, as shown in this example.

```
device(config)# ipv6 forward-source-route
```

## Dropping all IPv6 source-routed packets to IPv6 addresses from a specified source

This configuration demonstrates how to drop all IPv6 source-routed packets sent from a specified IPv6 address.

In this example, IPv6 ACL is configured to deny IPv6 source-routed packets with a destination address of 2001:DB8:1, and permit any other IPv6 packets.

```
device(config)# ipv6 access-list deny-access2
device(config-ipv6-access-list deny-access2)# deny host 2001:DB8:1 any routing-header-type 0
device(config-ipv6-access-list deny-access2)# permit ipv6 any any
device(config-ipv6-access-list deny-access2)# exit
```

The IPv6 ACL is then applied globally to the device for inbound traffic using the **ipv6 access-class** command as shown.

```
device(config)#ipv6 access-class deny-access2 in
```

By default, the device drops all IPv6 source-routed packets in hardware and software. Use the **ipv6 forward-source- route** and ipv6 source-route commands to enable forwarding of IPv6 source-routed packets, as shown.

```
device(config)# ipv6 forward-source-route
device(config)# ipv6 source-route
```

# Changing the IPv6 MTU

The IPv6 MTU is the maximum length of an IPv6 packet that can be transmitted on a particular interface. If an IPv6 packet is longer than an MTU, the host that originated the packet breaks the packet into fragments and transmits the fragments in multiple packets that are shorter than the configured MTU. You can configure the MTU on individual interfaces. Per RFC 2460, the minimum IPv6 MTU for any interface is 1280 bytes.

> NOTE
> The maximum number of unique MTUs that can be configured on a CES 2000 Series or CER 2000 Series device is 12.

To configure the MTU on interface 3/1 to 1280 bytes, enter the following commands.

```
device(config)# interface ethernet 3/1
device(config-if-e100-3/1)# ipv6 mtu 1280
```

**Syntax: [no] ipv6 mtu** *bytes*

You can specify between 1284 - (**default-max-frame-size** minus 18). If a non-default value is configured for an interface, router advertisements include an MTU option. The minimum values you can configure are: 1298 (**IP6_MIN_MTU** + 18) for Ethernet ports.

You can configure IPv6 MTU for to be greater than 1500 bytes, although the default remains at 1500 bytes.

At the global CLI level, use the **ipv6 global-mtu** command. To define IPv6 MTU globally, enter.

```
device(config)#ipv6 global-mtu 1300
```

**Syntax: [no] ipv6 global-mtu** *value*

> NOTE
> After the configuration of **ipv6 global-mtu** , the system needs to be rebooted to enable the management ethernet controller to reconfigure it.

To define IPv6 MTU on an interface, enter.

```
device(config-if-e1000-2/1)#ipv6 mtu
```

**Syntax: ipv6 mtu** *value*

> NOTE
> If the size of a jumbo packet received on a port is equal to the maximum frame size of - 18 (Layer 2 MAC header + CRC) and if this value is greater than the IPv4/IPv6 MTU of the outgoing port, it will be forwarded to the CPU.

## How to determine the actual IPv6 MTU value

An IPv6 port can obtain an MTU value from any of the following sources:

- Default IP MTU setting
- Global MTU Setting
- Interface MTU Setting

An interface determines the actual MTU value through these processes.

1. If an IPv6 interface MTU value is configured, that value is used.
2. If an IPv6 interface MTU value is not configured and an IPv6 global MTU value is configured, the configured global MTU value is used.
3. If neither an IPv6 interface MTU value or an IPv6 global MTU value are configured, the default IPv6 MTU value of 1500 is used.

# Configuring static neighbor entries

In some cases, a neighbor cannot be reached using neighbor discovery. In this situation, you can add a static entry to the IPv6 neighbor discovery cache, which causes a neighbor to be reachable at all times without using neighbor discovery. (A static entry in the IPv6 neighbor discovery cache functions like a static ARP entry in IPv4.)

For example, to add a static entry for a neighbor with the IPv6 address 2001:DB8:2678::2 and link-layer address 0000.002b.8641 that is reachable through Ethernet interface 3/1, enter the following command.

```
device(config)# ipv6 neighbor 2001:DB8:2678::2 ethernet 3/1 0000.002b.8641
```

**Syntax: [no] ipv6 neighbor** *ipv6-address* **ethernet** *port* | **ve** *ve-number* [ **ethernet** *port* ] *link-layer-address*

The *ipv6-address* parameter specifies the address of the neighbor.

The **ethernet | ve** parameter specifies the interface through which to reach a neighbor. If you specify an Ethernet interface, you must also specify the port number. The link-layer address is a 48-bit hardware address of the neighbor.

> NOTE
> If you specify a VE, you do not have to mandatorily specify the Ethernet port numbers associated with the VE.

If you attempt to add an entry that already exists in the neighbor discovery cache, the software changes the already existing entry to a static entry.

To remove a static IPv6 entry from the IPv6 neighbor discovery cache, use the **no** form of this command.

# Limiting the number of hops an IPv6 packet can traverse

By default, the maximum number of hops an IPv6 packet can traverse is 64. You can change this value to between 1 - 255 hops. For example, to change the maximum number of hops to 70, enter the following command.

```
device(config)# ipv6 hop-limit 70
```

**Syntax: [no] ipv6 hop-limit** *number*

The number of hops can be from 1 - 255.

# Information about IPv6 prefix list

An IPv6 prefix list comprises one or more conditional statements that pose an action (permit or deny) if a route matches a specified prefix. In prefix lists with multiple statements, you can specify a sequence number for each statement. The specified sequence number determines the order in which the statement appears in the prefix.

You can configure an IPv6 prefix list on a global basis and use it as input to other commands or processes, such as route aggregation, route redistribution, route distribution, route maps, and so on. When a device sends or receives an IPv6 route, it applies the statements within the IPv6 prefix list in their order of appearance to the packet. When a match occurs, the device takes the specified action (permit or deny the packet) and stops further comparison for that route.

You can use permit statements in the prefix list to specify the route that you want to send to the other feature. If you use deny statements, the route specified by the deny statements is not supplied to the other feature.

A device supports IPv6 prefix lists, which you can use for basic route filtering. You can configure up to 100 IPv6 prefix lists.
You must specify the ipv6-prefix parameter in hexadecimal using 16-bit values between colons as documented in RFC 4291. You must specify the prefix-length parameter as a decimal value. A slash mark (/) must follow the ipv6-prefix parameter and precede the prefix-length parameter.

# Displaying prefix list information

To display the IPv6 prefix lists configured on a device, use the **show ipv6 prefix-lists** command. Extreme

```
device# show ipv6 prefix-lists
routesfor2001
 ipv6 prefix-list routesfor2001
   seq 5 permit 2001::/16
   seq 10 permit 2001:db8::/32
```

# Managing a Device Over IPv6

You can perform system management tasks for the device using the **copy** , **ncopy** , **ping** , **telnet** , and **traceroute** commands and Secure Shell (SSH). These commands and SSH now function over IPv6.

This section describes the IPv6-related syntax added to the commands and SSH. It does not describe the already existing command syntax for IPv4.

## Using the IPv6 copy command

The **copy** command for IPv6 allows you to do the following:

- Copy a file from a specified source to an IPv6 TFTP server.
- Copy a file from an IPv6 TFTP server to a specified destination.

### *Copying a file to an IPv6 TFTP server*

You can copy a file from the following sources to an IPv6 TFTP server:

- Flash memory.
- Running configuration.
- Startup configuration.

### Copying a file from flash memory

For example, to copy the primary or secondary boot image from the device's flash memory to an IPv6 TFTP server, enter a command such as the following.

```
device# copy flash tftp ipv6 2001:db8:e0ff:7837::3 test.img secondary
```

This command copies the secondary boot image named test.img from flash memory to a TFTP server with the IPv6 address of 2001:db8:e0ff:7837::3.

**Syntax: copy flash tftp ipv6 source-file-name primary | secondary**

The *ipv6-address* parameter specifies the address of the TFTP server. You must specify this address in hexadecimal using 16-bit values between colons as documented in RFC 2373.

The *source-file-name* parameter specifies the name of the file you want to copy to the IPv6 TFTP server.

The **primary** keyword specifies the primary boot image, while the **secondary** keyword specifies the secondary boot image.

### Copying a file from the running or startup configuration

For example, to copy the running configuration to an IPv6 TFTP server, enter a command such as the following.

```
device# copy running-config tftp ipv6 2001:db8:e0ff:7837::3 newrun.cfg
```

This command copies the running configuration to a TFTP server with the IPv6 address of 2001:db8:e0ff:7837::3 and names the file on the TFTP server newrun.cfg.

**Syntax: copy running-config | startup-config tftp ipv6 destination-file-name**

Specify the **running-config** keyword to copy the running configuration file to the specified IPv6 TFTP server.

Specify the **startup-config** keyword to copy the startup configuration file to the specified IPv6 TFTP server.

The tftp *ipv6-address* parameter specifies the address of the TFTP server. You must specify this address in hexadecimal using 16-bit values between colons as documented in RFC 2373.

The *destination-file-name* parameter specifies the name of the file that is copied to the IPv6 TFTP server.

## *Copying a file from an IPv6 TFTP server*

You can copy a file from an IPv6 TFTP server to the following destinations:

- Flash memory.
- Running configuration.
- Startup configuration.

### Copying a file to flash memory

For example, to copy a boot image from an IPv6 TFTP server to the primary or secondary storage location in the device's flash memory, enter a command such as the following.

```
device# copy tftp flash ipv6 2001:db8:e0ff:7837::3 test.img secondary
```

This command copies an application image named test.img from an IPv6 TFTP server with the IPv6 address of 2001:db8:e0ff:7837::3 to the secondary storage location in the device's flash memory.

**Syntax: copy tftp flash ipv6 source-file-name primary | secondary**

The *ipv6-address* parameter specifies the address of the TFTP server. You must specify this address in hexadecimal using 16-bit values between colons as documented in RFC 2373.

The *source-file-name* parameter specifies the name of the file you want to copy from the IPv6 TFTP server.

The **primary** keyword specifies the primary storage location in the device's flash memory, while the **secondary** keyword specifies the secondary storage location in the device's flash memory.

### Copying a file to the running or startup configuration

For example, to copy a configuration file from an IPv6 TFTP server to the router's running or startup configuration, enter a command such as the following.

```
device# copy tftp running-config ipv6 2001:db8:e0ff:7837::3 newrun.cfg overwrite
```

This command copies the newrun.cfg file from the IPv6 TFTP server and overwrites the router's running configuration file with the contents of newrun.cfg.

> NOTE
> To activate this configuration, you must reload (reset) the device.

**Syntax: copy tftp running-config** | **startup-config** *ipv6-address source-file-name* [ **overwrite** ]

Specify the **running-config** keyword to copy the running configuration from the specified IPv6 TFTP server.

Specify the **startup-config** keyword to copy the startup configuration from the specified IPv6 TFTP server.

The *ipv6-address* parameter specifies the address of the TFTP server. You must specify this address in hexadecimal using 16-bit values between colons as documented in RFC 2373.

The *source-file-name* parameter specifies the name of the file that is copied from the IPv6 TFTP server.

The **overwrite** keyword specifies that the device should overwrite the current configuration file with the copied file. If you do not specify this parameter, the device copies the file into the current running or startup configuration but does not overwrite the current configuration.

> NOTE
> You cannot use the overwrite option from non-console sessions, because it will disconnect the session.

## Using the IPv6 ncopy command

The **ncopy** command for IPv6 allows you to do the following:

- Copy a primary or secondary boot image from flash memory to an IPv6 TFTP server.
- Copy the running configuration to an IPv6 TFTP server.
- Copy the startup configuration to an IPv6 TFTP server
- Upload various files from an IPv6 TFTP server.

### *Copying a primary or secondary boot image from flash memory to an IPv6 TFTP server*

For example, to copy the primary or secondary boot image from the device's flash memory to an IPv6 TFTP server, enter a command such as the following.

```
device# ncopy flash primary tftp ipv6 2001:db8:e0ff:7837::3 primary.img
```

This command copies the primary boot image named primary.img from flash memory to a TFTP server with the IPv6 address of 2001:db8:e0ff:7837::3.

**Syntax: ncopy flash primary | secondary tftp ipv6 source-file-name**

The **primary** keyword specifies the primary boot image, while the **secondary** keyword specifies the secondary boot image.

The **tftp***ipv6-address* parameter specifies the address of the TFTP server. You must specify this address in hexadecimal using 16-bit values between colons as documented in RFC 2373.

The *source-file-name* parameter specifies the name of the file you want to copy from flash memory.

## *Copying the running or startup configuration to an IPv6 TFTP server*

For example, to copy a device's running or startup configuration to an IPv6 TFTP server, enter a command such as the following.

```
device# ncopy running-config tftp ipv6 2001:db8:e0ff:7837::3 bakrun.cfg
```

This command copies a device's running configuration to a TFTP server with the IPv6 address of 2001:db8:e0ff:7837::3 and names the destination file bakrun.cfg.

**Syntax: ncopy running-config | startup-config tftp ipv6 destination-file-name**

Specify the **running-config** keyword to copy the device's running configuration or the **startup-config** keyword to copy the device's startup configuration.

The **tftp***ipv6-address* parameter specifies the address of the TFTP server. You must specify this address in hexadecimal using 16-bit values between colons as documented in RFC 2373.

The *destination-file-name* parameter specifies the name of the running configuration that is copied to the IPv6 TFTP server.

## *Uploading files from an IPv6 TFTP server*

You can upload the following files from an IPv6 TFTP server:

- Primary boot image.
- Secondary boot image.
- Running configuration.
- Startup configuration.

### Uploading a primary or secondary boot image from an IPv6 TFTP server

For example, to upload a primary or secondary boot image from an IPv6 TFTP server to a device's flash memory, enter a command such as the following.

```
device# ncopy tftp ipv6 2001:db8:e0ff:7837::3 primary.img flash primary
```

This command uploads the primary boot image named primary.img from a TFTP server with the IPv6 address of 2001:db8:e0ff:7837::3 to the device's primary storage location in flash memory.

**Syntax: ncopy tftp ipv6 source-file-name flash primary | secondary**

The **tftp***ipv6-address* parameter specifies the address of the TFTP server. You must specify this address in hexadecimal using 16-bit values between colons as documented in RFC 2373.

The *source-file-name* parameter specifies the name of the file you want to copy from the TFTP server.

The **primary** keyword specifies the primary location in flash memory, while the **secondary** keyword specifies the secondary location in flash memory.

### Uploading a running or startup configuration from an IPv6 TFTP server

For example to upload a running or startup configuration from an IPv6 TFTP server to a device, enter a command such as the following.

```
device# ncopy tftp ipv6 2001:db8:e0ff:7837::3 newrun.cfg running-config
```

This command uploads a file named newrun.cfg from a TFTP server with the IPv6 address of 2001:db8:e0ff:7837::3 to the device.

**Syntax: ncopy tftp ipv6 source-file-name running-config** | **startup-config**

The **tftp***ipv6-address* parameter specifies the address of the TFTP server. You must specify this address in hexadecimal using 16-bit values between colons as documented in RFC 2373.

The *source-file-name* parameter specifies the name of the file you want to copy from the TFTP server.

Specify the **running-config** keyword to upload the specified file from the IPv6 TFTP server to the device. The device copies the specified file into the current running configuration but does not overwrite the current configuration.

Specify the **startup-config** keyword to upload the specified file from the IPv6 TFTP server to the device. The device copies the specified file into the current startup configuration but does not overwrite the current configuration.

## Using the IPv6 ping command

The **ping** command allows you to verify the connectivity from a device to an IPv6 device by performing an ICMP for IPv6 echo test.

For example, to ping a device with the IPv6 address of 2001:db8:847f:a385:34dd::45 from the device, enter the following command.

```
device# ping ipv6 2001:db8:847f:a385:34dd::45
```

**Syntax: ping ipv6 ipv6-address** [ **outgoing-interface** [ **port** | **ve number** ] ] [ **source ipv6-address** ] [ **count number** ] [ **timeout milliseconds** ] [ **ttl number** ] [ **size bytes** ] [ **quiet** ] [ **numeric** ] [ **no-fragment** ] [ **verify** ] [ **data** *1-to-4 bytehex* ] [ **brief** ]

The *ipv6-address* parameter specifies the address of the router. You must specify this address in hexadecimal using 16-bit values between colons as documented in RFC 2373.

The **outgoing-interface** keyword specifies a physical interface over which you can verify connectivity. If you specify a physical interface, such as an Ethernet interface, you must also specify the port number of the interface. If you specify a virtual interface, such as a VE, you must specify the number associated with the VE.

The **source***ipv6-address* parameter specifies an IPv6 address to be used as the origin of the ping packets.

The**count***number* parameter specifies how many ping packets the router sends. You can specify from 1 - 4294967296. The default is 1.

The **timeout***milliseconds* parameter specifies how many milliseconds the router waits for a reply from the pinged device. You can specify a timeout from 1 - 4294967296 milliseconds. The default is 5000 (5 seconds).

The **ttl** *number* parameter specifies the maximum number of hops. You can specify a TTL from 1 - 255. The default is 64.

The**size***bytes* parameter specifies the size of the ICMP data portion of the packet. This is the payload and does not include the header. You can specify from 0 - 4000. The default is 16.

The **no-fragment** keyword turns on the "don't fragment" bit in the IPv6 header of the ping packet. This option is disabled by default.

The **quiet** keyword hides informational messages such as a summary of the ping parameters sent to the device and instead only displays messages indicating the success or failure of the ping. This option is disabled by default.

The **verify** keyword verifies that the data in the echo packet (the reply packet) is the same as the data in the echo request (the ping). By default the device does not verify the data.

The **data***1 - 4 byte hex* parameter lets you specify a specific data pattern for the payload instead of the default data pattern, "abcd", in the packet's data payload. The pattern repeats itself throughout the ICMP message (payload) portion of the packet.

> NOTE
> For parameters that require a numeric value, the CLI does not check that the value you enter is within the allowed range. Instead, if you do exceed the range for a numeric value, the software rounds the value to the nearest valid value.

The**brief** keyword causes ping test characters to be displayed. The following ping test characters are supported:

! Indicates that a reply was received.

. Indicates that the network server timed out while waiting for a reply.

U Indicates that a destination unreachable error PDU was received.

I Indicates that the user interrupted ping.

# Using the traceroute command with IPv6 addresses

The **traceroute** command allows you to trace a path from the device to an IPv6 host.

The command output displays traceroute information for each hop as soon as the information is received. Traceroute requests display all responses to a minimum TTL of 1 second and a maximum Time To Live (TTL) of 30 seconds. In addition, if there are multiple equal-cost routes to the destination, the device displays up to three responses.

For example, to trace the path from the device to a host with an IPv6 address of 2001:db8:349e:a384::34, enter the following command.

```
device# traceroute ipv6 2001:db8:349e:a384::34
```

Using the **source-ip** option, you can specify a source IP address. If the source IP address is an IPv6 link-local address, the destination address must be no more than one hop away in the network. An IPv6 link-local address cannot be routed.

```
device# traceroute ipv6 2001:db8:349e:a384::34 source-ip fec0:60:69bc:92:205:33ff:fe9e:3f20
```

**Syntax: traceroute ipv6** { *ipv6-address* | *ipv6-host-name* } [ **maxttl** *value* ] [ **minttl** *value* ] [ **numeric** ] [ **source-ip** *address* ] [ **timeout** *seconds* ] [ **vrf** *vrf-name* ]

The *ipv6-address* variable specifies the IPv6 address of a host. You must specify this address in hexadecimal using 16-bit values between colons as documented in RFC 2373.

# Using Telnet

This section explains how to do the following:

- Use the **telnet** command to establish a Telnet session from the device to a remote IPv6 host.
- Establish a Telnet session from a remote IPv6 host to the device.

## *Using the IPv6 Telnet command*

The **telnet** command allows a Telnet connection from a device to a remote IPv6 host using the console. Up to five read-access and one write-access inbound Telnet session are supported on the router at one time. Up to five simultaneous outbound Telnet sessions can also be supported from the console session, from inbound Telnet sessions, from inbound SSH sessions or from a Web session.

To see the Telnet sessions currently open on the device, enter the **show telnet** command; to see both the open Telnet and open SSH sessions, enter the **show who** command as shown below.

```
device# show who
Console connections:
        established
        3 days 17 hours 31 minutes 27 seconds in idle
Telnet server status: Enabled
Telnet connections (inbound):
 1      established, client ip address 10.53.1.86
        you are connecting to this session
        1 seconds in idle
 2      established, client ip address 10.53.1.86
        7 seconds in idle
 3      closed
 4      closed
 5      closed
Telnet connections (outbound):
 6      established, server ip address 10.47.2.200, from Telnet session 1
        4 seconds in idle
 7      closed
 8      closed
 9      closed
 10     closed
SSH server status: Enabled
SSH connections:
 1      closed
 2      closed
 3      closed
 4      closed
...
```

**Syntax: show who**

To establish a Telnet connection to a remote host, use the **telnet** command. The following example will establish an outbound Telnet connection to a remote host with the IPv6 address of 2001:db8:3de2:c37::6.

```
device# telnet 2001:db8:3de2:c37::6
```

**Syntax: telnet ipv6-address** [ **port-number** | **outgoing-interface ethernet port** | **ve number** ]

The **ipv6-address** parameter specifies the address of a remote host. You must specify this address in hexadecimal using 16-bit values between colons as documented in RFC 2373.

The *port-number* parameter specifies the port number on which the device establishes the Telnet connection. You can specify a value between 1 - 65535. If you do not specify a port number, the device establishes the Telnet connection on port 23.

If the IPv6 address you specify for the **telnet***ipv6-address* command is a link-local address, you must specify the **outgoing-interface** ethernet *port* | *ve number* parameter. This parameter specifies the interface that must be used to reach the remote host. If you specify an Ethernet interface, also specify the port number associated with the interface. If you specify a VE interface, also specify the VE number.

### *Establishing a Telnet session from an IPv6 host*

To establish a Telnet session from an IPv6 host to the device, open your Telnet application and specify the IPv6 address of the router.

# Using Secure Shell

Secure Shell (SSH) is a mechanism that allows secure remote access to management functions on the device. SSH provides a function similar to Telnet. You can log into and configure the device using a publicly or commercially available SSH client program, just as you can with Telnet. However, unlike Telnet, which provides no security, SSH provides a secure, encrypted connection to the device.

To open an SSH session from an IPv6 host running an SSH client program to the device, open your SSH client program and specify the IPv6 address of the router.

# Clearing global IPv6 information

You can clear the following global IPv6 information:

- Entries from the IPv6 cache.
- Entries from the IPv6 neighbor table.
- IPv6 routes from the IPv6 route table.
- IPv6 traffic statistics.
- IPv6 session flows

## Clearing the IPv6 cache

You can remove all entries from the IPv6 cache or specify an entry based on the following:

- IPv6 prefix.
- IPv6 address.
- Interface type.

For example, to remove entries for IPv6 address 2000:e0ff::1, enter the following command at any configuration level of the CLI.

```
device# clear ipv6 cache 2000:e0ff::1
```

**Syntax: clear ipv6 cache** [ *ipv6-prefix/prefix-length* | *ipv6-address* | **ethernet** *port* | **tunnel** *number* | **ve** *number* ] [ **vrf** *vrf-name* ]

You must specify the *ipv6-prefix* parameter in hexadecimal using 16-bit values between colons as documented in RFC 2373. You must specify the *prefix-length* parameter as a decimal value. A slash mark (/) must follow the *ipv6-prefix* parameter and precede the *prefix-length* parameter.

You must specify the *ipv6-address* parameter in hexadecimal using 16-bit values between colons as documented in RFC 2373.

The **ethernet** |**tunnel** |**ve** parameter specifies the interfaces for which you can remove cache entries. If you specify an Ethernet interface, you must also specify the port number associated with the interface. If you specify a VE or tunnel interface, also specify the VE or tunnel number, respectively.

The *vrf-name* parameter specifies the VRF for which you want to clear the cache entry. If no vrf parameter is entered, the default VRF is used.

## Clearing IPv6 neighbor information

You can remove all entries from the IPv6 neighbor table or specify an entry based on the following:

- IPv6 prefix.
- IPv6 address.
- Interface type.

For example, to remove entries for Ethernet interface 3/1, enter the following command at the Privileged EXEC level or any of the CONFIG levels of the CLI.

```
device# clear ipv6 neighbor ethernet 3/1
```

**Syntax: clear ipv6 neighbor** [ *ipv6-prefix/prefix-length* | *ipv6-address* | **ethernet** *port* | **ve** *number* ]

You must specify the *ipv6-prefix* parameter in hexadecimal using 16-bit values between colons as documented in RFC 2373. You must specify the *prefix-length* parameter as a decimal value. A slash mark (/) must follow the *ipv6-prefix* parameter and precede the *prefix-length* parameter.

You must specify the *ipv6-address* parameter in hexadecimal using 16-bit values between colons as documented in RFC 2373.

The **ethernet** |**ve** parameter specifies the interfaces for which you can remove cache entries. If you specify an Ethernet interface, you must also specify the port number associated with the interface. If you specify a VE, you must also specify the VE number.

## Clearing IPv6 routes from the IPv6 route table

You can clear all IPv6 routes or only those routes associated with a particular IPv6 prefix from the IPv6 route table and reset the routes.

For example, to clear IPv6 routes associated with the prefix 2000:7838::/32, enter the following command at any configuration level of the CLI.

```
device# clear ipv6 route 2000:7838::/32
```

**Syntax: clear ipv6 route** [ *ipv6-prefix/prefix-length* ] | **nexthop** *nexthop_ID*

The *ipv6-prefix/prefix-length* parameter clears routes associated with a particular IPv6 prefix. You must specify the *ipv6-prefix* parameter in hexadecimal using 16-bit values between colons as documented in RFC 2373. You must specify the *prefix-length* parameter as a decimal value. A slash mark (/) must follow the *ipv6-prefix* parameter and precede the *prefix-length* parameter.

The **nexthop** option clears the nexthop information for all next hops in the routing table or for a specific entry. The *nexthop_id* parameter is a specific nexthop entry from the next hop table.

## Clearing IPv6 traffic statistics

To clear all IPv6 traffic statistics (reset all fields to zero), enter the following command at the Privileged EXEC level or any of the Config levels of the CLI.

```
device(config)# clear ipv6 traffic
```

**Syntax: clear ipv6 traffic**

## Clearing statistics for IPv6 subnet rate limiting

To clear the rate limit statistics for IPv6 subnet addresses, enter the **clear rate-limit ipv6 subnet** command at the configuration level.

**Syntax: clear rate-limit ipv6 subnet**

# Displaying global IPv6 information

You can display output for the following global IPv6 parameters:

- IPv6 cache.
- IPv6 interfaces.
- IPv6 neighbors.
- IPv6 route table.
- Local IPv6 routers.

- IPv6 TCP connections and the status of individual connections.
- IPv6 traffic statistics.
- IPv6 session flows

# Displaying IPv6 cache information

The IPv6 cache contains an IPv6 host table with indices to the next hop gateway and the interface on which the route was learned.

To display IPv6 cache information, enter the following command at any CLI level.

```
device# show ipv6 cache
Total number of IPv6 and IPv6 VPN cache entries: 3
       IPv6 Address                      Next Hop                    Interface
1      6000::                            LOCAL                       ve 60
2      6000::2                           LOCAL                       ve 60
3      fe80::768e:f8ff:fe2a:6200         LOCAL                       ve 60
```

**Syntax: show ipv6 cache** [ *index-number* | *ipv6-prefix/prefix-length* | *ipv6-address* | **ethernet** *port* | **ve** *number* | **tunnel** *number* ] [ **vrf** *vrf-name* ]

The *index-number* parameter restricts the display to the entry for the specified index number and subsequent entries.

The *ipv6-prefix/prefix-length* parameter restricts the display to the entries for the specified IPv6 prefix. You must specify the *ipv6-prefix* parameter in hexadecimal using 16-bit values between colons as documented in RFC 2373. You must specify the *prefix-length* parameter as a decimal value. A slash mark (/) must follow the *ipv6-prefix* parameter and precede the *prefix-length* parameter.

The **ethernet | ve | tunnel** parameter restricts the display to the entries for the specified interface. The *ipv6-address* parameter restricts the display to the entries for the specified IPv6 address. You must specify this parameter in hexadecimal using 16-bit values between colons as documented in RFC 2373.

The **vrf***vrf-name* parameter specifies the VRF for which you want to display the cache entry. If a vrf parameter is not entered, then the default VRF is used.

If you specify an Ethernet interface, also specify the port number associated with the interface. If you specify a VE interface, you must also specify the VE number. If you specify a tunnel interface, you must also specify the tunnel number.

This display shows the following information.

**TABLE 21** IPv6 cache information fields

| This field... | Displays... |
|---|---|
| Total number of cache entries | The number of entries in the cache table. |
| IPv6 Address | The host IPv6 address. |
| Next Hop | The next hop, which can be one of the following:<br>• Direct - The next hop is directly connected to the device.<br>• Local - The next hop is originated on this device.<br>• *ipv6 address* - The IPv6 address of the next hop. |
| Port | The port on which the entry was learned. |

# Displaying IPv6 interface information

To display IPv6 interface information, enter the following command at any CLI level.

```
device# show ipv6 interface
device# show ipv6 interface
Type Codes - I:ISIS  O:OSPF R:RIP
```

```
Interface     Stat/Prot IGPs IPv6 Address                              VRF
eth 2/4       down/down                                                default-vrf
                        2001:db8:2017::c017:101/64
ve 60         up/up     fe80::768e:f8ff:fe2a:6200                      default-vrf
                        6000::2/64
                        6000::/64[Anycast]
```

**Syntax: show ipv6 interface** [ *interface* [ *port-number* | *number* ] ]

The *interface* parameter displays detailed information for a specified interface. For the interface, you can specify the **Ethernet**, **loopback**, **tunnel**, or **VE** keywords. If you specify an Ethernet interface, you must also specify the port number associated with the interface. If you specify a loopback, tunnel, or VE interface, you must also specify the number associated with the interface.

This display shows the following information.

**TABLE 22** General IPv6 interface information fields

| This field... | Displays... |
|---|---|
| Routing protocols | A one-letter code that represents a routing protocol that can be enabled on an interface. |
| Interface | The interface type, and the port number or number of the interface. |
| Status | The status of the interface. The entry in the Status field will be either "up/up" or "down/down". |
| Routing | The routing protocols enabled on the interface. |
| Global Unicast Address | The global unicast address of the interface. |

## *Displaying IPv6 interface information for a specified interface*

To display detailed information for a specific interface, enter a command such as the following at any CLI level.

```
device# show ipv6 interface ethernet 3/1
Brcoade# show ipv6 interface ethernet 2/2
Interface Ethernet 2/2 is up, line protocol is up
  IPv6 is enabled, link-local address is fe80::768e:f8ff:fe2a:6231 [Preferred]
  Global unicast address(es):
    180::1 [Preferred],  subnet is 180::/64
    180:: [Anycast],  subnet is 180::/64
  Joined group address(es):
    ff02::1:ff00:1
    ff02::1:ff00:0
    ff02::1:ff2a:6231
Port belongs to VRF: default-vrf
  MTU is 1500 bytes
  ICMP redirects are disabled
  ND DAD is enabled, number of DAD attempts: 3
  ND reachable time is 30000 Milliseconds
  ND advertised reachable time is 0 seconds
  ND retransmit interval is 1000 milliseconds
  ND advertised retransmit interval is 0 milliseconds
  ND next router advertisement will be sent in 270 seconds
  ND router advertisements live for 1800 seconds
  No Inbound Access List Set
  No Outbound Access List Set
  IPv6 RPF mode: None  IPv6 RPF Log: Disabled
  RxPkts:       5                    TxPkts:   16
  RxBytes:      730                  TxBytes:  1936
  IPv6 unicast RPF drop: 0
  IPv6 unicast RPF suppressed drop: 0
```

This display shows the following information.

**TABLE 23** Detailed IPv6 interface information fields

| This field... | Displays... |
|---|---|
| Interface/line protocol status | The status of interface and line protocol. If you have disabled the interface with the**disable** command, the status will be "administratively down". Otherwise, the status is either "up" or "down". |
| IPv6 status/link-local address | The status of IPv6. The status is either "enabled" or "disabled".<br>Displays the link-local address, if one is configured for the interface. |
| Global unicast address(es) | Displays the global unicast addresses, if one or more are configured for the interface. |
| Joined group address(es) | The multicast addresses that a device interface listens for and recognizes. |
| MTU | The setting of the maximum transmission unit (MTU) configured for the IPv6 interface. The MTU is the maximum length an IPv6 packet can have to be transmitted on the interface. If an IPv6 packet is longer than an MTU, the host that originated the packet fragments the packet and transmits its contents in multiple packets that are shorter than the configured MTU. |
| ICMP | The setting of the ICMP redirect parameter for the interface. |
| ND | The setting of the various neighbor discovery parameters for the interface. |
| Access List | The inbound and outbound access lists applied to the interface. |
| Routing protocols | The routing protocols enabled on the interface. |
| RxPkts | The number of packets received at the specified port. This field supports IPv4 and IPv6 packet and byte counters. |
| TxPkts | The number of packets transmitted from the specified port. This field supports IPv4 and IPv6 packet and byte counters. |
| RxBytes | The number of bytes received at the specified port. This field supports IPv4 and IPv6 packet and byte counters.s |
| TxBytes | The number of bytes transmitted from the specified port. This field supports IPv4 and IPv6 packet and byte counters. |

# Displaying interface counters for all ports

Previous versions of the NetIron software support IPv4 and IPv6 packet and byte counters. The contents of these counters can be displayed for all ports on a device or per port. Output from the**show ipv6 interface ethernet** command includes packet and byte counter information on a per-port basis. Refer to Displaying IPv6 interface information for a specified interface on page 200.

The default byte counters include the 20-byte per-packet Ethernet overhead. You can configure a device to exclude the 20-byte per-packet Ethernet overhead from byte accounting using the**vlan-counter exclude-overhead** command.

IPv4 and IPv6 commands display the interface counters for all ports on a device. The following example displays packet and byte counter information for all ports.

```
device# show ipv6 interface counters
Interface      RxPkts     TxPkts     RxBytes     TxBytes
eth 3/3        200        200        850000      850000
eth 3/4        500        500        40000       40000
```

**Syntax: show ipv6 interface counters**

Table 24 describes the fields that display interface counter statistics.

**TABLE 24** Interface counter display statistics

| This field... | Displays... |
|---|---|
| Interface | The interface for which counter statistics are being displayed. |

**TABLE 24** Interface counter display statistics (continued)

| This field... | Displays... |
|---|---|
| RxPkts | The number of packets received at the specified port. |
| TxPkts | The number of packets transmitted from the specified port. |
| RxBytes | The number of bytes received at the specified port. |
| TxBytes | The number of bytes transmitted from the specified port. |

### *Clearing the interface counters*

Use the following command to clear all interface counters on a device.

```
device# clear ipv6 interface counters
```

**Syntax: clear ipv6 interface counters**

Use the following command to clear the interface counters for a specified port.

```
device# clear ipv6 interface ethernet 3/2
```

**Syntax: clear ipv6 interface ethernet** *port-number*

The*port-number* variable specifies the slot and port number for which you want to clear the interface counters.

## Displaying IPv6 neighbor information

You can display the IPv6 neighbor table, which contains an entry for each IPv6 neighbor with which the device exchanges IPv6 packets.

To display the IPv6 neighbor table, enter the following command at any CLI level.

```
device(config)# show ipv neighbor ethernet 3/11
Total number of Neighbor entries: 1
Type Codes - *:Static
Entries on interface eth 3/11 :
    IPv6 Address                            VLAN LinkLayer-Addr State  Age  Port  R
1     128::                                   1    0024.3898.0f0a *REACH 422  3/11  0
device(config)#
device(config)# show ipv neighbor ethernet 3/11
Total number of Neighbor entries: 1
Entries on interface eth 3/11 :
    IPv6 Address                            VLAN LinkLayer-Addr State  Age  Port  R
1     128::                                   1    0024.3898.0f0a *REACH 432  3/11  0
After I Ping the neighbor
device(config)# ping ipv6 128::
Sending 1, 16-byte ICMPv6 Echo to 128::
timeout 5000 msec, Hop Limit 64
Type Control-c to abort
Reply from 128::: bytes=16 time=2ms Hop Limit=64
Success rate is 100 percent (1/1), round-trip min/avg/max=2/2/2 ms.
device(config)#
device(config)# show ipv neighbor ethernet 3/11
Total number of Neighbor entries: 2
Entries on interface eth 3/11 :
    IPv6 Address                            VLAN LinkLayer-Addr State  Age  Port  R
1     128::                                   1 0024.3898.0f0a *REACH 429493/11  0
2     fe80::224:38ff:fe98:f0a                 1 0024.3898.0f0a STALE  264 3/11  1
device(config)#
device(config)# show ipv neighbor ethernet 3/11
Total number of Neighbor entries: 2
Entries on interface eth 3/11 :
    IPv6 Address                            VLAN LinkLayer-Addr State  Age  Port  R
1     128::                                   1 0024.3898.0f0a *REACH 42949 3/11  0
2     fe80::224:38ff:fe98:f0a                 1 0024.3898.0f0a STALE 266 3/11  1
device(config)# show ipv neighbor ethernet 3/11
```

```
Total number of Neighbor entries: 2
Entries on interface eth 3/11 :
    IPv6 Address                              VLAN LinkLayer-Addr State  Age  Port  R
1     128::                                     1    0024.3898.0f0a *REACH 35   3/11  0
2     fe80::224:38ff:fe98:f0a                   1    0024.3898.0f0a STALE  60 3/11  1
```

**Syntax: show ipv6 neighbor** [ *ipv6-prefix/prefix-length* | *ipv6-address* | *interface* [ *port* | *number* ] ]

The *ipv6-prefix/prefix-length* parameters restrict the display to the entries for the specified IPv6 prefix. You must specify the *ipv6-prefix* parameter in hexadecimal using 16-bit values between colons as documented in RFC 2373. You must specify the *prefix-length* parameter as a decimal value. A slash mark (/) must follow the *ipv6-prefix* parameter and precede the *prefix-length* parameter.

The *ipv6-address* parameter restricts the display to the entries for the specified IPv6 address. You must specify this parameter in hexadecimal using 16-bit values between colons as documented in RFC 2373.

The *interface* parameter restricts the display to the entries for the specified Extreme device interface. For this parameter, you can specify the **Ethernet** or **VE** keywords. If you specify an Ethernet interface, you must also specify the port number associated with the interface. If you specify a VE interface, you must also specify the VE number.

This display shows the following information.

**TABLE 25** IPv6 neighbor information fields

| This field... | Displays... |
|---|---|
| Total number of neighbor entries | The total number of entries in the IPv6 neighbor table. |
| Type Codes | Shows the route type, which can be one of the following:<br>• B - The route is learned from BGP4+.<br>• C - The destination is directly connected to the router.<br>• I - The route is learned from IPv6 IS-IS.<br>• L - The route is the host address of a loopback interface that is assigned an IPv6 address.<br>• O - The route is learned from OSPFv3.<br>• R - The route is learned from RIPng.<br>• S - The route is a static route. |
| IPv6 Address | The 128-bit IPv6 address of the neighbor. |
| Link-Layer Address | The 48-bit interface ID of the neighbor. |
| State | The current state of the neighbor. Possible states are as follows:<br>• INCOMPLETE - Address resolution of the entry is being performed.<br>• REACH - The forward path to the neighbor is functioning properly.<br>• STALE - This entry has remained unused for the maximum interval. While stale, no action takes place until a packet is sent.<br>• DELAY - This entry has remained unused for the maximum interval, and a packet was sent before another interval elapsed.<br>• PROBE - Neighbor solicitation are transmitted until a reachability confirmation is received. |
| Age | Specifies the time (in seconds) for which the IPv6 neighbor is active. When the global timer 7200 seconds expires, the IPv6 neighbor entry is cleared from the neighbor table. |
| Port | The port on which the entry was learned.<br>If the ND6 entry is configured without the need to include physical interface on a VE interface, then the PORT value is indicated as INV. |
| R | Determines if the neighbor is a device or host: |

**TABLE 25** IPv6 neighbor information fields (continued)

| This field... | Displays... |
|---|---|
| | 0 - Indicates that the neighbor is a host.<br>1 - Indicates that the neighbor is a device. |

The following command example indicates specific static ND6 entries.

```
device#show ipv6 neighbor
Total number of Neighbor entries: 4
Entries in default VRF:
    IPv6 Address                              VLAN      LinkLayer-Addr   State       Age    Port  R
1     fe80::204:80ff:fea0:4060                  1     0004.80a0.4060      REACH       11     3/1   1
2     fe80::204:80ff:fea0:4061                  1     0004.80a0.4061      STALE       4622   3/2   1
3     99::2                                     1     0004.80a0.4060      *INCOMP           0       Inv   1
4     199::2                                    1     0004.80a0.4061      *REACH      13     3/2   1
```

# Displaying the IPv6 route table

To display the IPv6 route table, enter the following command at any CLI level.

```
device# show ipv6 route
IPv6 Routing Table - 2 entries:
Type Codes - B:BGP C:Connected I:ISIS L:Local O:OSPF R:RIP S:Static
BGP  Codes - i:iBGP e:eBGP
ISIS Codes - L1:Level-1 L2:Level-2
OSPF Codes - i:Inter Area 1:External Type 1 2:External Type 2
STATIC Codes - d:DHCPv6
Type IPv6 Prefix          Next Hop Router Interface   Dis/Metric Uptime  src-vrf
C    2001:db8::/64        ::              eth 1/7     0/0        45m18s  -
C    2001:db8:0:25::/64   ::              loopback 1  0/0        1h0m    -
L    2001:db8:0:25::1/128 ::              loopback 1  0/0        13m18s  -
C    2001:db8:2000::/64   ::              eth 1/13    0/0        1h0m    -
O    2001:db8:4000::1/128       fe80::202:17ff:fe6e:c41c  eth 1/13    110/1      2m42s   -
```

**Syntax: show ipv6 route** [ *ipv6-address* | *ipv6-prefix/prefix-length* | **bgp** | **connect** | **ospf** | **rip** | **isis** | **static** | **summary** | **tags** | **nexthop** *nexthop_id* | **ref-routes** ]

The *ipv6-address* parameter restricts the display to the entries for the specified IPv6 address. You must specify the *ipv6-address* parameter in hexadecimal using 16-bit values between colons as documented in RFC 2373.

The *ipv6-prefix*/*prefix-length* parameters restrict the display to the entries for the specified IPv6 prefix. You must specify the *ipv6-prefix* parameter in hexadecimal using 16-bit values between colons as documented in RFC 2373. You must specify the *prefix-length* parameter as a decimal value. A slash mark (/) must follow the *ipv6-prefix* parameter and precede the *prefix-length* parameter.

The**bgp** keyword restricts the display to entries for BGP4+ routes.

The**connect** keyword restricts the display to entries for directly connected interface IPv6 routes.

The**isis** keyword restricts the display to entries for IPv6 IS-IS routes.

The **ospf** keyword restricts the display to entries for OSPFv3 routes.

The**rip** keyword restricts the display to entries for RIPng routes.

The**static** keyword restricts the display to entries for static IPv6 routes.

The**summary** keyword displays a summary of the prefixes and different route types.

The **tags** keyword displays the label information for the IPv6 routes.

The **nexthop** option displays the next-hop information for all next hops in the routing table or for a specific entry. The *nexthop_id* parameter is a specific nexthop entry from the next hop table.

The **ref-routes** option allows you to display IPv6 routes in the forwarding table that refer to the specified nexthop entry.

The following table lists the information displayed by the**show ipv6 route** command.

**TABLE 26** IPv6 route table fields

| This field... | Displays... |
|---|---|
| Number of entries | The number of entries in the IPv6 route table. |
| Type | The route type, which can be one of the following:<br>• B - The route is learned from BGP4+.<br>• C - The destination is directly connected to the device.<br>• I - The route is learned from IPv6 IS-IS.<br>• L - The route is the host address of a loopback interface that is assigned an ipv6 address.<br>• O - The route is learned from OSPFv3.<br>• R - The route is learned from RIPng.<br>• S - The route is a static route. |
| OSPF Type | • i - an internal route calculated by OSPF.<br>• 1 - An OSPF type 1 external route.<br>• 2 - An OSPF type 2 external route.<br>• e - an external route calculated by OSPF. |
| IPv6 Prefix | The destination network of the route. |
| Next-Hop Router | The next-hop device. |
| Interface | The interface through which this device sends packets to reach the route destination. |
| Dis/Metric | The administrative distance and metric value of the route. |

To display a summary of the IPv6 route table, enter the following command at any CLI level.

```
device# show ipv6 route summary
IPv6 Routing Table - 7 entries:
  4 connected, 2 static, 0 RIP, 1 OSPF, 0 BGP
  Number of prefixes:
  /16: 1 /32: 1 /64: 3 /128: 2
```

Table 27 lists the information displayed by the **show ipv6 route summary** command.

**TABLE 27** IPv6 route table summary fields

| This field... | Displays... |
|---|---|
| Number of entries | The number of entries in the IPv6 route table. |
| Number of route types | The number of entries for each route type. |
| Number of prefixes | A summary of prefixes in the IPv6 route table, sorted by prefix length. |

To display the label information for the IPv6 route, enter the following command.

```
device# show ipv6 route tags
IPv6 Routing Table - 4 entries:
Type Codes - B:BGP C:Connected I:ISIS L:Local O:OSPF R:RIP S:Static
BGP  Codes - i:iBGP e:eBGP
ISIS Codes - L1:Level-1 L2:Level-2
OSPF Codes - i:Inter Area 1:External Type 1 2:External Type 2
Type IPv6 Prefix           Next Hop Router    Interface    Dis/Metric     Uptime
S    2001:db8:1::/64       2001:db8:1111::2   eth 1/1      1/1            1h3m
label information: 794624(IN)
Bi   2001:db8:2::/64       ::                 lsp toPE-4   200/1          30m20s
```

```
label information: 794624(OUT)
C    2001:db8:1111::/64    ::                eth 1/1      0/0           1h4m
label information: 794624(IN)
Bi   2001:db8:2222::/64    ::                lsp toPE-4   200/0         30m20s
label information: 794624(OUT)
```

The label information for the IPv6 route is shown in bold text in the previous output.

Table 28 describes the output parameters of the**show ipv6 route tags** command.

**TABLE 28** Output parameters of the **show ipv6 route tags** command

| Field | Description |
|---|---|
| Number of entries | Shows the number of entries in the IPv6 route table. |
| Type Codes | Shows the route type, which can be one of the following:<br>• B - The route is learned from BGP4+.<br>• C - The destination is directly connected to the router.<br>• I - The route is learned from IPv6 IS-IS.<br>• L - The route is the host address of a loopback interface that is assigned an IPv6 address. |
| Type Codes<br>(continued) | • O - The route is learned from OSPFv3.<br>• R - The route is learned from RIPng.<br>• S - The route is a static route. |
| BGP Codes | Shows the BGP type, which can be one of the following:<br>• i - An IBGP route.<br>• e - An EBGP route. |
| ISIS Codes | Shows the IS-IS type, which can be one of the following:<br>• L1 - An IS-IS level 1 route.<br>• L2 - An IS-IS level 2 route. |
| OSPF Codes | Shows the OSPF type, which can be one of the following:<br>• i - An internal route calculated by OSPF.<br>• 1 - An OSPF type 1 external route.<br>• 2 - An OSPF type 2 external route.<br>• e - An external route calculated by OSPF. |
| IPv6 Prefix | Shows the destination network of the route. |
| Next Hop Router | Shows the address of the next hop router. |
| Interface | Shows the interface through which this router sends the IPv6 packets to reach the destination. |
| Dis/Metric | Shows the administrative distance and metric value of the IPv6 route. |
| Uptime | Shows the amount of time the interface has been running. |
| label information | Shows the label information for the IPv6 route. |

## *Using the nexthop option*

You can display nexthop information for all next hops in the routing table or for a specific entry. To display all the nexthop entries, use the **show ipv6 route nexthop** command, and then use the option to display the next hop for a specific table entry.

```
device# show ipv6 route nexthop
Total number of IPv6 nexthop entries: 261; Forwarding Use: 259
       NextHopIp        Port             RefCount      ID      Age
1      ::               eth 1/2          1/1           1       973
2      ::               drop             1/1           65536   1013
5      ::               ve 257           1/1           898     973
6      ::               ve 279           1/1           920     973
```

```
7      ::              ve 299          1/1           940     973
8     192::1           eth 1/2      255959/255960    65538   1109
...
```

**Syntax: show ipv6 route nexthop** *nexthop_id*

The *nexthop_id* is under the column labeled ID in the output of the **show ip route nexthop** command. In the following example, the output of the **show ip route nexthop** command is displayed for a nexthop ID 65538.

```
device# show ipv6 route nexthop 65538
       NextHopIp        Port          RefCount        ID      Age
1      192::1            eth 1/2      255950/255951   65538   1384
```

## *Displaying IPv6 routes with nexthop ID*

By using the **nexthop** option with the **ref-routes** keyword, you can display IPv6 routes in the forwarding table that refer to a specified nexthop entry, as the following example illustrates (using nexthop ID 65538).

```
device#show ipv6 route nexthop 65538 ref-routes
Type Codes - B:BGP D:Connected I:ISIS O:OSPF R:RIP S:Static; Cost - Dist/Metric
ISIS Codes - L1:Level-1 L2:Level-2
OSPF Codes - i:Inter Area 1:External Type 1 2:External Type 2 s:Sham Link
     Destination      Gateway        Port       Cost     Type   Uptime
1    3::/64           300:300::2     eth 1/2    20/0     B      15m27s
2    4::/64           300:300::2     eth 1/2    20/0     B      18m17s
3    4:21:103::0/126  300:300::2     eth 1/2    20/0     B      15m48s
4    4:23:112::0/126  300:300::2     eth 1/2    20/0     B      19m12s
5    4:23:113::0/126  300:300::2     eth 1/2    20/0     B      19m12s
6    4:23:114::0/126  300:300::2     eth 1/2    20/0     B      19m12s
```

**Syntax: show ipv6 route nexthop** *nexthop_id* **ref-routes**

### Description of command output fields

The following table lists the information in the **show ipv6 route** command output when you run the **show ipv6 route nexthop***nexthop_id***ref-routes** command.

This display shows the following information.

**TABLE 29** show ipv6 route nexthop ref-routes information fields

| This field... | Displays... |
|---|---|
| *Destination* | The destination network of the IPv6 route. |
| Gateway | The next-hop router. |
| Port | The port through which this device sends packets to reach the route's destination. |
| Cost | The route's cost. |
| Type | The route type, which can be one of the following:<br>• B - The route was learned from BGP.<br>• D - The destination is directly connected to this device.<br>• I - The route is an ISIS route.<br>• O - The route is an OSPF route.<br>• R - The route was learned from RIP.<br>• S - The route is a static route.<br>• * - The route is a candidate default route. |
| Uptime | The amount of time since the route was last modified. The format of this display parameter may change depending upon the age of the route to |

**TABLE 29** show ipv6 route nexthop ref-routes information fields (continued)

| This field... | Displays... |
|---|---|
| | include the seconds (s), minutes (m), hours (h), and days (d), as described in the following:<br>• 400d - Only days (d) displayed<br>• 20d23h - days (d) and hours (h) displayed<br>• 14h33m - hours (h) and minutes (m) displayed<br>• 10m59s - minutes (m) and seconds (s) displayed |

### *Displaying IPv6 routes using the detail option*

By using the **detail** option with the **show ipv6 route** command, you can display the nexthop entry and the reference count. The following command output is displayed for a nexthop ID 65538.

```
device#show ipv6 route nexthop 65538 ref-routes
Type Codes - B:BGP D:Connected I:ISIS O:OSPF R:RIP S:Static; Cost - Dist/Metric
ISIS Codes - L1:Level-1 L2:Level-2
OSPF Codes - i:Inter Area 1:External Type 1 2:External Type 2 s:Sham Link
     Destination      Gateway        Port       Cost     Type   Uptime
1    3::/64           300:300::2     eth 1/2    20/0     B      15m27s
2    4::/64           300:300::2     eth 1/2    20/0     B      18m17s
3    4:21:103::0/126  300:300::2     eth 1/2    20/0     B      15m48s
4    4:23:112::0/126  300:300::2     eth 1/2    20/0     B      19m12s
Nexthop Entry ID:65538, Paths: 1, Ref_Count:256001/256002
```

**Syntax: show ipv6 route** *specific-route* **detail**

# Displaying local IPv6 devices

The device can function as an IPv6 host, if you configure IPv6 addresses on the interfaces but do not enable IPv6 routing using the **ipv6 unicast-routing** command.

From the IPv6 host, you can display information about IPv6 devices to which the host is connected. The host learns about the devices through their router advertisement messages. To display information about the IPv6 devices connected to an IPv6 host, enter the following command at any CLI level.

```
device# show ipv6 router
Router fe80::2e0:80ff:fe46:3431 on Ethernet 50, last update 0 min
Hops 64, Lifetime 1800 sec
Reachable time 0 msec, Retransmit time 0 msec
```

**Syntax: show ipv6 router**

If you configure your device to function as an IPv6 device (configure IPv6 addresses on the interfaces and enable IPv6 routing using the **ipv6 unicast-routing** command) and then enter the **show ipv6 router command** , you will receive the following output.

```
No IPv6 router in table
```

Meaningful output for this command is generated for devices configured to function as IPv6 hosts only.

This display shows the following information.

**TABLE 30** IPv6 local router information fields

| This field... | Displays... |
|---|---|
| Router *ipv6 address* on *interface port* | The IPv6 address for a particular interface. |
| Last update | The amount of elapsed time (in minutes) between the current and previous updates received from a device. |

**TABLE 30** IPv6 local router information fields (continued)

| This field... | Displays... |
|---|---|
| Hops | The default value that should be included in the Hop Count field of the IPv6 header for outgoing IPv6 packets. The hops value applies to the device for which you are displaying information and should be followed by IPv6 hosts attached to the device. A value of 0 indicates that the device leaves this field unspecified. |
| Lifetime | The amount of time (in seconds) that the device is useful as the default device. |
| Reachable time | The amount of time (in milliseconds) that a device assumes a neighbor is reachable after receiving a reachability confirmation. The reachable time value applies to the device for which you are displaying information and should be followed by IPv6 hosts attached to the device. A value of 0 indicates that the device leaves this field unspecified. |
| Retransmit time | The amount of time (in milliseconds) between retransmissions of neighbor solicitation messages. The retransmit time value applies to the device for which you are displaying information and should be followed by IPv6 hosts attached to the device. A value of 0 indicates that the device leaves this field unspecified. |

# Displaying IPv6 TCP information

You can display the following IPv6 TCP information:

- General information about each TCP connection on the device, including the percentage of free memory for each of the internal TCP buffers.
- Detailed information about a specified TCP connection.

To display general information about each TCP connection on the device, enter the following command at any CLI level.

```
device# show ipv6 tcp connections
Local IP address:port <-> Remote IP address:port TCP state
192.168.182.110:23 <->    192.168.8.186:4933     ESTABLISHED
192.168.182.110:8218 <->  192.168.182.106:179    ESTABLISHED
192.168.182.110:8039 <->  192.168.2.119:179      SYN-SENT
192.168.182.110:8159 <->  192.168.2.102:179      SYN-SENT
2001:db8::110:179 <->     2001:db8::106:8222       ESTABLISHED (1440)
Total 5 TCP connections
TCP MEMORY USAGE PERCENTAGE
FREE TCB = 98 percent
FREE TCP QUEUE BUFFER = 99 percent
FREE TCP SEND BUFFER = 97 percent
FREE TCP RECEIVE BUFFER = 100 percent
FREE TCP OUT OF SEQUENCE BUFFER = 100 percent
```

**Syntax: show ipv6 tcp connections**

This display shows the following information.

**TABLE 31** General IPv6 TCP connection fields

| This field... | Displays... |
|---|---|
| Local IP address:port | The IPv4 or IPv6 address and port number of the local interface over which the TCP connection occurs. |
| Remote IP address:port | The IPv4 or IPv6 address and port number of the remote interface over which the TCP connection occurs. |
| TCP state | The state of the TCP connection. Possible states include the following:<br>• LISTEN - Waiting for a connection request. |

**TABLE 31** General IPv6 TCP connection fields (continued)

| This field... | Displays... |
|---|---|
| | • SYN-SENT - Waiting for a matching connection request after having sent a connection request.<br>• SYN-RECEIVED - Waiting for a confirming connection request acknowledgment after having both received and sent a connection request.<br>• ESTABLISHED - Data can be sent and received over the connection. This is the normal operational state.<br>• FIN-WAIT-1 - Waiting for a connection termination request from the remote TCP, or an acknowledgment of the connection termination request previously sent.<br>• FIN-WAIT-2 - Waiting for a connection termination request from the remote TCP.<br>• CLOSE-WAIT - Waiting for a connection termination request from the local user.<br>• CLOSING - Waiting for a connection termination request acknowledgment from the remote TCP.<br>• LAST-ACK - Waiting for an acknowledgment of the connection termination request previously sent to the remote TCP (which includes an acknowledgment of its connection termination request).<br>• TIME-WAIT - Waiting for enough time to pass to be sure the remote TCP received the acknowledgment of its connection termination request.<br>• CLOSED - There is no connection state. |
| FREE TCB = *percentage* | The percentage of free TCP control block (TCB) space. |
| FREE TCB QUEUE BUFFER = *percentage* | The percentage of free TCB queue buffer space. |
| FREE TCB SEND BUFFER = *percentage* | The percentage of free TCB send buffer space. |
| FREE TCB RECEIVE BUFFER = *percentage* | The percentage of free TCB receive buffer space. |
| FREE TCB OUT OF SEQUENCE BUFFER = *percentage* | The percentage of free TCB out of sequence buffer space. |

To display detailed information about a specified TCP connection, enter a command such as the following at any CLI level.

```
device# show ipv6 tcp status 2001:db8::110 179 2001:db8::106 8222
TCP: TCB = 0x217fc300
TCP: 2001:db8::110:179 <-> 2001:db8::106:8222: state: ESTABLISHED Port: 1
  Send: initial sequence number = 242365900
  Send: first unacknowledged sequence number = 242434080
  Send: current send pointer = 242434080
  Send: next sequence number to send = 242434080
  Send: remote received window = 16384
  Send: total unacknowledged sequence number = 0
  Send: total used buffers 0
  Receive: initial incoming sequence number = 740437769
  Receive: expected incoming sequence number = 740507227
  Receive: received window = 16384
  Receive: bytes in receive queue = 0
  Receive: congestion window = 1459
```

**Syntax: show ipv6 tcp status** *local-ip-address local-port-number remote-ip-address remote-port-number*

The *local-ip-address* parameter can be the IPv4 or IPv6 address of the local interface over which the TCP connection is taking place.

The *local-port-number* parameter is the local port number over which a TCP connection is taking place.

The *remote-ip-address* parameter can be the IPv4 or IPv6 address of the remote interface over which the TCP connection is taking place.

The *remote-port-number* parameter is the local port number over which a TCP connection is taking place.

This display shows the following information.

**TABLE 32** Specific IPv6 TCP connection fields

| This field... | Displays... |
|---|---|
| TCB = *location* | The location of the TCB. |
| *local-ip-address local-port-number remote-ip-address remote-port-number state port* | This field provides a general summary of the following:<br>• The local IPv4 or IPv6 address and port number.<br>• The remote IPv4 or IPv6 address and port number.<br>• The state of the TCP connection. For information on possible states, refer to Table 31.<br>• The port numbers of the local interface. |
| Send: initial sequence number = *number* | The initial sequence number sent by the local device. |
| Send: first unacknowledged sequence number = *number* | The first unacknowledged sequence number sent by the local device. |
| Send: current send pointer = *number* | The current send pointer. |
| Send: next sequence number to send = *number* | The next sequence number sent by the local device. |
| Send: remote received window = *number* | The size of the remote received window. |
| Send: total unacknowledged sequence number = *number* | The total number of unacknowledged sequence numbers sent by the local device. |
| Send: total used buffers *number* | The total number of buffers used by the local device in setting up the TCP connection. |
| Receive: initial incoming sequence number = *number* | The initial incoming sequence number received by the local device. |
| Receive: expected incoming sequence number = *number* | The incoming sequence number expected by the local device. |
| Receive: received window = *number* | The size of the local device receive window. |
| Receive: bytes in receive queue = *number* | The number of bytes in the local device receive queue. |
| Receive: congestion window = *number* | The size of the local device receive congestion window. |

# Displaying IPv6 traffic statistics

To display IPv6 traffic statistics, enter the following command at any CLI level.

```
device# show ipv6 traffic
IP6 Statistics
  36947 received, 66818 sent, 0 forwarded, 36867 delivered, 0 rawout
  0 bad vers, 23 bad scope, 0 bad options, 0 too many hdr
  0 no route, 0 can't forward, 0 redirect sent, 0 source routed
  0 frag recv, 0 frag dropped, 0 frag timeout, 0 frag overflow
  0 reassembled, 0 fragmented, 0 ofragments, 0 can't frag
  0 too short, 0 too small, 11 not member
  0 no buffer, 66819 allocated, 21769 freed
  0 forward cache hit, 46 forward cache miss
ICMP6 Statistics
Received:
  0 dest unreach, 0 pkt too big, 0 time exceeded, 0 param prob
  2 echo req, 1 echo reply, 0 mem query, 0 mem report, 0 mem red
  0 router soli, 2393 router adv, 106 nei soli, 3700 nei adv, 0 redirect
  0 bad code, 0 too short, 0 bad checksum, 0 bad len
  0 reflect, 0 nd toomany opt, 0 badhopcount
Sent:
  0 dest unreach, 0 pkt too big, 0 time exceeded, 0 param prob
  1 echo req, 2 echo reply, 0 mem query, 0 mem report, 0 mem red
  0 router soli, 2423 router adv, 3754 nei soli, 102 nei adv, 0 redirect
  0 error, 0 can't send error, 0 too freq
Sent Errors:
  0 unreach no route, 0 admin, 0 beyond scope, 0 address, 0 no port
```

```
  0 source address policy, 0 reject route
  0 pkt too big, 0 time exceed transit, 0 time exceed reassembly
  0 param problem header, 0 nextheader, 0 option, 0 redirect, 0 unknown
UDP Statistics
  470 received, 7851 sent, 6 no port, 0 input errors
TCP Statistics
  57913 active opens, 0 passive opens, 57882 failed attempts
  159 active resets, 0 passive resets, 0 input errors
  565189 in segments, 618152 out segments, 171337 retransmission
```

**Syntax: show ipv6 traffic**

This display shows the following information.

**TABLE 33** IPv6 traffic statistics fields

| This field... | Displays... |
|---|---|
| **IPv6 statistics** | |
| received | The total number of IPv6 packets received by the device. |
| sent | The total number of IPv6 packets originated and sent by the device. |
| forwarded | The total number of IPv6 packets received by the Extreme device and forwarded to other devices. |
| delivered | The total number of IPv6 packets delivered to the upper layer protocol. |
| rawout | This information is used by Extreme Technical Support. |
| bad vers | The number of IPv6 packets dropped by the device because the version number is not 6. |
| bad scope | The number of IPv6 packets dropped by the device because of a bad address scope. |
| bad options | The number of IPv6 packets dropped by the device because of bad options. |
| too many hdr | The number of IPv6 packets dropped by the device because the packets had too many headers. |
| no route | The number of IPv6 packets dropped by the device because there was no route. |
| can not forward | The number of IPv6 packets the device could not forward to another device. |
| redirect sent | This information is used by Extreme Technical Support. |
| source routed | The number of IPv6 source-routed packets dropped. |
| frag recv | The number of fragments received by the device. |
| frag dropped | The number of fragments dropped by the device. |
| frag timeout | The number of fragment timeouts that occurred. |
| frag overflow | The number of fragment overflows that occurred. |
| reassembled | The number of fragmented IPv6 packets that the device reassembled. |
| fragmented | The number of IPv6 packets fragmented by the device to accommodate the MTU of this device or of another device. |
| ofragments | The number of output fragments generated by the device. |
| can not frag | The number of IPv6 packets the device could not fragment. |
| too short | The number of IPv6 packets dropped because they are too short. |
| too small | The number of IPv6 packets dropped because they do not have enough data. |
| not member | The number of IPv6 packets dropped because the recipient is not a member of a multicast group. |

**TABLE 33** IPv6 traffic statistics fields (continued)

| This field... | Displays... |
|---|---|
| no buffer | The number of IPv6 packets dropped because there is no buffer available. |
| forward cache miss | The number of IPv6 packets received for which there is no corresponding cache entry. |
| **ICMP6 statistics**<br>Some ICMP statistics apply to both Received and Sent, some apply to Received only, some apply to Sent only, and some apply to Sent Errors only. | |
| **Applies to Received and Sent** | |
| dest unreach | The number of Destination Unreachable messages sent or received by the device. |
| pkt too big | The number of Packet Too Big messages sent or received by the device. |
| time exceeded | The number of Time Exceeded messages sent or received by the device. |
| param prob | The number of Parameter Problem messages sent or received by the device. |
| echo req | The number of Echo Request messages sent or received by the device. |
| echo reply | The number of Echo Reply messages sent or received by the device. |
| mem query | The number of Group Membership Query messages sent or received by the device. |
| mem report | The number of Membership Report messages sent or received by the device. |
| mem red | The number of Membership Reduction messages sent or received by the device. |
| router soli | The number of Router Solicitation messages sent or received by the device. |
| router adv | The number of Router Advertisement messages sent or received by the device. |
| nei soli | The number of Neighbor Solicitation messages sent or received by the device. |
| nei adv | The number of Router Advertisement messages sent or received by the device. |
| redirect | The number of redirect messages sent or received by the device. |
| **Applies to Received Only** | |
| bad code | The number of Bad Code messages received by the device. |
| too short | The number of Too Short messages received by the device. |
| bad checksum | The number of Bad Checksum messages received by the Extreme device. |
| bad len | The number of Bad Length messages received by the device. |
| nd toomany opt | The number of Neighbor Discovery Too Many Options messages received by the device. |
| badhopcount | The number of Bad Hop Count messages received by the device. |
| **Applies to Sent Only** | |
| error | The number of Error messages sent by the device. |
| can not send error | The number of times the device encountered errors in ICMP error messages. |
| too freq | The number of times the device has exceeded the frequency of sending error messages. |
| **Applies to Sent Errors Only** | |
| unreach no route | The number of Unreachable No Route errors sent by the device. |

**TABLE 33** IPv6 traffic statistics fields (continued)

| This field... | Displays... |
|---|---|
| admin | The number of Admin errors sent by the device. |
| beyond scope | The number of Beyond Scope errors sent by the device. |
| address | The number of Address errors sent by the device. |
| no port | The number of No Port errors sent by the device. |
| pkt too big | The number of Packet Too Big errors sent by the device. |
| source address policy | The number of ICMPv6 destination unreachable messages sent with code 5 because an IPv6 packet is dropped by an Access Control policy and the IPv6 source address of a packet matches the source address filtering policy. |
| reject route | The number of ICMPv6 destination unreachable messages sent code 6 because an IPv6 packet is dropped due to the destination address in the packet matching a route that has been configured to drop the packet. |
| time exceed transit | The number of Time Exceed Transit errors sent by the device. |
| time exceed reassembly | The number of Time Exceed Reassembly errors sent by the device. |
| param problem header | The number of Parameter Problem Header errors sent by the device. |
| nextheader | The number of Next Header errors sent by the device. |
| option | The number of Option errors sent by the device. |
| redirect | The number of Redirect errors sent by the device. |
| unknown | The number of Unknown errors sent by the device. |
| **UDP statistics** | |
| received | The number of UDP packets received by the device. |
| sent | The number of UDP packets sent by the device. |
| no port | The number of UDP packets dropped because the packet did not contain a valid UDP port number. |
| input errors | This information is used by Extreme Technical Support. |
| **TCP statistics** | |
| active opens | The number of TCP connections opened by the device by sending a TCP SYN to another device. |
| passive opens | The number of TCP connections opened by the device in response to connection requests (TCP SYNs) received from other devices. |
| failed attempts | This information is used by Extreme Technical Support. |
| active resets | The number of TCP connections the device reset by sending a TCP RESET message to the device at the other end of the connection. |
| passive resets | The number of TCP connections the device reset because the device at the other end of the connection sent a TCP RESET message. |
| input errors | This information is used by Extreme Technical Support. |
| in segments | The number of TCP segments received by the device. |
| out segments | The number of TCP segments sent by the device. |
| retransmission | The number of segments that the device retransmitted because the retransmission timer for the segment had expired before the device at the other end of the connection had acknowledged receipt of the segment. |

# Displaying statistics for IPv6 subnet rate limiting

Run the **show rate-limit ipv6 subnet** and **show rate-limit protocol** commands to display information about IPv6 rate limiting.

```
device# show rate-limit ipv6 subnet
  Fwd:        252                    Drop:  155 bytes
  Re-mark:    0                      Total: 407 bytes
```

**Syntax: show rate-limit ipv6-subnet**

Table 34 describes the fields from the output of **show rate-limit ipv6 subnet** command.

**TABLE 34** Output from the **show rate-limit ipv6 subnet** command

| Field | Description |
|---|---|
| Fwd | IPv6 traffic that has been forwarded after the device was started or the counter was reset due to the rate limit policy. |
| Drop | IPv6 traffic that has been dropped after the device was started or the counter was reset due to the rate limit policy. |
| Re-mark | The number of IPv6 packets whose priority has been remarked as a result of exceeding the bandwidth available in the CIR bucket for the specific rate limit policy. |
| Total | IPv6 traffic that has been carried on the interface after the device was started or the counter was reset due to the rate limit policy. |

```
device# show rate-limit protocol
Index               0
In use              TRUE
Protocol            0 (arp)
Policy Map          abc
Index               2
In use              TRUE
Protocol            2 (ipv6 subnet)
Policy Map          abc
```

**Syntax: show rate-limit protocol**

Table 35 describes the fields from the output of **show rate-limit protocol** command.

**TABLE 35** Output from the **show rate-limit protocol** command

| Field | Description |
|---|---|
| Index | Numeric index of the protocol supported by the device. |
| In use | Whether the protocol is in use or not (True: In use / False: Not in use) |
| Protocol | Protocol name (0: arp / 2: IPv6 subnet) |
| Policy Map | The rate limit policy applied on this protocol. |

# Displaying IPv6 information for Router Advertisement Options

Run the **show ipv6** command to display IPv6 information about the newly configured DNS recursive server addresses, domain name suffixes, and the corresponding lifetime values on an IPv6 host network.

```
device# show ipv6
Global Settings
  IPv6 Router-Id: 2.2.2.1   load-sharing path: 4
  unicast-routing enabled, ipv6  allowed to run, hop-limit 64
  reverse-path-check disabled
  host drop cam limit disabled
  urpf-exclude-default disabled
```

```
  session-logging-age 5
  No Inbound Access List Set
  No Outbound Access List Set
  source-route disabled, forward-source-route disabled, icmp-redirect disabled
OSPF (default VRF): enabled
BGP: enabled, 1 active neighbor(s) configured
  ND6 RA DNS Attributes
    ipv6 nd ra-dns-server abcd:abcd:abcd::3 lifetime 122
    ipv6 nd ra-dns-server 1::1 lifetime 150
    ipv6 nd ra-dns-server abcd:abcd:abcd::2 lifetime 196
    ipv6 nd ra-dns-server abcd:abcd:abcd::1 lifetime 200
    ipv6 nd ra-domain-name extreme.com.abc.123.abbbc lifetime 102
    ipv6 nd ra-domain-name abc-011223.extreme.com lifetime 141
    ipv6 nd ra-domain-name abc.com lifetime 155
    ipv6 nd ra-domain-name abcd.com.abc.123 lifetime 200
device#
```

**Syntax: show ipv6**

# Displaying IPv6 interface information for Router Advertisement Options

Run the **show ipv6 interface** command to display IPv6 interface information about the newly configured DNS recursive server addresses, domain name suffixes, and the corresponding lifetime values on an IPv6 host network.

```
device# show ipv6 interface ethernet 2/1
Interface Ethernet 2/1 is up, line protocol is up
  IPv6 is enabled, link-local address is fe80::224:38ff:fe90:e430 [Preferred]
  Global unicast address(es):
    7:7:7::1 [Preferred],  subnet is 7:7:7::/64
    7:7:7:: [Anycast],  subnet is 7:7:7::/64
  Joined group address(es):
    ff02::1:ff00:1
    ff02::1:ff00:0
    ff02::1:ff90:e430
    ff02::2
    ff02::1
  Port belongs to VRF: default-vrf
  MTU is 1500 bytes
  ICMP redirects are disabled
  ND DAD is enabled, number of DAD attempts: 3
  ND reachable time is 30000 Milliseconds
  ND advertised reachable time is 0 seconds
  ND retransmit interval is 1000 milliseconds
  ND advertised retransmit interval is 0 milliseconds
  ND next router advertisement will be sent in 258 seconds
  ND router advertisements live for 1800 seconds
  No Inbound Access List Set
  No Outbound Access List Set
  IPv6 RPF mode: None  IPv6 RPF Log: Disabled
  RxPkts:       0                     TxPkts:   63
  RxBytes:      0                     TxBytes:  12010
  IPv6 unicast RPF drop: 0
  IPv6 unicast RPF suppressed drop: 0
  ND6 RA DNS Attributes
    ipv6 nd ra-dns-server 11::1176 lifetime 176
    ipv6 nd ra-dns-server 11::11 lifetime 200
    ipv6 nd ra-domain-name abc.com.abb lifetime 150
    ipv6 nd ra-domain-name abc-aaa.com lifetime 199
    ipv6 nd ra-domain-name abc.com lifetime 200
device#
```

**Syntax: show ipv6 interface**

# IPv4 Static Routing



## Overview of static routing

Static routes are manually configured entries in the IP routing table.

The IP route table can receive routes from several sources, including static routes. Other route sources include directly connected networks, RIP, OSPF, and BGP4 protocols.

Static routes can be used to specify desired routes, backup routes, or routes of last resort. Static routing can help provide load balancing and can use routing information learned from other protocols.

In setting up static routes, you can specify several types of destinations:

- Destination network, using an IP address and network mask or prefix length
- Default network route
- Next-hop router
- Next-hop network protocol type
- Ethernet interface, typically used for directly attached destination networks
- Virtual interface
- Null interface

You can influence the preference a route is given in the following ways:

- By setting a route metric higher than the default metric
- By giving the route an administrative distance
- By specifying a route tag for use with a route map.

Static routes can be configured to serve as any of the following:

- Default routes
- Primary routes
- Backup routes
- Null routes for intentionally dropping traffic when the desired connection fails

- Alternative routes to the same destination to help load balance traffic.

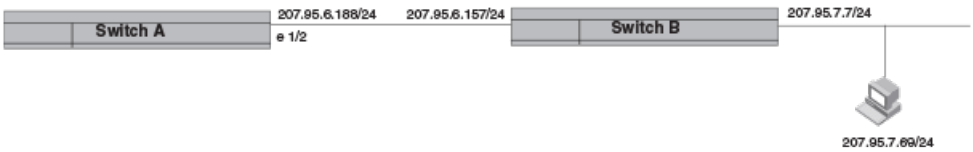
## Static route states follow port states

IP static routes remain in the IP route table only as long as the port or virtual interface used by the route is available and the next-hop IP address is valid; otherwise, the software removes the static route from the IP route table. If the port or virtual routing interface becomes available again later and the next-hop is valid, the software adds the route back to the route table.

This feature allows the router to adjust to changes in network topology. The router does not continue trying to use routes on unavailable paths but instead uses routes only when their paths are available.

In the following example, a static route is configured on Switch A. The route configuration is shown following the figure.

**FIGURE 19** Example of static route




The following command configures a static route to 207.95.7.0 destinations, using 207.95.6.157 as the next-hop gateway.

```
device(config)# ip route 207.95.7.0/24 207.95.6.157
```

When you configure a static IP route, you specify the destination address for the route and the next-hop gateway or Layer 3 interface through which the Layer 3 device can reach the route. The device adds the route to the IP route table. In this case, Switch A knows that 207.95.6.157 is reachable through port 1/2, and also assumes that local interfaces within that subnet are on the same port. Switch A deduces that IP interface 207.95.7.7 is also on port 1/2.

The software automatically removes a static IP route from the IP route table if the port used by that route becomes unavailable or the IP address is not valid. When the port becomes available again, the software automatically re-adds the route to the IP route table.

# Configuring a basic IP static route

To configure a basic IP static route, perform these steps.

1. Enter global configuration mode.

   ```
   device# configure terminal
   ```

2. Enter the IP address and prefix length, or enter the IP address and network mask for the route destination network. On the same command line, enter the IP address for the next hop.

   ```
   device(config)# ip route 10.0.0.0 255.0.0.0 10.1.1.1
   ```

   This example configures an IP static route with a destination network address of 10.0.0.0, a destination mask of 255.0.0.0, and a next hop address of 10.1.1.1.

   NOTE
   In the network mask, "1's" are significant bits, and "0's" allow any value. The mask 255.255.255.0 matches all hosts within the Class C subnet address specified in the destination IP address. You can use "/24" as the equivalent address prefix.

The following example configures an IP static route with a destination network address of 10.0.0.0, a prefix length of 24 bits, and a next hop address of 10.1.1.1.

```
device# configure terminal
device(config)# ip route 10.0.0.0/24 10.1.1.1
```

# Adding metrics to a static route

You can influence route preference by adding a cost metric or an administrative distance to a static route.

Follow these steps to create an IP static route with cost metrics.

1. Enter global configuration mode.

```
device# configure terminal
```

2. Designate the route destination and next hop, and add a route priority parameter.

| Option | Description |
|---|---|
| Cost metric | The value is compared to the metric for other static routes in the IPv4 route table to the same destination. Two or more routes to the same destination with the same metric load share traffic to the destination. The value may range from 1 through 16. The default is 1. A route with a cost of 16 is considered unreachable. |
| Administrative distance | This value is compared to the administrative distance of all routes to the same destination. Static routes by default take precedence over learned protocol routes. However, to give a static route a lower priority than a dynamic route, give the static route the higher administrative distance. The value is preceded by the keyword **distance** and can range from 1 to 255. The default is 1. A value of 255 is considered unreachable. |

```
device(config)# ip route 10.128.2.71/24 10.111.10.1 distance 10
```

This example configures a static route with an administrative distance of 10.

NOTE
The device replaces a static route if it receives a route to the same destination with a lower administrative distance.

```
device(config)# ip route 10.128.2.69/24 10.111.10.1 2
```

This example configures a static route with a metric of 2.

The following example configures a static route to destinations with an IP address beginning with 10.0.0.0. The route uses IP address 10.111.10.1 as the next hop. The static route is assigned an administrative distance of 3.

```
device# configure terminal
device(config)# ip route 10.0.0.0/24 10.111.10.1 distance 3
```

# Naming an IP static route

IPv4 static route names are optional and non-unique. You can give a group of static routes the same name to help identify them.

To configure an IP static route with a name, follow these steps.

1. Enter global configuration mode.

```
device# configure terminal
```

2. Enter the destination IP address and mask or prefix length followed by the IP address of the next hop. On the same command line, enter the keyword name followed by the identifying ASCII string.

```
device(config)# ip route 10.22.22.22 255.255.255.0 10.1.1.1 name corporate
```

The following example creates a static route to IP destination network addresses beginning with 10.22.22.22 through the next-hop address 10.1.1.1. The route is given the non-unique name "corporate."

```
device# configure terminal
device(config)# ip route 10.22.22.22 255.255.255.0 10.1.1.1 name corporate
```

# Removing a name or a static route

When an IP route has a name, the **no** form of the full **ip route** command removes the name. Use the **no** form of the command a second time to remove the route.

1. Enter configuration mode.

```
device# configure terminal
```

2. Enter **no ip route** followed by the full route designation.

```
device# configure terminal
device(config)# no ip route 10.22.22.22 255.255.255.255 10.1.1.1 name xyz
```

   This example removes only the name of the route.

3. If necessary, repeat the **no ip route** command with the full route designation.

```
device(config)# no ip route 10.22.22.22 255.255.255.255 10.1.1.1
```

   This example repeats the previous route. Because the route has no name, the command removes the designated static route.

The following example removes the name of the designated static route, removes the route, and saves the change to the IP routing table.

```
device# configure terminal
device(config)# no ip route 10.22.22.22 255.255.255.255 10.1.1.1 name xyz
device(config)# no ip route 10.22.22.22 255.255.255.255 10.1.1.1
```

# Configuring a physical interface as next hop

The interface you use for the static route's next hop must have at least one IP address configured on it. The address does not need to be in the same subnet as the destination network.

NOTE
You cannot add an interface-based static route to a network if there is already a static route of any type with the same metric you specify for the interface-based route.

NOTE
ARP will be generated for a forwarded packet destination IP address when an interface is configured as the next hop.

To configure an IP static route with an IP physical interface as the next hop, follow these steps.

1. Enter global configuration mode.

```
device# configure terminal
```

2. Enter the IP address and prefix-length, or enter the IP address and network mask for the route destination network. On the same command line, enter the keyword **ethernet** followed by the interface number to be used as next hop.

```
device(config)# ip route 10.128.2.69 255.255.255.0 ethernet 1/4
```

   This example configures an IP static route with a destination network address of 10.128.2.69, a network mask of 255.255.255.0, and Ethernet port 1/4 as the next hop.

# Configuring a virtual interface as next hop

The virtual interface you use for the static route's next hop must have at least one IP address configured on it. The address does not need to be in the same subnet as the destination network.

To configure an IP static route that uses a virtual interface as the next hop, follow these steps.

1. Enter global configuration mode.

```
device# configure terminal
```

2. Enter the IP destination address and the network mask or prefix-length. On the same command line, enter the keyword **ve** followed by the appropriate ID number.

```
device(config)# ip route 10.128.2.0 255.255.255.0 ve 3
```

The following example configures an IP static route with a destination address of 10.128.2.0, a prefix-length of /24, and a virtual interface (ve 3) as the next hop.

```
device# configure terminal
device(config)# ip route 10.128.2.0/24 ve 3
```

# Configuring an MPLS next hop for an IPv4 static route

You can set the next hop for a static route to the egress router of an LSP tunnel if the destination route is contained in the MPLS routing table.

The destination route must be contained in the MPLS routing table.

MPLS route resolution must be enabled for an LSP route to be active.

In this configuration, the static route is updated with the LSP routes and reverts to its original next-hop interface when this feature is disabled or when the LSP goes down. The route through the LSP can also be used as the default route.

To enable the MPLS next-hop feature for IPv4 static routes and configure routes that use the feature, follow these steps.

1. Enter global configuration mode.

```
device# configure terminal
```

2. Enter the following command.

```
device(config)# ip route next-hop-enable-mpls
```

This command enables static routes to use an LSP tunnel as next hop.

NOTE
Use the **no** form of the command to disable MPLS next hops for static routes.

3. Configure a static route with an LSP next hop.

```
device(config)# ip route 10.12.12.12/32 lsp-next-hop extreme1 distance 110 3
```

This example sets up an LSP tunnel as the next hop using its pre-configured name.

4. To verify the destination route, enter the **show ip route** command.

```
device(config)# exit
device# show ip route
Total number of IP routes: 6
Type Codes - B:BGP D:Connected I:ISIS S:Static R:RIP O:OSPF; Cost - Dist/Metric
        Destination        Gateway         Port           Cost           Type
1       10.47.6.0/24       DIRECT          mgmt 1         0/0            D
2       10.11.11.11/32     DIRECT          loopback 1     0/0            D
3       10.12.12.12/32     10.1.0.1        eth 5/1        110/3          O
        10.12.12.12/32     10.12.12.12     lsp extremel   110/3          O
        10.12.12.12/32     10.12.12.12     lsp t11        110/3          O <----
4       10.13.13.13/32     10.1.0.1        eth 5/1        110/2          O
5       10.1.0.0/24        DIRECT          eth 5/1        0/0            D
6       10.1.0.0/24        10.1.0.1        eth 5/1        110/2          O
device#
```

As shown in the example, there is a route to the destination network 10.12.12.12/32 through the gateway at IP address 10.12.12.12, which is on LSP 11 (lsp t11). The route has an administrative distance of 110 and a cost metric of 3.

5. To verify that an LSP is up and that MPLS has a route to it, enter the **show mpls lsp** and **show mpls route** commands as shown in the following examples.

```
device# show mpls lsp
Note: LSPs marked with * are taking a Secondary Path
                                Admin  Oper  Tunnel   Up/Dn  Retry   Active
Name            To              State State  Intf     Times  No.     Path
t6              10.12.12.13     UP    DOWN    --       0      292     --
t5              10.12.12.12     UP    UP     tnl3     1      0       --
extremel        10.12.12.12     UP    UP     tnl1     1      0       --     <----
t7              10.12.12.12     UP    UP     tnl5     1      0       one
extremelongl    10.12.12.12     UP    UP     tnl9     1      0
--
t4              10.12.12.12     UP    UP     tnl2     1      0       --
t1              10.12.12.12     UP    UP     tnl0     1      0       --
```

The following example enables MPLS as a next hop. It configures a static route to the destination network addresses 10.12.12.12/32 through the LSP named extreme1. The route has an administrative distance of 110 and a metric cost of 3.

```
device# configure terminal
device(config)# ip route next-hop-enable-mpls
device(config)# ip route 10.12.12.12/32 lsp-next-hop extreme1 distance 110 3
```

# Configuring a static route for use with a route map

You can configure a static route with a tag that can be referenced in a route map.

Perform these steps to configure a static route with a tag that can be referenced in a route map.

1. Enter global configuration mode.

```
device# configure terminal
```

2. Enter the **ip route** command followed by the destination network IP address and prefix-length and the next-hop IP address. On the same line, enter the keyword **tag** followed by a decimal tag number.

```
device(config)# ip route 10.0.0.0/24 10.1.1.1 tag 3
```

NOTE
An address mask may be used instead of the prefix-length (such as 255.255.255.0 instead of /24).

The following example creates an IP static route to destination IP addresses beginning with 10.0.0.0 through the next-hop address 10.1.1.1. The static route includes the tag "3" for later use in a route map.

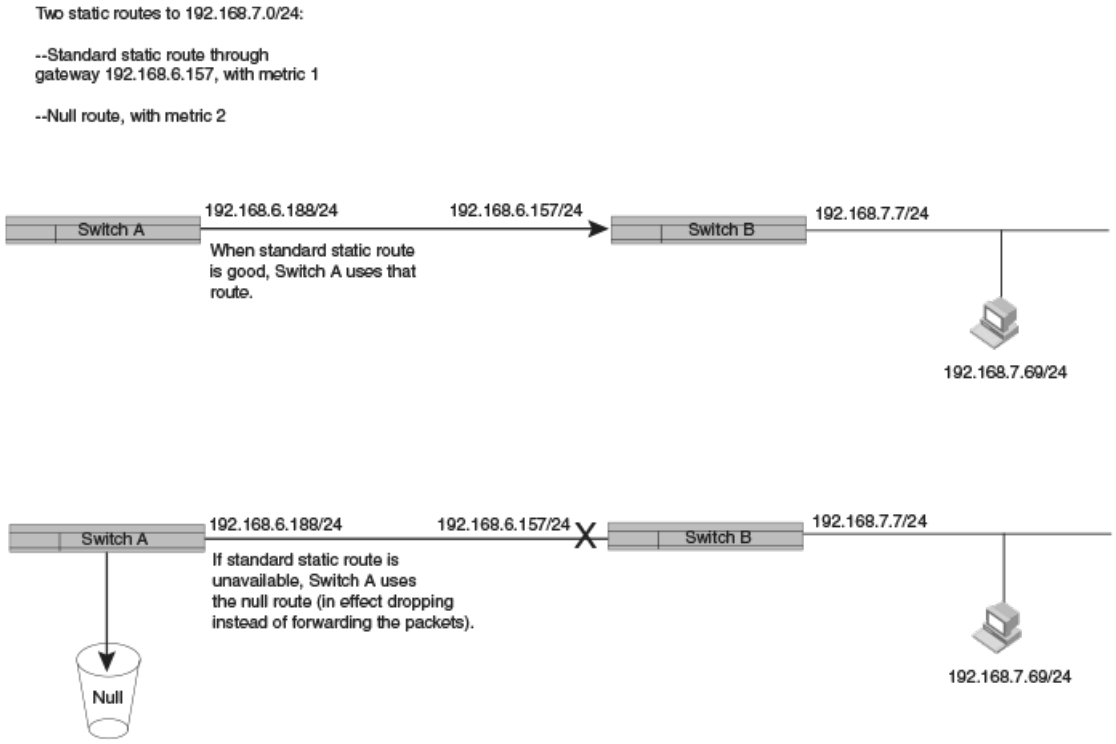
```
device# configure terminal
device(config)# ip route 10.0.0.0/24 10.1.1.1 tag 3
```

# Configuring a null route

You can configure a null static route to drop packets to a certain destination. This is useful when the traffic should not be forwarded if the preferred route is unavailable.

The following figure depicts how a null static route works with a standard route to the same destination.

**FIGURE 20** Null route and standard route to same destination




To configure a null route with a lower priority than the preferred route, perform the following steps.

1. NOTE
   You cannot add a null static route to a network if there is already a static route of any type with the same metric you specify for the null route.

   Enter global configuration mode.

   ```
   device# configure terminal
   ```

2. Configure the preferred route to a destination.

   ```
   device(config)# ip route 192.168.7.0/24 192.168.6.157
   ```

   This example creates a static route to destination network addresses that have an IP address beginning with 192.168.7.0. These destinations are routed through the next-hop gateway 192.168.6.157. The route carries the default metric of 1.

3. Configure the null route to the same destination with a higher metric.

   ```
   device(config)# ip route 192.168.7.0/24 null0 2
   ```

   This example creates a null static route to the same destination. The metric is set higher so that the preferred route is used if it is available. When the preferred route becomes unavailable, the null route is used, and traffic to the destination is dropped.

The following example creates a primary route to all destinations beginning with 192.168.7.0. It creates an alternative null route to drop the packets when the primary route is not available.

```
device# configure terminal
device(config)# ip route 192.168.7.0/24 192.168.6.157
device(config)# ip route 192.168.7.0/24 null0 2
```

## Dropping traffic to the null route in hardware

As an option, you can configure the device to drop packets sent to the null route in hardware.

You can program the CAM discard traffic sent to the null0 interface to improve forwarding efficiency and reduce the burden on the device CPU.

To assign the default route as the null route and drop packets in hardware, follow these steps.

1. Enter global configuration mode.

```
device# configure terminal
```

2. Configure the default route (0.0.0.0 0.0.0.0) as the null route.

```
device(config)# ip route 0.0.0.0 0.0.0.0 null0
```

3. Configure the hardware to drop packets for the default route.

```
device(config)# ip hw-drop-on-def-route
```

The following example configures the default IP route as the null route and specifies that packets to the default route be dropped in hardware.

```
device# configure terminal
device(config)# ip route 0.0.0.0 0.0.0.0 null0
device(config)# ip hw-drop-on-def-route
```

## Configuring CAM default route aggregation

You can enable the CAM Default Route Aggregation feature to prevent blocking CAM space.

Configuring the device to drop traffic sent to the default IP route address in hardware causes the device to program 32-bit host CAM entries for each destination address using the default route, which could consume the CAM space. To prevent this, enable the CAM Default Route Aggregation feature by entering the following commands.

1. Enter global configuration mode.

```
device# configure terminal
```

2. Enter the following command.

```
device(config)# ip dr-aggregate
```

The following example configures the null route as the default route, discards packets to the null route in hardware, and configures the CAM Default Aggregation feature to prevent consumption of CAM space.

```
device# configure terminal
device(config)# ip route 0.0.0.0 0.0.0.0 null0
device(config)# ip hw-drop-on-def-route
device(config)# ip dr-aggregate
```

# Configuring a default static route

You can manually create a default static route that the router uses if there are no other default routes to a destination.

If the default route is a protocol route, that protocol needs to be enabled to resolve static routes. Use the **ip route next-hop** command to allow protocol resolution through the default route.

If the default route itself is a static route, you must configure the **ip route next-hop-enable-default** command to resolve other static routes through the default route. You may also configure recursive lookup to resolve the next hop.

Perform these steps to configure a default route.

1. Enter global configuration mode.

```
device# configure terminal
```

2. Enter 0.0.0.0 0.0.0.0 as the destination route and network mask. On the same line, enter a valid next-hop address.

```
device(config)# ip route 0.0.0.0 0.0.0.0 10.24.4.1
```

   The example creates a default route through IP address 10.24.4.1.

3. (Optional) Enable the default network route for static route next-hop resolution.

```
device(config)# ip route next-hop-enable-default
```

   NOTE
   This command can be independently applied on a per-VRF basis.

4. (Optional) Configure next-hop recursive lookup to resolve the next-hop gateway.

```
device# configure terminal
device(config)# ip route next-hop-recursion
```

The following example configures static routing next-hop recursion to three levels (the default). It configures the network default static route through next-hop IP address 10.24.4.1 and allows the default route to resolve other static routes.

NOTE
You can specify a level of recursion up to 10.

```
device# configure terminal
device(config)# ip route next-hop-recursion
device(config)# ip route 0.0.0.0 0.0.0.0 10.24.4.1
device(config)# ip route next-hop-enable-default
```

# Resolving a static route using other static routes

You can use next-hop recursive lookup to resolve a static route.

Perform these steps to enable next-hop recursive lookup.

NOTE
This command can be independently applied on a per-VRF basis.

1. Enter global configuration mode.

```
device# configure terminal
```

2. Enter the following command, and, as an option, specify the level of recursion.

```
device(config)# ip route next-hop-recursion
```

   This example configures recursive static route lookup to three levels (the default).

The following example configures recursive static route lookup to five levels.

```
device# configure terminal
device(config)# ip route next-hop-recursion 5
```

# Resolving the next hop through a protocol

You can use routes from a specific protocol to resolve a configured static route.

Perform these steps to resolve the next hop for a static route using learned routes from a protocol.

NOTE
Connected routes are always used to resolve static routes.

1. Enter global configuration mode.

   ```
   device# configure terminal
   ```

2. Enable resolution through the desired protocol.

   These protocol options are available:

   - bgp
   - isis
   - ospf
   - rip

   ```
   device(config)# ip route next-hop ospf
   ```

   This example enables route resolution through OSPF.

   ```
   device(config)# ip route next-hop bgp
   ```

   This example resolves static routes through BGP. Both iBGP and eBGP routes are used to resolve the routes.

   ```
   device(config)# ip route next-hop isis
   ```

   This example resolves static routes through ISIS.

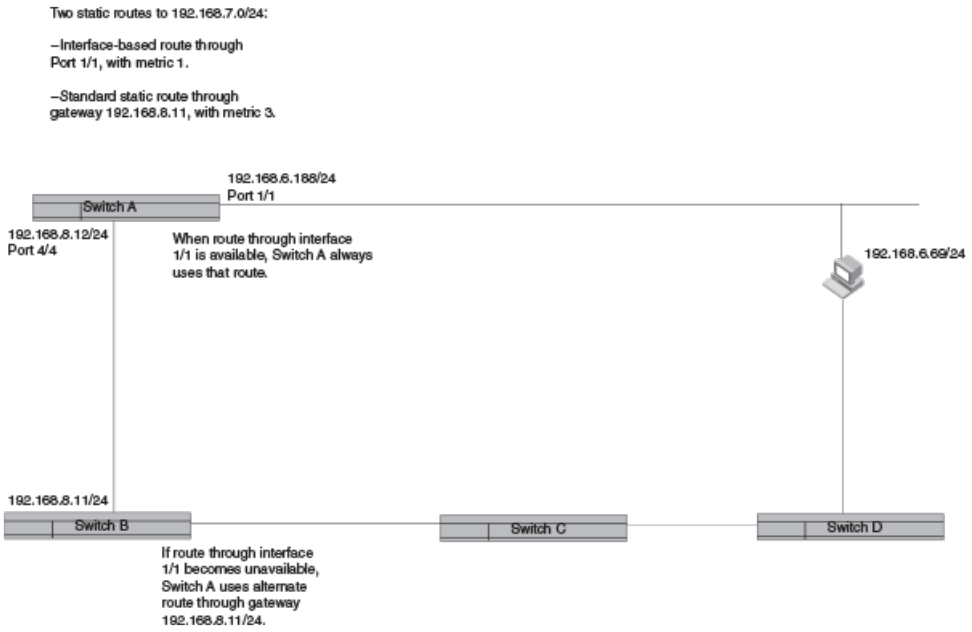
# Configuring load sharing and redundancy

You can configure multiple IP static routes to the same destination to set up load sharing or backup routes.

If you configure more than one static route to the same destination with different next-hop gateways but the same metrics, the router load balances among the routes using a basic round-robin method.

If you configure multiple static IP routes to the same destination with different next-hop gateways and different metrics, the router always uses the route with the lowest metric. If this route becomes unavailable, the router fails over to the static route with the next-lowest metric. The following figure depicts two routes with different metrics configured for the same destination.

FIGURE 21 Two static routes to same destination




To set up multiple routes for load sharing or redundancy, perform the following steps.

NOTE
You can also use administrative distance to set route priority; however, be sure to give a static route a lower administrative distance than other types of routes, unless you want other route types to be preferred over the static route.

1. Enter global configuration mode.

```
device# configure terminal
```

2. Enter multiple routes to the same destination using different next hops.

```
device(config)# ip route 10.128.2.0/24 10.157.22.1
device(config)# ip route 10.128.2.0/24 10.111.10.1
device(config)# ip route 10.128.2.0/24 10.1.1.1
```

This example creates three next-hop gateways to the destination. Traffic will alternate among the three paths through next-hop 10.157.22.1, next-hop 10.111.10.1, and next hop 10.1.1.1.

3. To prioritize the three routes, use different metrics for each of the three potential next hops.

```
device(config)# ip route 10.128.2.0/24 10.157.22.1
device(config)# ip route 10.128.2.0/24 10.111.10.1 2
device(config)# ip route 10.128.2.0/24 10.1.1.1 3
```

This example creates three alternate routes to the destination. The primary next hop is 10.157.22.1, which has the default metric of 1 (the default metric is not entered in the CLI). If this path is not available, traffic is directed to 10.111.10.1, which has the next lowest metric of 2. If the second path fails, traffic is directed to 10.1.1.1, which has a metric of 3.

# Displaying IPv4 static routes

You can check configured IPv4 routes, static routes, directly connected routes, routes configured for different protocols, the cost associated with each route, and the time the route has been available.

1. To display a list of active static routes and their connection times, at the device prompt, enter the **show ip route static** command.
2. To show all active IP routes and their connection times, enter the **show ip route** command.

The following example shows three configured routes. All routes have been active for over six days.

There are two configured IPv4 routes from the management port. The first is the default route, 0.0.0.0/0. This is a static route ("S") that uses the next-hop gateway 10.25.224.1. The default route has an administrative distance of 1 and a cost metric of 1. The other route is a direct route with no additional administrative or cost metric. A third route is a direct route configured as a loopback.

```
device# show ip route
Total number of IP routes: 3
Type Codes - B:BGP D:Connected I:ISIS O:OSPF R:RIP S:Static; Cost - Dist/Metric
BGP  Codes - i:iBGP e:eBGP
ISIS Codes - L1:Level-1 L2:Level-2
OSPF Codes - i:Inter Area 1:External Type 1 2:External Type 2 s:Sham Link
STATIC Codes - d:DHCPv6
        Destination        Gateway         Port           Cost          Type Uptime src-vrf
1       0.0.0.0/0          10.25.224.1     mgmt 1         1/1           S    6d1h   -
2       1.1.1.1/32         DIRECT          loopback 1     0/0           D    6d1h   -
3       10.25.224.0/24     DIRECT          mgmt 1         0/0           D    6d1h   -
```

# IPv6 Static Routing



## Overview of static routing

Static routes can be used to specify desired routes, backup routes, or routes of last resort. Static routing can help provide load balancing.

Static routes are manually configured entries in the existing IPv6 routing table. In setting up static routes, you can specify several types of destinations:

- Destination network, using an IP address and network mask or prefix length
- Default network route
- Next hop router
- Next hop tunnel gateway
- Next-hop network protocol type
- Ethernet interface, typically used for directly attached destination networks
- Virtual interface
- Null interface

You can influence the preference a route is given in the following ways:

- By setting a route metric higher than the default metric
- By giving the route an administrative distance

Static routes can be configured to serve as any of the following:

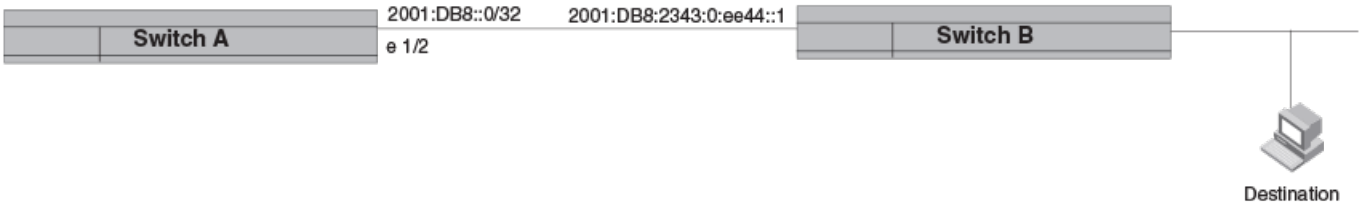
- Default routes
- Primary routes
- Backup routes
- Null routes for intentionally dropping traffic when the desired connection fails
- Alternative routes to the same destination to help load balance traffic.

## Static route states follow port states

IP static routes remain in the IP route table only as long as the port or virtual interface used by the route is available. If the port or virtual routing interface becomes unavailable, the software removes the static route from the IP route table. If the port or virtual routing interface becomes available again later, the software adds the route back to the route table. This feature allows the router to adjust to changes in network topology. The router does not continue trying to use routes on unavailable paths but instead uses routes only when their paths are available.

In the following example, a static route is configured on Switch A.

**FIGURE 22** Example of static route




The following command configures a static route to 2001:DB8::0/32, using 2001:DB8:2343:0:ee44::1 as the next-hop gateway.

```
device(config)# ipv6 route 2001:DB8::0/32 2001:DB8:2343:0:ee44::1
```

When you configure a static IP route, you specify the destination address for the route and the next-hop gateway or Layer 3 interface through which the Layer 3 device can reach the route. The device adds the route to the IP route table. In this case, Switch A knows that 2001:DB8:2343:0:ee44::1 is reachable through port 1/2, and also assumes that local interfaces within that subnet are on the same port. Switch A deduces that IP interface 2001:DB8::0/32 is also on port 1/2.

# Configuring a basic IPv6 static route

To configure a basic IPv6 static route, specify the IPv6 destination address, the address mask, and the IPv6 address of the next hop.

Before configuring a static IPv6 route, you must enable the forwarding of IPv6 traffic on the Layer 3 switch using the **ipv6 unicast-routing** command and enable IPv6 on at least one interface by configuring an IPv6 address or explicitly enabling IPv6 on that interface.

To configure a basic IPv6 static route, perform these steps.

1. Enter global configuration mode.

   ```
   device# configure terminal
   ```

2. Designate the route destination as an IPv6 address in hexadecimal with 16-bit values between colons, as specified in RFC 2373, and include the address prefix length preceded by a slash. On the same command line, enter the IPv6 address of the next-hop gateway.

   ```
   device(config)# ipv6 route 2001:DB8::0/32 2001:DB8:0:ee44::1
   ```

The following example configures an IPv6 static route for a destination network with the prefix 2001:DB8::0/32 and a next-hop gateway with the global address 2001:DB8:0:ee44::1

```
device# configure terminal
device(config)# ipv6 route 2001:DB8::0/32 2001:DB8:0:ee44::1
```

# Naming an IPv6 static route

A non-unique name can be applied to several IPv6 static routes to the same destination to help identify them.

Before configuring a static IPv6 route, you must enable the forwarding of IPv6 traffic on the Layer 3 switch using the **ipv6 unicast-routing** command and enable IPv6 on at least one interface by configuring an IPv6 address or explicitly enabling IPv6 on that interface.

An IPv6 static route can be given an optional name. Route names are not unique, and are therefore useful for identifying routes of a particular type, for example, all routes with a particular destination.

Follow these steps to name an IPv6 static route.

1. Enter global configuration mode.

```
device# configure terminal
```

2. Designate the route destination as and IPv6 address in hexadecimal with 16-bit values between colons, as specified in RFC 2373, and include the address prefix length preceded by a slash. On the same command line, enter the IPv6 address of the next-hop gateway. Still on the same command line, add the keyword **name** and the route name as an ASCII string.

```
device(config)# ipv6 route 2001:DB8::0/32 2001:DB8:0:ee44::1 name finance
```

The following example configures an IPv6 static route for a destination network with the prefix 2001:DB8::0/32 and a next-hop gateway with the global address 2001:DB8:0:ee44::1. The route has the name "finance." The name may be shared by other routes.

```
device# configure terminal
device(config)# ipv6 route 2001:DB8::0/32 2001:DB8:0:ee44::1 name finance
```

# Removing an IPv6 static route

Before configuring a static IPv6 route, you must enable the forwarding of IPv6 traffic on the Layer 3 switch using the **ipv6 unicast-routing** command and enable IPv6 on at least one interface by configuring an IPv6 address or explicitly enabling IPv6 on that interface.

The **no** form of the ipv6 route command must be entered with exact parameters to remove the command. If the route is configured in a non-default VRF, the **no** form of the ipv6 route command must be entered in VRF configuration mode.

Follow these steps to remove an IPv6 static route.

1. (Optional) To view configured routes and confirm exact parameters, enter the command **show ipv6 route** to display the IPv6 route table.

```
device# show ipv6 route
```

2. (Optional) Enter the **show ipv6 static route** command to narrow the output to static routes only.

```
device# show ipv6 static route
```

3. Enter global configuration mode.

```
device# configure terminal
```

4. Enter **no** followed by the **ipv6 route** command, including destination and next-hop, as shown in the following example. (You do not need to include cost metric, distance, or tag parameters.)

```
device(config)# no ipv6 route 2224::1/128 fe80::205:33ff:fee6:a501 ve 2
```

This example removes the IPv6 route specified.

The following example removes an existing IPv6 static route from the VRF named corporate.

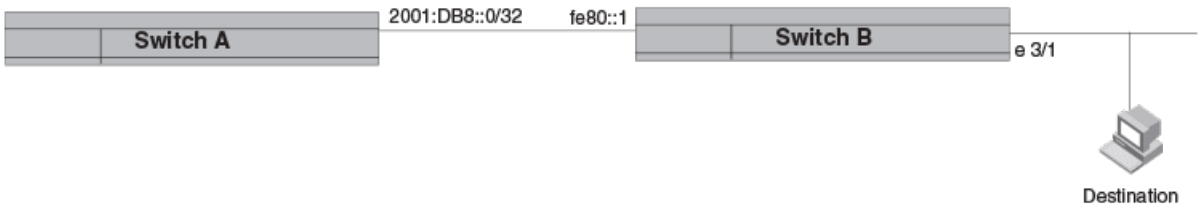
```
device# configure terminal
device(config)# vrf corporate
device(config-vrf-corporate)# rd 20:10
device(config-vrf-corporate)# address-family ipv6 unicast
device(config-vrf-corporate-ipv6)# no ipv6 route 2002::/64 ethernet 1/1
```

# Configuring an interface as next hop

Before configuring a static IPv6 route, you must enable the forwarding of IPv6 traffic on the Layer 3 switch using the **ipv6 unicast-routing** command and enable IPv6 on at least one interface by configuring an IPv6 address or explicitly enabling IPv6 on that interface.

To configure an IPv6 static route with an interface as the next hop as depicted in the following illustration, perform these steps.

**FIGURE 23** IPv6 static route with an interface as next hop




1. Enter global configuration mode.

   ```
   device# configure terminal
   ```

2. Designate the route destination. On the same command line, enter the keyword **ethernet** followed by the interface number as the next-hop, followed by its link-local IPv6 address.

   ```
   device(config)# ipv6 route 2001:DB8::0/32 ethernet 3/1 fe80::1
   ```

The following example configures a static IPv6 route for a destination network with the prefix 2001:DB8::0/32 and a next-hop gateway with the link-local address fe80::1 that the Layer 3 switch can access through Ethernet interface 3/1.

```
device# configure terminal
device(config)# ipv6 route 2001:DB8::0/32 ethernet 3/1 fe80::1
```

# Configuring a virtual interface as next hop

Before configuring a static IPv6 route, you must enable the forwarding of IPv6 traffic on the Layer 3 switch using the **ipv6 unicast-routing** command and enable IPv6 on at least one interface by configuring an IPv6 address or explicitly enabling IPv6 on that interface.

To configure a basic IPv6 static route with a virtual interface as a next hop, perform these steps.

1. Enter global configuration mode.

   ```
   device# configure terminal
   ```

2. Enter the IP address prefix and prefix length for the route destination network.

   ```
   device(config)# ipv6 route 2001:DB8::0/32
   ```

   This example shows the first half of the command, the route destination, IPv6 2001:DB8::0/32 network addresses.

3. On the same command line, add the keyword **ve** followed by the virtual interface ID to be used as the next hop, along with its link-local address.

```
device(config)# ipv6 route 2001:DB8::0/32 ve 3 fe80::1
```

This example shows the next-hop destination as virtual interface (ve) 3, with a link-local address of fe80::1.

The following example configures an IPv6 static route to IPv6 2001:DB8::0/32 destinations through next-hop virtual interface 3.

NOTE
The packet destination will be used as next hop over the virtual interface if the next-hop address is not configured in the command.

```
device# configure terminal
device(config)# ipv6 route 2001:DB8::0/32 ve 3 fe80::1
```

# Configuring a tunnel as next hop

Before configuring a static IPv6 route, you must enable the forwarding of IPv6 traffic on the Layer 3 switch using the **ipv6 unicast-routing** command and enable IPv6 on at least one interface by configuring an IPv6 address or explicitly enabling IPv6 on that interface.

To configure a basic IPv6 static route through a next-hop tunnel, perform these steps.

1. Enter global configuration mode.

```
device# configure terminal
```

2. Enter the IPv6 destination address and prefix, followed by the keyword **ipv6_tnl** and the tunnel ID, followed by the link-local address of the next hop.

```
device(config)# ipv6 route 2001:DB8::0/32 ipv6_tnl 1 fe80::1
```

This example configures Tunnel 1 as the next-hop gateway for IPV6 2001:DB8::0/32 destinations through the link-local address as next hop.

NOTE
To configure a tunnel that allows both IPv4 and IPv6 packets to be carried as IPv4 packets, use the keyword **6to4_tnl** plus the tunnel ID and link-local address to be used as the next-hop gateway.

The following example configures an IPv6 static route to 2001:DB8::0/32 destinations with a next-hop gateway through Tunnel 1.

```
device# configure terminal
device(config)# ipv6 route 2001:DB8::0/32 ipv6_tnl 1 fe80::1
```

# Configuring a VRF as next hop for an IPv6 static route

A non-default VRF can be configured as the next-hop gateway for an IPv6 static route.

Before configuring a static IPv6 route, you must enable the forwarding of IPv6 traffic on the Layer 3 switch using the **ipv6 unicast-routing** command and enable IPv6 on at least one interface by configuring an IPv6 address or explicitly enabling IPv6 on that interface.

NOTE
The VRF designated in the procedure must be a valid VRF.

To configure a VRF as the next hop for an IPv6 static route, follow these steps.

1. Enter global configuration mode.

```
device# configure terminal
```

2. Enter the **ipv6 route** command followed by the IPv6 destination address and prefix length. On the same command line, enter the keyword **next-hop-vrf** followed by the name of the VRF that contains the next-hop gateway router and its IPv6 address.

```
device(config)# ipv6 route 2001:DB8::0/32 next-hop-vrf partners 2001:DB8:0:ee44::1
```

This example creates an IPv6 static route to IPv6 2001:DB8::0/32 destinations through the VRF named "partners" and the next-hop router with the IPv6 address 2001:DB8:0:ee44::1.

# Adding metrics to an IPv6 static route

You can influence how likely a static route is to be used by modifying the cost metric or the administrative distance.

Before configuring a static IPv6 route, you must enable the forwarding of IPv6 traffic on the Layer 3 switch using the **ipv6 unicast-routing** command and enable IPv6 on at least one interface by configuring an IPv6 address or explicitly enabling IPv6 on that interface.

Follow these steps to create an IPv6 static route with cost metrics.

1. Enter global configuration mode.

```
device# configure terminal
```

2. Designate the route destination and next hop, and add a route priority parameter.

| Option | Description |
|---|---|
| Cost metric | The value is compared to the metric for other static routes in the IPv6 route table to the same destination. Two or more routes to the same destination with the same metric load share traffic to the destination. The value may range from 1 through 16. The default is 1. A route with a cost of 16 is considered unreachable. |
| Administrative distance | This value is compared to the administrative distance of all routes to the same destination. Static routes by default take precedence over learned protocol routes. However, to give a static route a lower priority than a dynamic route, give the static route the higher administrative distance. The value is preceded by the keyword **distance** and can range from 1 to 255. The default is 1. A value of 255 is considered unreachable. |

```
device(config)# ipv6 route 2001:DB8::0/64 2001:DB8:2343:0:ee44::1 2
```

This example configures a static IPv6 route for a destination network with the prefix 2001:DB8::0/64 and a next-hop gateway with the global address 2001:DB8:0:ee44::1 and assigns the route a metric of 2.

```
device(config)# ipv6 route 2001:DB8::0/64 2001:DB8:2343:0:ee44::1 distance 254
```

This example configures a static route with an administrative distance of 254.

The following example configures an IPv6 static route for a destination network with the prefix 2001:DB8::0/64 and a next-hop gateway with the global address 2001:DB8:0:ee44::1. The static route is assigned an administrative distance of 3.

```
device# configure terminal
device(config)# ipv6 route 2001:DB8::0/64 2001:DB8:0:ee44::1 distance 3
```

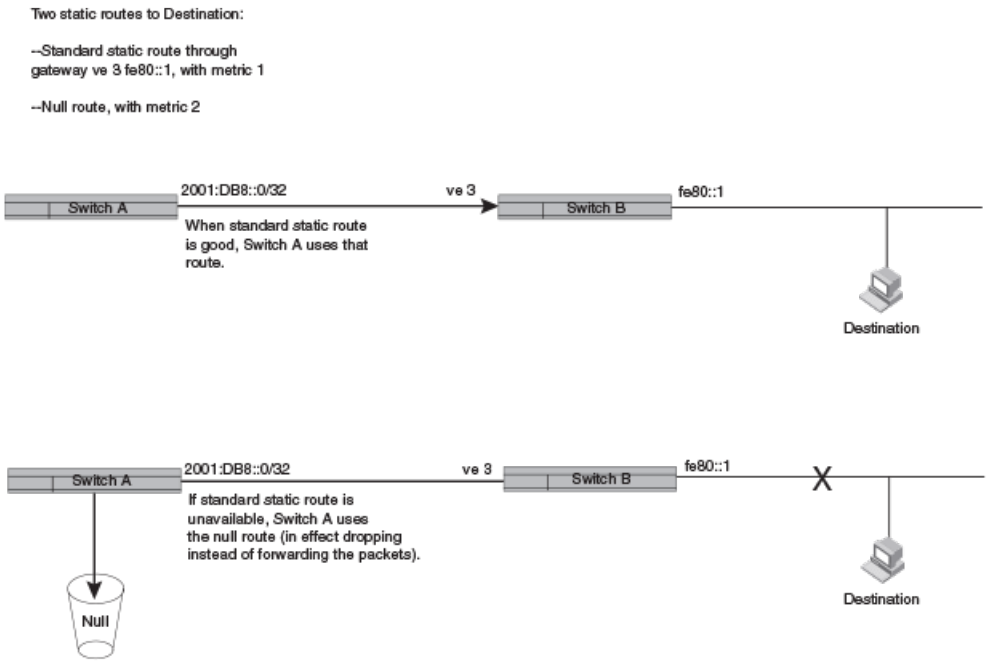
# Configuring a null route

You can configure a null static route to drop packets to a certain destination. This is useful when the traffic should not be forwarded if the preferred route is unavailable.

> NOTE
> You cannot add a null or interface-based static route to a network if there is already a static route of any type with the same metric you specify for the null or interface-based route.

The following figure depicts how a null static route works with a standard route to the same destination.

**FIGURE 24** Null route and standard route to same destination




The following procedure creates a preferred route and a null route to the same destination. The null route drops packets when the preferred route is not available.

1. Enter global configuration mode.

   ```
   device# configure terminal
   ```

2. Configure the preferred route to a destination.

   ```
   device(config)# ipv6 route 2001:DB8::0/64 ve 3 fe80::1
   ```

   This example creates a static route to IPv6 2001 : DB8 : : 0/64 destination addresses. These destinations are routed through link-local address fe80::1 and the next hop gateway virtual interface (ve) 3. The route uses the default cost metric of 1.

3. Configure the null route to the same destination with a higher metric.

   ```
   device(config)# ipv6 route 2001:DB8::0/64 null0 2
   ```

   This example creates a null static route to the same destination. The metric is set higher so that the preferred route is used if it is available.

The following example creates a primary route to all 2001 : DB8 : : 0/64 destinations through virtual interface (ve) 3. It creates an alternative null route to drop the packets when the primary route is not available.

```
device# configure terminal
device(config)# ipv6 route 2001 : DB8 : : 0/64 ve 3 fe80::1
device(config)# ipv6 route 2001 : DB8 : : 0/64 null0 2
```

# Configuring a default static route

You can manually create a default static route that the router uses if there are no other default routes to a destination.

Because the default route is a static route, you must configure the **ip route next-hop-enable-default** command to resolve other static routes through the default route.

Perform these steps to configure a default route.

1. Enter global configuration mode.

   ```
   device# configure terminal
   ```

2. Enable the default network route for static route resolution of routes to a particular destination.

   ```
   device(config)# ipv6 route next-hop-enable-default
   ```

   This example enables the default static route to resolve the next hop for IPv6 static routes.

   NOTE
   This command can be independently applied on a per-VRF basis.

3. (Optional) Configure the default route for recursive lookup of the next-hop.

   ```
   device# configure terminal
   device(config)# ipv6 route next-hop-recursion
   ```

   This example allows three levels of recursion in looking up the next hop for any IPv6 static route. The default is 3. You may enter any value from 1 to 10.

4. Enter the following destination route and network mask followed by a valid next-hop address.

   ```
   device(config)# ipv6 route ::/0 2001:DB8:0:ee44::1
   ```

The following example configures a default static route to global IPv6 address 2001:DB8:0:ee44::1. The route is able to resolve static routes using next-hop recursion to three levels (the default).

```
device# configure terminal
device(config)# ipv6 route next-hop-enable-default
device(config)# ipv6 route next-hop-recursion
device(config)# ipv6 route ::/0 2001:DB8:0:ee44::1
```

# Resolving the IPv6 static route through a protocol

You can use routes from another protocol to resolve a static route.

Perform these steps to resolve the next hop for an IPv6 static route using learned routes from BGP, OSPF, RIP, or ISIS protocol.

1. Enter global configuration mode.

   ```
   device# configure terminal
   ```

2. Designate next-hop resolution through BGP, RIP, OSPF, or ISIS protocol.

The following example specifies that IPv6 static routes can be resolved through directly connected OSPF routers (instead of link-local IPv6 route tables, for example).

```
device# configure terminal
device(config)# ipv6 route next-hop ospf
```

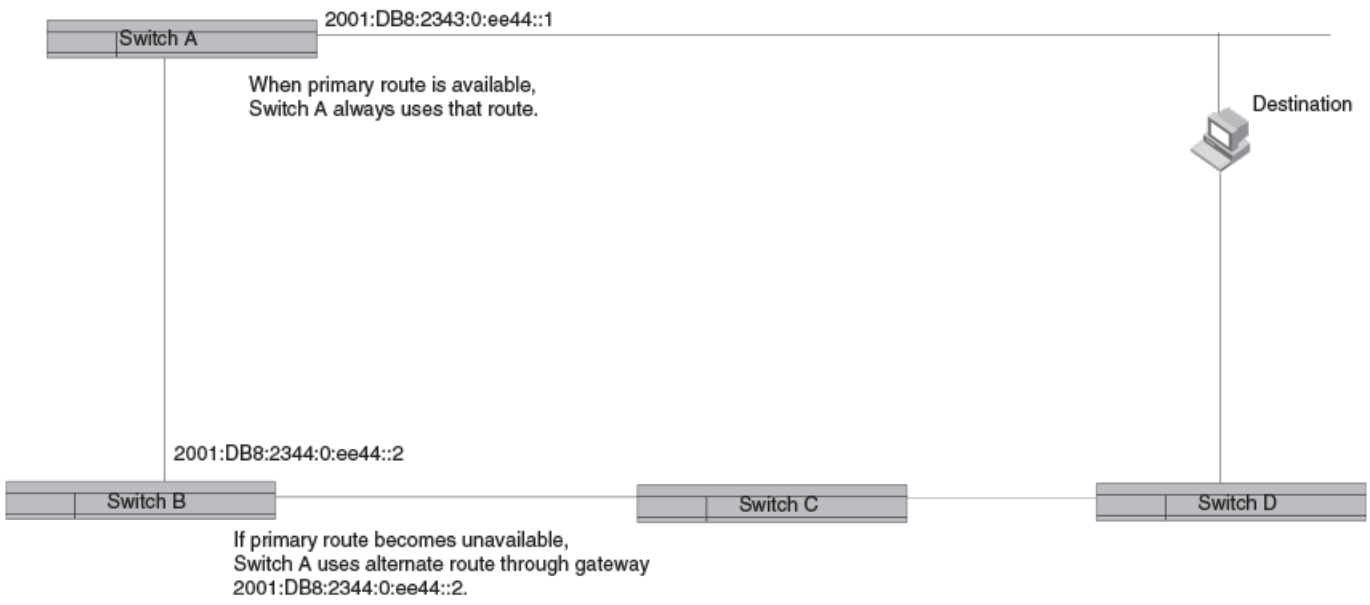
# Configuring load sharing and redundancy

You can configure multiple IP static routes to the same destination to set up load sharing or backup routes.

If you configure more than one static route to the same destination with different next-hop gateways but the same metrics, the router load balances among the routes using a basic round-robin method.

If you configure multiple static IP routes to the same destination with different next-hop gateways and different metrics, the router always uses the route with the lowest metric. If this route becomes unavailable, the router fails over to the static route with the next-lowest metric. The following figure depicts multiple routes with different metrics configured for the same destination.

**FIGURE 25** Two static routes to same destination




To set up multiple routes for load sharing or redundancy, perform the following steps.

NOTE
You can also use administrative distance to set route priority; however, be sure to give a static route a lower administrative distance than other types of routes, unless you want other route types to be preferred over the static route.

1. Enter global configuration mode.

```
device# configure terminal
```

2. Enter multiple routes to the same destination using different next hops.

```
device(config)# ipv6 route 2001:DB8::0/64 2001:DB8:2343:0:ee44::1
device(config)# ipv6 route 2001:DB8::0/64 2001:DB8:2344:0:ee44::2
```

This example creates two next-hop gateways for all 2001:DB8::0/64 destinations. Traffic will alternate between the two paths.

3. To prioritize multiple routes, use different metrics for each possible next hop.

```
device(config)# ipv6 route 2001:DB8::0/64 2001:DB8:2343:0:ee44::1
device(config)# ipv6 route 2001:DB8::0/64 2001:DB8:2344:0:ee44::2 2
```

This example creates an alternate route to all 2001:DB8::0/64 destinations. The primary route uses 2001:DB8:2343:0:ee44::1 as the next hop. The route has the default metric of 1. If this path is not available, traffic is directed through 2001:DB8:2344:0:ee44::2, which has the next lowest metric (2).

# Adding an IPv6 static route tag for use with route-maps

Before configuring a static IPv6 route, you must enable the forwarding of IPv6 traffic on the Layer 3 switch using the **ipv6 unicast-routing** command and enable IPv6 on at least one interface by configuring an IPv6 address or explicitly enabling IPv6 on that interface.

To configure an IPv6 static route with a tag that can be referenced in a route-map, follow these steps.

1. Enter global configuration mode.

```
device# configure terminal
```

2. Configure the IPv6 static route destination address and next-hop address. On the same command line, enter the keyword tag, followed by the decimal number to be referenced later in a route-map.

```
device(config)# ipv6 route 2001:DB8::0/64 2001:DB8:0:ee44::1 tag 3
```

The following example configures an IPv6 route to IPv6 2001:DB8::0/64 destinations through next-hop 2001:DB8:0:ee44::1. The route has the tag ID "3," which can be referenced later in a route-map.

```
device# configure terminal
device(config)# ipv6 route 2001:DB8::0/64 2001:DB8:0:ee44::1 tag 3
```

# IPv6 multicast static routes

IPv6 multicast routes allow you to control the network path used by multicast traffic.

Static multicast routes are especially useful when the unicast and multicast topologies of a network are different. You can avoid the need to make the topologies similar by instead configuring static multicast routes.

You can configure more than one static IPv6 multicast route. The device by default uses the most specific route that matches a multicast source address. Thus, if you want to configure a multicast static route for a specific multicast source and also configure another multicast static route for all other sources, you can configure two static routes.

You can also influence route preference using cost metrics and administrative distance parameters.

NOTE
Regardless of the administrative distances, the device always prefers directly connected routes over other routes.

The following example configures an IPv6 multicast static route for a destination network with the prefix 2001:db8::0/32, a next-hop gateway with the global address 2001:db8:0:ee44::1, and an administrative distance of 110.

```
device# configure terminal
device(config)# ipv6 mroute 2001:db8::0/32 2001:db8:0:ee44::1 distance 110
```

# Configuring IPv6 multicast routes in a non-default VRF

You can specify a default or non-default VRF for an IPv6 multicast static route. If no VRF is specified, the default VRF is used.

The VRF specified must be an existing VRF.

Follow these steps to create an IPv6 multicast static route in a non-default VRF.

1. Enter global configuration mode.

   ```
   device# configure terminal
   ```

2. Designate the non-default VRF for the IPv6 multicast static route.

   ```
   device# configure terminal
   device(config)# vrf corporate
   ```

   This example configures "corporate" as the non-default VRF.

3. Give the route a Route Descriptor.

   ```
   device(config-vrf-corporate)# rd 20:10
   ```

   This example configures 20:10 as the route descriptor.

4. Specify the address family as IPv6.

   ```
   device(config-vrf-corporate)# address-family ipv6
   ```

5. Configure the IPv6 static route, including destination IP address, mask prefix, and next-hop information.

   ```
   device(config-vrf-corporate-ipv6)# ipv6 mroute 2002::/64 ethernet 1/1
   ```

   This example configures an IPv6 static route to IP address 2002::/64 destinations via next-hop interface 1/1.

The following example creates an IPv6 multicast route with the RD 20:10 to 2002::/64 IP address via next-hop Ethernet interface 1/1. The route has a cost metric of 5.

```
device# configure terminal
device(config)# vrf corporate
device(config-vrf-corporate)# rd 20:10
device(config-vrf-corporate)# address-family ipv6
device(config-vrf-corporate-ipv6)# ipv6 mroute 2002::/64 ethernet 1/1 fe80::1 5
```

# Displaying information on IPv6 static routes

You can consult the IPv6 route table for information on connected, static, and protocol routes.

To display information on IPv6 static routes, use the following commands.

1. To check whether IPv6 is enabled, enter the **show ipv6** command. The command can be entered at the device prompt or in global or interface configuration mode.

```
device# show ipv6
Global Settings
  IPv6 Router-Id: 1.1.1.1   load-sharing path: 4
  unicast-routing enabled, ipv6  allowed to run, hop-limit 64
  reverse-path-check disabled
  nd6 proxy disabled
  host drop cam limit disabled
  urpf-exclude-default disabled
  session-logging-age 5
  selective-routes-download enabled
  No Inbound Access List Set
  No Outbound Access List Set
  source-route disabled, forward-source-route disabled, icmp-redirect disabled  icmp-mpls-response
enabled
OSPF (default VRF): enabled
```

   This example shows IPv6 is enabled on the device.

2. To display IPv6 route table information, enter the **show ipv6 route** command.

```
device# show ipv6 route
IPv6 Routing Table - 2 entries:
Type Codes - B:BGP C:Connected I:ISIS L:Local O:OSPF R:RIP S:Static
BGP  Codes - i:iBGP e:eBGP
ISIS Codes - L1:Level-1 L2:Level-2
OSPF Codes - i:Inter Area 1:External Type 1 2:External Type 2
STATIC Codes - d:DHCPv6
        Type IPv6 Prefix           Next Hop Router    Interface     Dis/Metric     Uptime src-vrf
1       C    2001:1::/32           ::                 ve 201        0/0            17m36s -
2       S    2002:1::/32           ::                 ve 201        1/1            13m55s -
                                   fe80::1            ve 201                              -
```

   The example shows a directly connected and a static route in the IPv6 routing table. Both routes are through virtual interface (ve) 201.

# GPRS Tunneling Protocol



# GPRS Tunneling Protocol Overview

Use GPRS Tunneling Protocol (GTP) to selectively filter and load-balance on GTP fields for GTP packets on GTP enabled ports.

The GPRS core network provides mobility management, session management and transport for Internet Protocol packet services in GSM and WCDMA networks. GPRS is similar to a GSM network except for two new nodes (SGSN and GGSN) and a piece of hardware (PCU).

- The Packet Control Unit (PCU) differentiates data destined for the standard GSM network or Circuit Switched Data and data destined for the GPRS network or Packet Switched Data.
- The Serving GPRS Support Node (SGSN) takes care of some important tasks, including routing, handover and IP address assignment.
- The Gateway GPRS Support Node (GGSN) is the last node in the GPRS network before a connection between an ISP or corporate network's router occurs.

The connection between these two GPRS Support Nodes is made with a protocol called GPRS Tunnelling Protocol (GTP).

NOTE
The GTP is only supported on Gen2+ cards.

# GPRS Tunneling Protocol Filtering and Load-balancing

The GPRS Tunneling Protocol (GTP) Filtering and Load-balancing feature allows you to selectively filter and load-balance on GTP fields for GTP packets on GTP enabled ports.

This feature supports GTP packets and incorporates the Tunnel Endpoint Identifier (TEID) in the Trunk/ECMP hashing logic.

## About GTP load-balance configuration

The GTP profile configuration feature allows you to specify the desired interfaces that are required to process GTP packets, and what options are needed to allow the NetIron device to look deeper into the packet.

NOTE
This feature is supported only on Gen2+ cards, except 24x10G cards.

You can specify the ports using the **ports** command under the GTP profile as shown below.

**Syntax: ports all-ethernet** | **ethernet** *slot/port* **ethernet** | **to** *slot/port*

The GTP profile allows you to configure additional information into the load-balancing hash algorithm through the following commands.

Use the **load-balance port-gtpc-teid-hash-ena** command to enable the GTPc packets to add the TEID field into the hash mechanism along with the L4 header information such as TCP/UDP source port, destination port, L3 header information such as source IP address, destination IP address, and protocol ID, and L2 header information such as source MAC-address, destination MAC address, and VLAN ID for the ports of the profile.

```
device(config-gtp-gtp2)# load-balance port-gtpc-teid-hash-ena
```

Use the **load-balance port-gtpu-innerl3-hash-ena** command to enable the tunneled L3 and L4 information such as TCP/UDP source and destination port, source and destination IP address and protocol ID along with the TEID field and L2 information such as source MAC-address, destination MAC-address, and VLAN ID from GTPu packets to be added into the hash mechanism for the ports of the profile.

```
device(config-gtp-gtp2)# load-balance port-gtpu-innerl3-hash-ena
```

Use the **load-balance port-gtpu-teid-hash-ena** command to enable the TEID field from GTPu packets to be added into the hash mechanism for the ports of the profile.

```
device(config-gtp-gtp2)# load-balance port-gtpu-teid-hash-ena
```

Use the **ingress-inner-filter** command to change the behavior of ACLs and policy-based routing (PBR) on the GTP profile ports. This enables the ACL or PBR to match on the inner Layer 3 and Layer 4 header information of GTPu packets.

```
device(config-gtp-gtp2)# ingress-inner-filter
```

NOTE
The inner ingress filter does not work when the outer IPv6 header contains other headers such as the fragment header between the IPv6 and UDP headers.

NOTE
You can configure up to 16 profiles on a system. A profile names can be up to 64 characters long. The valid range for profile IDs is from 1 through 16.

Use the **load-balance port-gtp-srcport-ena** command to allow decoding the GTP packets based on UDP source port along with UDP destination port. The the **load-balance port-gtp-srcport-ena** command is an additional option to decode and process GTPc or GTPu packets based on the UDP source port for the ports of the profile.

```
device(config-gtp-gtp1)# load-balance port-gtp-srcport-ena
```

NOTE
By default, the GTP packets are processed based on the UDP destination port.

## *Example GTP Configuration*

The following is an example of creating a GTP configuration.

```
device(config)# gtp gtp2 12
device(config-gtp-gtp2)# ports ethernet 10/1 to 10/4
device(config-gtp-gtp2)# load-balance port-gtpc-teid-hash-ena
device(config-gtp-gtp2)# load-balance port-gtpu-teid-hash-ena
device(config-gtp-gtp2)# load-balance port-gtpu-innerl3-hash-ena
device(config-gtp-gtp2)# ingress-inner-filter
device(config-gtp-gtp2)# load-balance port-gtp-srcport-ena
```

### *Example of GTP Configuration*

The following example displays information about a GTP profile.

```
device# show gtp
Total no. of GTP profiles :: 1
===========GTP gtp2 (12)============
GTP configuration
Port count :: 4
Ports :: eth 10/1 to 10/4
Loadbalance hashing options
GTPC TEID Hash enabled          :: Yes
GTPU Inner L3 Hash enabled      :: Yes
GTPU TEID Hash enabled          :: Yes
GTP Source Port enabled         :: Yes
Ingress Inner L4 filter enabled :: Yes
```

## Enable masking of the TEID (Tunnel endpoint identifier) field for GTP packets

To mask information that has been added to the LAG hash algorithm for GTP profile ports, the following commands are available. Each masks the indicated field inside the tunneled L3 and L4 headers, or the GTP header.

By default, the TEID field will not be masked and will be used in hashing calculations only for GTP packets on GTP enabled ports. For non-GTP enabled interfaces, it will not be used, even if GTP packets are received on them. This does not affect the non-GTP packets.

The masking options are available to mask certain fields while calculating the hash for load-balancing. For example, the following command will mask the src-ip address for GTP Ipv4 packets.

```
device(config)#load-balance mask gtp ipv4 src-ip
```

Basic command format is **load-balance mask gtp ipv4**|**ipv6**|**teid** *field option slot*|**all**

The following are the **load-balance mask gtp** options available.

- **ipv4** Mask IPv4 header fields
- **ipv6** Mask IPv6 header fields
- **teid** Mask TEID information in Trunk/ECMP hash

The following are the **load-balance mask gtp ipv4** options available.

- **dst-ip** Mask Destination IP address
- **dst-l4-port** Mask Destination L4 port
- **protocol** Mask IP protocol id
- **src-ip** Mask Source IP address
- **src-l4-port** Mask Source L4 port

The following are the **load-balance mask gtp ipv6** options available.

- **dst-ip** Mask Destination IP address
- **dst-l4-port** Mask Destination L4 port
- **next-hdrl**Mask next header id
- **src-ip** Mask Source IP address
- **src-l4-port** Mask Source L4 port

The following are the **load-balance mask gtp ipv4 dst-ip** options available.

- **DECIMAL** Slot Number
- **all** All slots

# MPLS unknown label handling

This task shows how to use MPLS unknown label handling.

If the MPLS unknown label handling is turned on, all unknown MPLS packets entering the specified ingress interface will be stripped of their MPLS labels, and sent to specified egress interface.

> NOTE
> This command can only be applied to Gen2+ cards, except 24x10G cards.

Enter the **mpls-unknown-label-forward** command to direct all unknown MPLS packets to the specified interface.

The command format is **mpls-unknown-lable-forward ingress** ingress port **egress** egress port

```
device(config)#mpls-unknown-label-forward ingress 10/8 egress 8/1
Reload required. Please write memory and then reload the system.
Failure to reload could cause system instability on failover.
Newly configured mpls catch-all value will not take effect during hitless-reload.
device(config)#
```

# Enabling internal loopback

The internal loopback feature allows the configured interface to redirect packets that would normally egress the interface and be physically loop backed via a short fiber, or specialized hardware device, to the ingress of the interface without a physical device.

This behavior is supported on the following modules:

- BR-MLX-40Gx4-M
- BR-MLX-10Gx20
- BR-MLX-100Gx2-CFP2

> NOTE
> Usage of an optic in interfaces that are configured as internal loopback is not supported.

The following example shows the command to enable this:

```
device(config)#int e 1/1
device(config-if-e10000-1/1)#loopback system
```

# GTP Profile configuration commands

Use the following commands to maintain GTP hashing and filtering configurations in unique GTP profiles. A maximum of 16 profiles are supported.

## *Naming a GTP profile*

Maintain GTP hashing and filtering configurations in unique GTP profiles. A maximum of sixteen profiles are supported.

### Syntax

**gtp_profile** { **ASCII string** | **Decimal** }

### Parameters

| | |
|---|---|
| **ASCII string** | Name of the GTP profile |
| **Decimal** | ID of the GTP profile. |

### Modes

This command operates under the configuration mode (config).

### Usage Guidelines

Use this command to maintain and create GTP hashing and filtering configurations in unique GTP profiles. A maximum of sixteen profiles are supported.

### Examples

```
device(config)#gtp_profile gtp1
```

### History

| Release version | Command history |
|---|---|
| 5.7.00 | This command was introduced. |

## *Adding port list to enable GTP with this profile*

The following section describes how to add a port list to enable GTP in a profile.

### Syntax

**ports** { **slot/port** | **all** }

### Parameters

| | |
|---|---|
| **slot/port** | The port to add to the enable GTP list. |
| **All** | Enable all ports. |

### Modes

This command operates under the GTP configuration mode (config-gtp).

### Usage Guidelines

If the port is LAG, you will need to specify the primary port of the LAG. For LAG, this profile will be applied to all ports of the LAG. When new ports are added to LAG, they will inherit the same profile. If a secondary port specified, it will be rejected.

### Examples

```
device(config-gtp-gtp1)# ports 1/1
```

### History

| Release version | Command history |
|---|---|
| 5.7.00 | This command was introduced. |

## *show gtp*

Displays the GTP profile information.

### Syntax

**show gtp** { **id** | **name** | **interface**}

### Parameters

| | |
|---|---|
| **id** | Displays the GTP by ID. |
| **name** | Displays the GTP by name. |
| **interfacee** | Displays the GTP interfaces. |

### Usage Guidelines

**show gtp**

### Examples

The following is an example of the **show gtp** command.

```
Total no. of GTP profiles :: 1
===========GTP gtp2 (12)============
GTP configuration
Port count :: 4
Ports :: eth 10/1 to 10/4
Loadbalance hashing options
GTPC TEID Hash enabled          :: Yes
GTPU Inner L3 Hash enabled      :: Yes
GTPU TEID Hash enabled          :: Yes
GTP Source Port enabled         :: Yes
Ingress Inner L4 filter enabled :: Yes
```

The following examples displays output based on the GTP ID.

```
device# show gtp id 12
Total no. of GTP profiles :: 1
===========GTP gtp2 (12)============
GTP configuration
Port count :: 4
  Ports :: eth 10/1 to 10/4

Loadbalance hashing options
        GTPC TEID Hash enabled          :: Yes
        GTPU Inner L3 Hash enabled      :: Yes
        GTPU TEID Hash enabled          :: Yes
Ingress Inner L4 filter enabled         :: Yes
```

The following examples displays output based on the GTP name.

```
device# show gtp name gtp2
Total no. of GTP profiles :: 1
===========GTP gtp2 (12)============
GTP configuration
Port count :: 4
  Ports :: eth 10/1 to 10/4

Loadbalance hashing options
        GTPC TEID Hash enabled          :: Yes
        GTPU Inner L3 Hash enabled      :: Yes
        GTPU TEID Hash enabled          :: Yes
Ingress Inner L4 filter enabled         :: Yes
```

The following examples displays output based on the GTP interfaces.

```
device# show gtp interfaces
        Slot/Interface          GTP Profile Name
            10/1               gtp2
            10/2               gtp2
            10/3               gtp2
            10/4               gtp2
```

## History

| Release version | Command history |
|---|---|
| 5.7.00 | This command was introduced. |
| 6.2.00 | The **show gtp** command output was updated. |

## *Show loadbalance mask-options command*

Use this command to display the load balance masking information.

### Syntax

**Show loadbalance mask-options** { **gtp**}

### Usage Guidelines

### Examples

The following is an example of the **show loadbalance mask-options gtp** command output.

```
device# show load-balance mask-options gtp

Mask GTP options -

  Mask GTP TEID is enabled on -
  No Slots

  Mask GTP IPv4 Source IP address is enabled on -
  Slot 1

  Mask GTP IPv4 Destination IP address is enabled on -
  All Slots

  Mask GTP IPv4 Destination L4 port  is enabled on -
  No Slots

  Mask GTP IPv4 Source L4 port is enabled on -
  No Slots

  Mask GTP IPv6 Source IP address is enabled on -
  No Slots

  Mask GTP IPv6 Destination IP address is enabled on -
  No Slots

  Mask GTP IPv6 Source L4 port is enabled on -
  No Slots

  Mask GTP IPv6 Destination L4 port is enabled on -
  No Slots

  Mask GTP IPv6 Next Header information is enabled on -
  No Slots
```

### History

| Release version | Command history |
|---|---|
| 5.7.00 | This command was introduced. |

# BFD



# Bidirectional Forwarding Detection overview

Bidirectional Forwarding Detection (BFD) is a unified detection mechanism used to rapidly detect link faults. BFD improves network performance by providing fast forwarding path failure detection times.

BFD provides rapid detection of the failure of a forwarding path by checking that the next-hop device is alive. When BFD is not enabled, it can take time to detect that a neighboring device is not operational. This causes packet loss due to incorrect routing information at a level unacceptable for real-time applications such as VOIP and video over IP.

Using BFD, you can detect a forwarding path failure in 300 milliseconds or less, depending on your configuration.

A BFD session is automatically established when a neighbor is discovered for a protocol, provided that BFD is enabled on the interface on which the neighbor is detected and BFD is also enabled for the protocol at the interface level or globally. Once a session is established, each device transmits control messages at a high rate of speed that is negotiated by the devices during the session setup. To provide a detection time of 150 milliseconds, it is necessary to process 20 messages per second of about 70 to 100 bytes each per session. A similar number of messages also need to be transmitted out per session. Once a session is established, that same message is continuously transmitted at the negotiated rate and a check is made that the expected control message is received at the agreed frequency from the neighbor. If the agreed upon messages are not received from the neighbor within a negotiated timeout period, the neighbor is considered to be down.

For CER 2000 Series and CES 2000 Series devices, there are 20 Bidirectional Sessions per LP and 40 Bidirectional sessions system-wide.

BFD can provide failure detection on any kind of path between systems, including direct physical links, multihop routed paths, and tunnels. Multiple BFD sessions can be established between the same pair of systems when multiple paths between them are present in at least one direction, even if a lesser number of paths are available in the other direction.

NOTE

BFD session establishment on an interface does not start until 180 seconds after the interface comes up. The reason for this delay is to ensure that the link is not effected by unstable link conditions which could cause BFD to flap. This delay time is not user configurable.

The BFD Control Message is an UDP message with destination port 3784.

NOTE
The source port for BFD control packets is in the range 49152 through 65535. The source port number is unique among all BFD sessions on the system.

NOTE
For single-hop sessions, all BFD control packets are sent with a time to live (TTL) or hop limit value of 255. All received BFD control packets are discarded if the received TTL or hop limit is not equal to 255.

NOTE
BFD supports multi-slot LAGs in cases where all BFD packets are transmitted only on a single path which does not change unless the LAG active membership changes. BFD is not be supported on multi-slot LAGs where per-packet switching is used such that the path taken by the BFD packets will vary per packet.

NOTE
When BFD is configured with stringent values of 100/300 msec, BFD may flap when learning a large number of routes.

NOTE
BFD sessions configured with lower timer values may exhibit flaps when configured alongside MACSec on same line card. This issue is a known limitation. However, the BFD sessions are stable with 250ms*3 timer value or more in such scenarios.

# General BFD considerations and limitations

There are a number of general points to consider when configuring BFD:

- BFD protocol version 1 is supported. BFD version 0 is not supported. BFD version 1 and BFD version 0 are not compatible.
- BFD single-hop sessions always use the primary IP address as the source address. Secondary IP addresses cannot be used as source addresses on single-hop sessions.

Refer to BFD considerations and limitations for Layer 3 protocols on page 256 and the *BFD considerations and limitations for static routes* section of the "IP Route Policy" chapter for more information on BFD considerations and limitations.

# BFD for Layer 3 protocols

BFD can be used by Layer 3 protocols for rapid failure detection in the forwarding path between two adjacent routers, including the interfaces, data links, and forwarding planes.

BFD can be configured for use with the following protocols:

- OSPFv2
- OSPFv3
- IS-IS
- BGP4
- BGP4+

BFD must be enabled at both the interface and routing protocol levels. BFD asynchronous mode, which depends on the sending of BFD control packets between two systems to activate and maintain BFD neighbor sessions between routers, is supported. Therefore, in order for a BFD session to be created, BFD must be configured on both BFD peers.

Once BFD is enabled on the interfaces and at the router level for the appropriate routing protocols, a BFD session is created. BFD timers are then negotiated, and the BFD peers begin to send BFD control packets to each other at the negotiated interval.

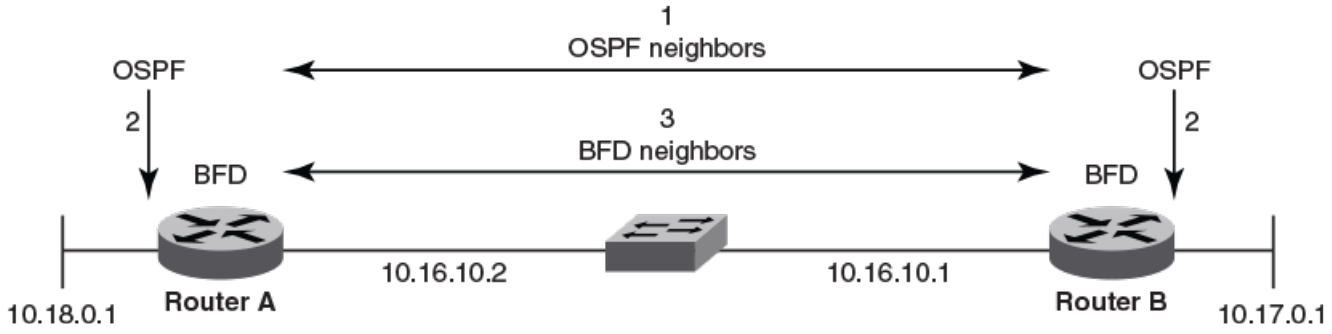
BFD provides a single point of forwarding path monitoring. This means that when more than one Layer 3 application wants to monitor a single host, BFD runs a single session for that host and provides the status to multiple applications, instead of multiple applications running individual sessions to the host.

By sending rapid failure detection notices to the routing protocols in the local device to initiate the routing table recalculation process, BFD contributes to greatly reducing overall network convergence time.

NOTE
BFD brings IS-IS and OSPF down with it when RSTP path-cost changes are made to the switch Alt Discarding port.

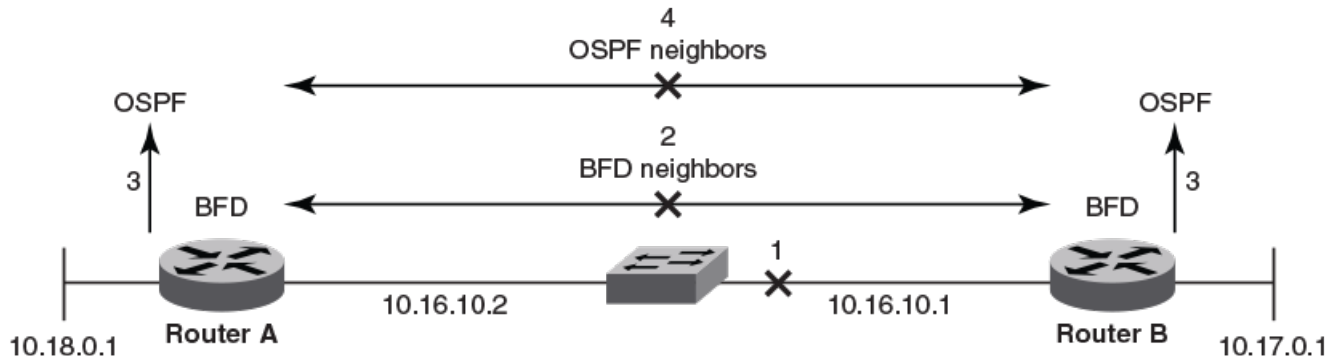
The following figure shows the establishment of a BFD session where OSPF discovers a neighbor and sends a request to BFD requesting that a BFD neighbor session be created with the OSPF neighbor router.

FIGURE 26 Establishing a BFD neighbor session




The following figure shows the termination of a BFD neighbor session after a failure occurs in the network.

FIGURE 27 Termination of a BFD neighbor session



# BFD considerations and limitations for Layer 3 protocols

There are a number of things to consider when configuring BFD for Layer 3 protocols:

- BFD supports both single-hop and multihop sessions.
- For single-hop sessions, the IP address that matches the subnet of the destination IP address is always used as the source IP address of the BFD control packet. If the source IP address is changed, the session is brought down unless another corresponding address with the same subnet is available.
- For single-hop sessions, an IPv6 address that matches the prefix and scope of the destination IPv6 address is used as the source IPv6 address of the BFD control packet.
- For multihop sessions, BFD clients provide the source IP address and BGP is notified of any change to the source IP address of the BFD control packet.
- Multicast or anycast address IP addresses can not be used as source IP addresses.
- BFD establishes only a single BFD session per data protocol path (IPv4 or IPv6) regardless of the number of protocols that want to utilize it.
- Registration is global across all VRFs, even if a neighbor does not support BFD.
- Inactive sessions are created when BFD is not enabled on a remote device that contains the destination IP address. Sessions created are in the Admin Down state and are transmitted at a slower rate than configured values.
- BFD sessions can be established on both primary and secondary IP and IPv6 addresses.
- BFD sessions can be established on link-local IPv6 addresses.
- Changing the BFD parameters does not reset the current BFD session.
- When BFD notifies BGP or OSPF that a session has transitioned from up to down, the protocol does not immediately bring down the session if the holdover timer is configured. The protocol waits until the period of time specified for the holdover timer has expired. If BFD declares a session up before this period of time expires, no action is taken by the protocol.
- BFD sessions remain intact, even after a HA failover, unless there is no topology change or the BFD session went down from the remote end.
- On point-to-point links, the source address of a BFD control packet cannot be used to identify the session.
- If you unconfigure a BFD session that is in the up state, OSPF or BGP tell BFD to delete the session and set the reason as Admin Down. Upon receipt of this notification, BFD deletes the session and communicates this change to the remote BFD peer. The remote BFD neighbor keeps the session in the down state and propagates Remote Admin Down event to the routing protocols.
- BFD session establishment on an interface does not start until 180 seconds after the interface comes up. The reason for this delay is to ensure that the link is not effected by unstable link conditions which could cause BFD to flap. This delay time is not user configurable.

# BFD for Layer 3 protocols on virtual Ethernet interfaces

BFD can be configured for Layer 3 protocols on virtual Ethernet (VE) interfaces.

When configuring BFD for Layer 3 protocols on VE interfaces, one of the physical ports is chosen to set up the session. The physical port used for setting up the BFD session is allocated after Address Resolution Protocol (ARP) is resolved for that neighbor.

If a member of a VE port goes down and a new egress port is identified, the session is moved to another physical port. If a new egress port is not identified, BFD tries to locate a destination IPv4 or IPv6 address. If the new egress port is not learned within the BFD detection time, the session is declared as down.

# Configuring BFD on an interface

BFD can be configured on device interfaces. Repeat the steps in this procedure for each interface on which you want to configure BFD sessions.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **interface** command and specify an interface.

```
device(config)# interface ethernet 1/1
```

3. Enter the **bfd interval** command, specifying an interval, with the **min-rx** and **multiplier** keywords to configure BFD session parameters on the interface.

```
device(config-if-e1000-1/1)# bfd interval 110 min-rx 120 multiplier 15
```

The following example configures BFD on a specific Ethernet interface by setting the baseline BFD session parameters on that interface.

```
device# configure terminal
device(config)# interface ethernet 1/1
device(config-if-e1000-1/1)# bfd interval 110 min-rx 120 multiplier 15
```

# BFD for BGP

BFD support for BGP4 and BGP4+ can be configured so that BGP is a registered protocol with BFD and receives forwarding path detection failure messages from BFD.

BFD is supported for BGP and is disabled by default. When BFD for BGP is enabled, BFD rapidly detects faults on links between BGP peers and reports faults to BGP. BFD for BGP is supported for both for single-hop and multihop iBGP and eBGP sessions with either IPv4 or IPv6 neighbors in the default VRF and nondefault VRF instances. BFD behavior is identical for iBGP and eBGP single-hop and multihop sessions, and for IPv4 and IPv6 neighbors.

NOTE
BFD for unnumbered EMCP interfaces is supported for BGP in IP Fabric deployments. For further information, refer to the *Extreme Network OS IP Fabrics Configuration Guide*.

Consider the following when configuring BFD for BGP:

- Registration is global across all VRFs once BGP sends a registration message to BFD.
- BFD sessions for remote BGP neighbors are not triggered if BFD is not configured on these neighbors. Each neighbor can have its own transmit interval, receive internal, and detect multiplier. If the value is not configured, the configured global value is inherited.
- As soon as a BGP session enters the Established state, BGP requests that BFD start a BFD session.
- BFD sessions are maintained across a BGP graceful restart.

# Configuring BFD session parameters for BGP

BFD session parameters can be set globally for BGP-enabled interfaces.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **router bgp** command to enable BGP routing.

```
device(config)# router bgp
```

3. Enter the **bfd-enable** command to enable BFD globally.

```
device(config-bgp)# bfd-enable
```

4. Enter the **bfd** command with the **min-tx**, **min-rx**, and **multiplier** keywords to configure BFD session parameters globally for BGP-enabled interfaces.

```
device(config-bgp)# bfd min-tx 110 min-rx 120 multiplier 15
```

NOTE
The **bfd** command is used for singlehop sessions only. Multihop sessions in BGP use either the values configured at interface level using the **bfd interval** command or the default interval values.

5. Enter the **bfd holdover-interval** command and specify a value to set the BFD holdover interval globally for BGP-enabled interfaces.

```
device(config-bgp)# bfd holdover-interval 15
```

The following example enables BFD globally and configures BFD session parameters for BGP-enabled interfaces.

```
device# configure terminal
device(config)# router bgp
device(config-bgp)# bfd-enable
device(config-bgp)# bfd min-tx 110 min-rx 120 multiplier 15
device(config-bgp)# bfd holdover-interval 15
```

# Enabling BFD sessions for a specified BGP neighbor

BFD sessions can be configured for specified BGP neighbors.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **router bgp** command to enable BGP routing.

```
device(config)# router bgp
```

3. Enter the **bfd-enable** command to enable BFD globally.

```
device(config-bgp)# bfd-enable
```

4. Enter the **neighbor bfd** command, specifying an IP address, with the **min-tx**, **min-rx**, and **multiplier** keywords to set the BFD session timer values for the specified BGP neighbor.

```
device(config-bgp)# neighbor 10.10.1.1 bfd min-tx 120 min-rx 140 multiplier 10
```

5. Enter the **neighbor bfd holdover-interval** command, specifying an IP address, and enter a value to set the BFD holdover interval for the specified BGP neighbor.

```
device(config-bgp)# neighbor 10.10.1.1 bfd holdover-interval 12
```

The following example configures the BFD session parameters for a BGP neighbor with the IP address 10.10.1.1.

```
device# configure terminal
device(config)# router bgp
device(config-bgp)# bfd-enable
device(config-bgp)# neighbor 10.10.1.1 bfd min-tx 120 min-rx 140 multiplier 10
device(config-bgp)# neighbor 10.10.1.1 bfd holdover-interval 12
```

# Enabling BFD sessions for a specified BGP neighbor in a nondefault VRF

BFD sessions can be configured for specified BGP neighbors in a nondefault VRF instance.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **router bgp** command to enable BGP routing.

```
device(config)# router bgp
```

3. Enter the **address-family ipv4 unicast** command with the **vrf** keyword, specifying a VRF name, to enter BGP address-family IPv4 unicast VRF configuration mode.

```
device(config-bgp)# address-family ipv4 unicast vrf green
```

4. Enter the **bfd-enable** command to enable BFD globally.

```
device(config-bgp-ipv4u-vrf)# bfd-enable
```

5. Enter the **neighbor bfd** command, specifying an IP address, with the **min-tx**, **min-rx**, and **multiplier** keywords to set the BFD session timer values for the specified BGP neighbor.

```
device(config-bgp-ipv4u-vrf)# neighbor 10.10.1.1 bfd min-tx 120 min-rx 140 multiplier 10
```

6. Enter the **neighbor bfd holdover-interval** command, specifying an IP address, and enter a value to set the BFD holdover interval for the specified BGP neighbor.

```
device(config-bgp-ipv4u-vrf)# neighbor 10.10.1.1 bfd holdover-interval 12
```

7. Enter the **neighbor fail-over** command, specifying an IP address and using the **bfd-enable** keyword, to enable BFD support for failover for the BGP neighbor.

```
device(config-bgp-ipv4u-vrf)# neighbor 10.10.1.1 fail-over bfd-enable
```

The following example configures the BFD session parameters for a BGP neighbor with the IP address 10.10.1.1 and enables BFD support for failover for the BGP neighbor.

```
device# configure terminal
device(config)# router bgp
device(config-bgp)# address-family ipv4 unicast vrf green
device(config-bgp-ipv4u-vrf)# bfd-enable
device(config-bgp-ipv4u-vrf)# neighbor 10.10.1.1 bfd min-tx 120 min-rx 140 multiplier 10
device(config-bgp-ipv4u-vrf)# neighbor 10.10.1.1 bfd holdover-interval 12
device(config-bgp-ipv4u-vrf)# neighbor 10.10.1.1 fail-over bfd-enable
```

# Enabling BFD sessions for a specified BGP peer group

BFD sessions can be configured for specified BGP peer groups.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **router bgp** command to enable BGP routing.

```
device(config)# router bgp
```

3. Enter the **bfd-enable** command to enable BFD globally.

```
device(config-bgp)# bfd-enable
```

4. Enter the **neighbor** *peer-group-name* **peer-group** command to create a peer group.

```
device(config-bgp)# neighbor pg1 peer-group
```

5. Enter the **neighbor bfd** command, specifying a peer group, with the **min-tx**, **min-rx**, and **multiplier** keywords to set the BFD session timer values for the specified BGP peer group.

```
device(config-bgp)# neighbor pg1 bfd min-tx 120 min-rx 140 multiplier 10
```

6. Enter the **neighbor bfd holdover-interval** command, specifying a peer group, and enter a value to set the BFD holdover interval for the specified BGP peer group.

```
device(config-bgp)# neighbor pg1 bfd holdover-interval 12
```

The following example configures the BFD session parameters for a BGP peer group called “pg1”.

```
device# configure terminal
device(config)# router bgp
device(config-bgp)# bfd-enable
device(config-bgp)# neighbor pg1 peer-group
device(config-bgp)# neighbor pg1 bfd min-tx 120 min-rx 140 multiplier 10
device(config-bgp)# neighbor pg1 bfd holdover-interval 12
```

# Enabling BFD sessions for a specified BGP peer group in a nondefault VRF

BFD sessions can be configured for specified BGP peer groups in a nondefault VRF instance.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **router bgp** command to enable BGP routing.

```
device(config)# router bgp
```

3. Enter the **address-family ipv4 unicast** command with the **vrf** keyword, specifying a VRF name, to enter BGP address-family IPv4 unicast VRF configuration mode.

```
device(config-bgp)# address-family ipv4 unicast vrf green
```

4. Enter the **bfd-enable** command to enable BFD globally.

```
device(config-bgp-ipv4u-vrf)# bfd-enable
```

5. Enter the **neighbor** *peer-group-name* **peer-group** command to create a peer group.

```
device(config-bgp-ipv4u-vrf)# neighbor pg2 peer-group
```

6. Enter the **neighbor bfd** command, specifying a peer group, with the **min-tx**, **min-rx**, and **multiplier** keywords to set the BFD session timer values for the specified BGP peer group.

```
device(config-bgp-ipv4u-vrf)# neighbor pg2 bfd min-tx 130 min-rx 130 multiplier 12
```

7. Enter the **neighbor bfd holdover-interval** command, specifying a peer group, and enter a value to set the BFD holdover interval for the specified BGP peer group.

```
device(config-bgp-ipv4u-vrf)# neighbor pg2 bfd holdover-interval 14
```

8. Enter the **neighbor fail-over** command, specifying a peer group and using the **bfd-enable** keyword, to enable BFD support for failover for the BGP peer group.

```
device(config-bgp-ipv4u-vrf)# neighbor pg2 fail-over bfd-enable
```

The following example configures the BFD session parameters for a BGP peer group called “pg1” and enables BFD support for failover for the BGP neighbor.

```
device# configure terminal
device(config)# router bgp
device(config-bgp)# address-family ipv4 unicast vrf green
device(config-bgp-ipv4u-vrf)# bfd-enable
device(config-bgp-ipv4u-vrf)# neighbor pg2 peer-group
device(config-bgp-ipv4u-vrf)# neighbor pg2 bfd min-tx 130 min-rx 130 multiplier 12
device(config-bgp-ipv4u-vrf)# neighbor pg2 bfd holdover-interval 14
device(config-bgp-ipv4u-vrf)# neighbor pg2 fail-over bfd-enable
```

# BFD for OSPF

BFD support for OSPFv2 and OSPFv3 can be configured so that OSPF is a registered protocol with BFD and receives forwarding path detection failure messages from BFD.

BFD sessions can rapidly detect link faults and notify OSPF so that it quickly responds to network topology changes. BFD is supported for OSPF and is disabled by default.

Consider the following when configuring BFD for OSPF:

- OSPF uses single-hop BFD sessions.
- Virtual links are not supported.
- BFD for OSPF can be enabled in interface subtype configuration mode, OSPF VRF configuration mode, or OSPFv3 configuration mode. BFD must be enabled at both the interface level and global level to enable BFD for OSPF sessions.
- OSPF sends BFD a registration message even if BFD is not enabled on an interface or globally at the router OSPF level.
- Registration is global across all VRFs.
- BFD sessions are maintained across OSPF graceful restart.
- BFD for OSPF does not support authentication for BFD.

# BFD for OSPF session creation and deletion

OSPF neighbors are discovered dynamically using the OSPF hello protocol. However, in the case of non-broadcast multiple access (NBMA) and point to multipoint (P2MP), neighbors must be manually configured. Regardless of whether neighbors are discovered dynamically or statically configured, all neighbors are associated with an interface. When BFD is enabled on an interface and at the global level, BFD sessions are created for all OSPF neighbors that are in greater than the 2-way state. A BFD session is created once it progresses to the INIT state.

Each interface can have its own BFD timers. OSPF does not influence the BFD timer value for the session. The configured BFD value, or the default value, is used by the BFD module when creating the session. A single-hop session is created for OSPF neighbors.

NOTE
When BFD for an OSPF session is configured, the normal OSPF hello mechanism is not disabled.

NOTE
OSPF neighbor sessions do not flap when BFD is enabled or disabled on the interface where the OSPF session is associated.

NOTE
OSPF assumes full connectivity between all systems on multi-access media such as LANs. If BFD is running on only a subset of systems on such a network, the assumptions of the control protocol may be violated, with unpredictable results.

OSPF BFD session deletion can happen in the following instances:

- If an OSPF neighbor session moves to a state below 2-way, OSPF triggers a BFD session deletion setting the reason as Path Down. When BFD receives this notification, it deletes the corresponding session. The remote BFD neighbor detects this as a detection timer expiry and propagates this session down to OSPF to terminate the session.
- If OSPF is disabled on an interface, all BFD sessions associated with each neighbor are deleted.
- If BFD is administratively disabled on an interface, BFD reports this event as a port change notification to OSPF. Upon receipt of this notification, a BFD session deletion for each neighbor is sent and BFD removes these sessions if no other client is using the same sessions. BFD communicates this change to the remote peer. When the remote BFD peer receives this Remote Admin Down event, it propagates this event to the OSPF protocol. OSPF does not provide any action for this event.

# Enabling BFD on a specified OSPFv2-enabled interface

BFD sessions can be configured on one or more OSPFv2-enabled interfaces.

BFD sessions are initiated on specified OSPFv2-enabled interfaces only if BFD is enabled globally using the **bfd all-interfaces** command in OSPF VRF configuration mode.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **interface** command and specify an interface.

```
device(config)# interface ethernet 1/1
```

3. Enter the **ip ospf bfd** command to enable BFD on the specified interface.

```
device(config-if-e1000-1/1)# ip ospf bfd
```

The following example enables BFD on a specified OSPFv2-enabled Ethernet interface.

```
device# configure terminal
device(config)# interface ethernet 1/1
device(config-if-e1000-1/1)# ip ospf bfd
```

# Configuring BFD for OSPFv2 globally

BFD for OSPFv2 can be configured globally on interfaces where BFD has been configured.

BFD must be configured using the **ip ospf bfd** command on each OSPFv2 interface to initiate BFD sessions.

1. Enter the **configure terminal** command to access global configuration mode.

   ```
   device# configure terminal
   ```

2. Enter the **router ospf** command to enter OSPF router configuration mode and enable OSPFv2 on the device.

   ```
   device(config)# router ospf
   ```

3. Enter the **bfd all-interfaces** command to enable BFD globally.

   ```
   device(config-ospf-router)# bfd all-interfaces
   ```

4. Enter the **bfd holdover-interval** command to set the BFD holdover interval.

   ```
   device(config-ospf-router)# bfd holdover-interval 12
   ```

The following example enables BFD globally and sets the BFD holdover interval to 12.

```
device# configure terminal
device(config)# router ospf
device(config-ospf-router)# bfd all-interfaces
device(config-ospf-router)# bfd holdover-interval 12
```

# Enabling BFD on a specified OSPFv3-enabled interface

BFD sessions can be configured on one or more OSPFv3-enabled interfaces.

BFD sessions are initiated on specified OSPFv3-enabled interfaces only if BFD is enabled globally using the **bfd** command in OSPFv3 VRF configuration mode.

1. Enter the **configure terminal** command to access global configuration mode.

   ```
   device# configure terminal
   ```

2. Enter the **interface** command and specify an interface.

   ```
   device(config)# interface ethernet 1/1
   ```

3. Enter the **ipv6 ospf bfd** command to enable BFD on the specified OSPFv3-enabled interface.

   ```
   device(config-if-e1000-1/1)# ipv6 ospf bfd
   ```

The following example enables BFD on an OSPFv3-enabled Ethernet interface.

```
device# configure terminal
device(config)# interface ethernet 1/1
device(config-if-e1000-1/1)# ipv6 ospf bfd
```

## Configuring BFD for OSPFv3 globally

BFD for OSPFv3 can be configured globally on interfaces where BFD has been configured.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **ipv6 router ospf** command to enter OSPFv3 router configuration mode and enable OSPFv3.

```
device(config)# ipv6 router ospf
```

3. Enter the **bfd all-interfaces** command to enable BFD for all interfaces.

```
device(config-ospf6-router)# bfd all-interfaces
```

4. Enter the **bfd holdover-interval** command to set the BFD holdover interval.

```
device(config-ospf6-router)# bfd holdover-interval 20
```

The following example enables BFD and sets the BFD holdover interval to 20.

```
device# configure terminal
device(config)# ipv6 router ospf
device(config-ospf6-router)# bfd all-interfaces
device(config-ospf6-router)# bfd holdover-interval 20
```

# BFD for IS-IS

BFD support for IS-IS (for both IPv4 and IPv6 IS-IS neighbors) can be configured so that IS-IS is a registered protocol with BFD. BFD support for the IS-IS protocol can be configured globally or for specific interfaces.

NOTE
You cannot configure BFD for an IS-IS session when one side of the IS-IS adjacency is using IPv4 only and the other side is using IPv6 only.

## Enabling BFD on a specified IS-IS-enabled interface

BFD sessions can be configured on one or more IS-IS-enabled interfaces.

BFD sessions are initiated on specified IS-IS interfaces only if BFD is enabled globally using the **bfd** command in ISIS router configuration mode.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **interface** command and specify an interface.

```
device(config)# interface ethernet 1/4
```

3. Enter the **isis bfd** command to enable BFD on the specified interface.

```
device(config-if-e1000-1/4)# isis bfd
```

The following example enables BFD on a specified IS-IS-enabled Ethernet interface.

```
device# configure terminal
device(config)# interface ethernet 1/4
device(config-if-e1000-1/4)# isis bfd
```

## Configuring BFD for IS-IS globally

BFD for IS-IS can be configured globally on interfaces where BFD has been configured.

BFD must be configured using the **isis bfd** command on each IS-IS interface to initiate BFD sessions.

1. Enter the **configure terminal** command to access global configuration mode.

   ```
   device# configure terminal
   ```

2. Enter the **router isis** command to enter ISIS router configuration mode and enable IS-IS on the device.

   ```
   device(config)# router isis
   ```

3. Enter the **bfd** command with the **all-interfaces** keyword to enable BFD globally.

   ```
   device(config-isis-router)# bfd all-interfaces
   ```

4. Enter the **bfd holdover-interval** command, and enter a value, to set the BFD holdover interval.

   ```
   device(config-isis-router)# bfd holdover-interval 12
   ```

The following example enables BFD globally and sets the BFD holdover interval to 12.

```
device# configure terminal
device(config)# router isis
device(config-isis-router)# bfd all-interfaces
device(config-isis-router)# bfd holdover-interval 12
```

# BFD for static routes

Unlike dynamic routing protocols, such as OSPF and BGP, static routing has no method of peer discovery and fault detection. BFD for IPv4 and IPv6 static routes provides rapid detection of failure in the bidirectional forwarding path between BFD peers.

BFD for static routes allows you to detect failures that impact the forwarding path of a static route. This feature supports both single-hop and multihop BFD static routes for both IPv4 and IPv6. Unless the BFD session is up, the gateway for the static route is considered unreachable, and the affected routes are not installed in the routing table. BFD can remove the associated static route from the routing table if the next-hop becomes unreachable indicating that the BFD session has gone down.

Static routes and BFD neighbors are configured separately. A static route is automatically associated with a static BFD neighbor if the static route's next-hop exactly matches the neighbor address of the static BFD neighbor and BFD monitoring is enabled for the static route.

When a static BFD neighbor is configured, BFD asks the routing table manager (RTM) if there is a route to the neighbor. If a route exists, and if the route is directly connected, then BFD initiates a single-hop session. If the route is not directly connected, BFD establishes a multi-hop session. Once the session comes up, BFD adds the corresponding static routes to RTM. If no route exists, then BFD will not add the corresponding static routes to RTM.

When a BFD session goes down because the BFD neighbor is no longer reachable, static routes monitored by BFD are removed from the routing table manager. The removed routes can be added back if the BFD neighbor becomes reachable again. Single-hop BFD

sessions use the BFD timeout values configured on the outgoing interface. Timeout values for multihop BFD sessions are specified along with each BFD neighbor. Multiple static routes going to the same BFD neighbor use the same BFD session and timeout values.

## Configuration considerations

- In a multi-hop session, the protocol must be stated in the **ip route next-hop** command.
- BFD multi-hop is supported for a nexthop resolved through OSPF, BGP, ISIS, RIP, and MPLS.
- BFD multi-hop is not supported for a nexthop resolved through Default Route.
- BFD for static routes is not supported for static routes with an LSP name as nexthop.
- BFD session establishment on an interface does not start until 180 seconds after the interface comes up. The reason for this delay is to ensure that the link is not effected by unstable link conditions which could cause BFD to flap. This delay time is not user configurable.
- BFD for static routes will not support interface-based static routes for both IPv4 and IPv6.
- When CER 2000 Series or CES 2000 Series devices are heavily loaded or under stress, BFD sessions may flap if the configured BFD interval is less than 500 milliseconds with a multiplier value of 3.

## BFD for static routes configuration

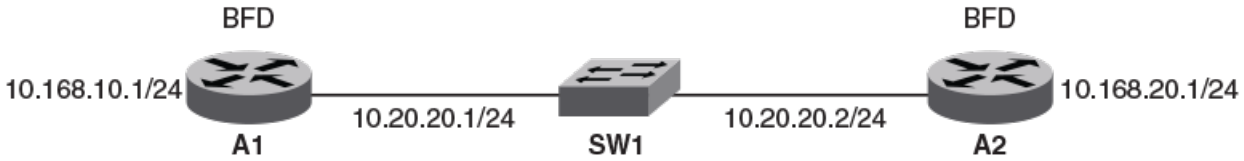
Single-hop BFD IPv4 static route sessions use the timer values configured for the outgoing interface to the directly connected neighbors. Multihop BFD IPv4 static route sessions use the timer values configured using the **ip route static bfd** and **ip route static bfd holdover-interval** commands. If the timer values configured conflict with the timer values set for BGP for the same next-hop, BFD uses the smaller value to meet the more stringent requirement.

Single-hop BFD IPv6 static route sessions use the timer values configured for the outgoing interface to the directly connected neighbors. Multihop BFD IPv6 static route sessions use the timer values configured using the **ipv6 route static bfd** and **ipv6 route static bfd holdover-interval** commands. If the timer values configured conflict with the timer values set for BGP for the same next-hop, BFD uses the smaller value to meet the more stringent requirement.

If you remove a static BFD session, the corresponding session is removed by BFD without removing static routes from the routing table and ongoing traffic is not disrupted. If a BFD session goes down because a BFD neighbor is no longer reachable, all associated static routes are removed from the routing table. Existing traffic on these static routes is interrupted.

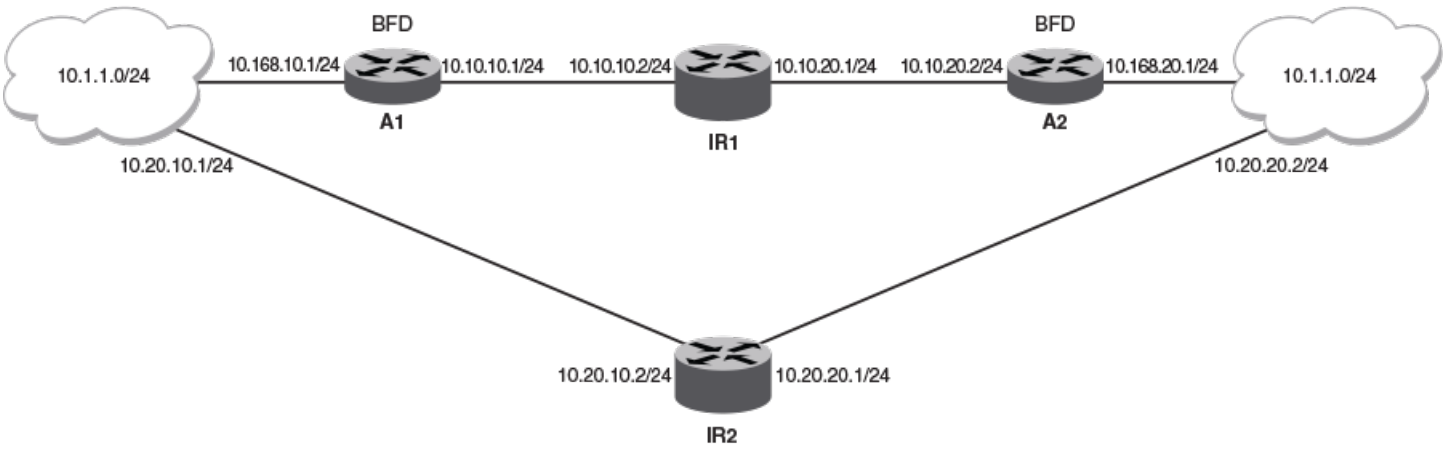
The following figure shows a single-hop static BFD session. A1 has a static route to 10.168.20.0/24 with the next-hop as 10.20.20.2. A2 has a static route to 10.168.10.0/24 with the next-hop as 10.20.20.1. A switch is connected between the routers. BFD can be configured to monitor next-hop 10.20.20.2 on A1 and 10.20.20.1 on A2.

**FIGURE 28** Single-hop static BFD session




The following figure shows a multihop static BFD session. Static routes are configured on A1 to reach the 10.1.1.0/24 subnet by way of 10.168.20.1. 10.168.20.1 is reachable by way of the intermediate routers IR1 and IR2. A static route is configured on A2 to reach the 10.1.1.0/24 subnet by way of 10.168.10.1. This next-hop is reachable in turn by way of the intermediate routers IR1 and IR2. If one BFD session goes down, the corresponding route is removed from the routing table and the data packets take another path.

**FIGURE 29** Multihop ECMP static BFD session




NOTE

When configuring BFD for static routes, static routes are already installed in the routing table and traffic is running on those static routes. When you configure BFD on these static routes, a similar BFD configuration also occurs on BFD neighbors. If BFD session creation fails or a BFD session does not come up, associated static routes are not removed from the routing table; therefore ongoing traffic on these static routes is not interrupted. A BFD session may not be established if a neighbor is busy or if the maximum number of sessions has been reached on the neighbor. Ongoing traffic on installed static routes is not interrupted.

# Configuring BFD on an IP static route

BFD can be configured globally on IP static routes. Repeat the steps in this procedure on each BFD neighbor.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **ip route-static bfd** command, specifying a source and destination IP address, and use the **interval**, **min-rx**, and **multiplier** keywords to configure BFD session parameters on an IP static route.

```
device(config)# ip route static-bfd 10.0.2.1 10.1.1.1 interval 500 min-rx 500 multiplier 5
```

The following example configures BFD session parameters on an IP static route where the destination IP address is 10.0.2.1 and the source IP address is 10.1.1.1.

```
device# configure terminal
device(config)# ip route static-bfd 10.0.2.1 10.1.1.1 interval 500 min-rx 500 multiplier 5
```

# Configuring BFD on an IP static route in a nondefault VRF instance

BFD can be configured on IP static routes in a nondefault VRF instance. Repeat the steps in this procedure on each BFD neighbor.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **vrf** command and specify a VRF name to enter Virtual Routing and Forwarding (VRF) configuration mode.

```
device(config)# vrf green
```

3. Enter the **address-family ipv4** command to enter IPv4 address-family configuration mode..

```
device(config-vrf-green)# address-family ipv4 unicast
```

4. Enter the **ip route static-bfd** command, specifying a source and destination IP address, and use the **interval**, **min-rx**, and **multiplier** keywords to configure BFD session parameters on an IP static route in a nondefault VRF instance.

```
device(config-vrf-green-ipv4)# ip route static-bfd 10.0.0.1 10.1.1.2 interval 500 min-rx 500
multiplier 5
```

The following example configures BFD session parameters on an IP static route in a nondefault VRF instance, where the destination IP address is 10.0.0.1 and the source IP address is 10.1.1.2.

```
device# configure terminal
device(config)# vrf green
device(config-vrf-green)# address-family ipv4
device(config-vrf-green-ipv4)# ip route static-bfd 10.0.0.1 10.1.1.2 interval 500 min-rx 500 multiplier 5
```

# Configuring BFD on an IPv6 static route

BFD can be configured globally on IPv6 static routes. Repeat the steps in this procedure on each BFD neighbor.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **ipv6 route static-bfd** command, specifying a source and destination IPv6 address and an interface type, and use the **interval**, **min-rx**, and **multiplier** keywords to configure BFD session parameters on an IPv6 static route.

```
device(config)# ipv6 route static-bfd fe80::a fe80::b 20 interval 100 min-rx 100 multiplier 10
```

The following example configures BFD session parameters on an IPv6 static route where the destination IPv6 address is fe80::a and the source IPv6 address is fe80::b. A VE interface is specified and the BFD holdover interval is set globally to 25 for IPv6 static routes.

```
device# configure terminal
device(config)# ipv6 route static-bfd fe80::a fe80::b 20 interval 100 min-rx 100 multiplier 10
```

# Configuring BFD on an IPv6 static route in a nondefault VRF instance

BFD can be configured on IPv6 static routes in a nondefault VRF instance. Repeat the steps in this procedure on each BFD neighbor.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **vrf** command and specify a VRF name to enter Virtual Routing and Forwarding (VRF) configuration mode.
3. Enter the **address-family ipv6** command to enter IPv6 address-family configuration mode..

```
device(config-vrf-blue)# address-family ipv6 unicast
```

4. Enter the **ipv6 route static-bfd** command, specifying a source and destination IPv6 address and an interface type, and use the **interval**, **min-rx**, and **multiplier** keywords to configure BFD session parameters on an IPv6 static route in a nondefault VRF instance.

```
device(config-vrf-blue-ipv6)# ipv6 route static-bfd fe70::a fe60::b interval 1000 min-rx 2000
multiplier 20
```

The following example configures BFD session parameters on an IPv6 static route in a nondefault VRF instance, where the destination IPv6 address is fe80::a and the source IPv6 address is fe80::b.

```
device# configure terminal
device(config)# vrf blue
device(config-vrf-blue)# address-family ipv6
device(config-vrf-blue-ipv6)# ipv6 route static bfd fe70::a fe60::b interval 1000 min-rx 2000 multiplier 20
```

# BFD for RSVP-TE LSP

BFD provides a mechanism to detect data plane failure for MPLS LSP in the order of sub-second. Unlike MPLS LSP ping where MPLS control plane can be verified against the data plane, BFD is used only to detect data plane failure. There are a couple advantages in using BFD instead of LSP ping for MPLS data plane failure detection. BFD provides a faster failure detection mechanism because it does not require control plane verification. BFD can also be used to detect faults for a large number of LSPs without manual trigger for each LSP the way LSP ping does.

BFD session for RSVP LSP is very similar to the BFD session set up for ISIS and OSPF with the following differences:

- MPLS BFD session is bootstrapped using LSP ping.
- The IP TTL for transmitted MPLS BFD control packets from ingress LSR to egress LSR must be set to 1 instead of 255.
- After MPLS BFD session is up, the local discriminator and the source IP address are not allowed to change without bringing down the MPLS BFD session.
- The transmit and receive portion of the session can be on different LPs because the LSP is unidirectional and the returned path from egress to ingress LSR depends on IP routing.

MPLS BFD is set up between the ingress and egress LSR. The MPLS BFD session uses an asynchronous operating mode without echo function. BFD demands an operating mode and echo function that are not currently defined for MPLS BFD. Authentication is not supported (it is currently not supported for OSPF or ISIS either).

BFD configuration on the MLX Series and XMR Series devices are provided at the global MPLS configuration level and at the LSP level as follows:

- The global configuration provides you a convenient way to enable and disable BFD for all LSPs that have BFD enabled (this is similar to how it is provided for ISIS and OSPF). In addition, the you can change the default settings for transmit and receive intervals and detection time multiplier to be used for all BFD sessions. On egress LSR, this global setting is used to decide whether a BFD session starts upon receiving MPLS echo request with BFD Discriminator TLV and the time intervals to be included in the BFD control packet are to be sent back to the ingress LSR.
- Under the LSP configuration, you are able to enable or disable BFD and change the default settings for minimum transmit and receive intervals as well as detection time multiple. If those parameters are not specified, the values from the global configuration are used.

BFD can be enabled or disabled without program exit at the global MPLS level or for each individual LSP without affecting the LSP operational status. In addition, the BFD parameters can also be changed without program exit This does not change the state of the BFD session.

One BFD session is created for each non-redundant LSP. BFD session is associated with the active path which can be normal or protected, or detour path. For redundant LSP, a separate BFD session is created for the currently active secondary path.

# BFD session creation for RSVP-TE LSP

On ingress, one BFD session is created for each LSP. A BFD session is created after the LSP comes UP. The BFD session status is displayed as part of the show LSP output. When the BFD session is not UP, a failure reason is displayed as part of the show LSP detail output. Possible failure reasons include exceeding the maximum number of BFD sessions the system can support or the global BFD configuration is disabled. In the case where BFD session is not brought up because BFD packet from the egress LSR is not received, the MPLS Echo Request with the BFD Discriminator TLV is resent until the session is UP. The retry timer is exponentially backed off.

On egress, a BFD session is created after an MPLS Echo Request is received with the BFD Discriminator TLV and the MPLS BFD enabled globally and the maximum number of BFD sessions has not been reached yet. The BFD session created on egress LSR also is counted towards the maximum number of BFD sessions. When the number of BFD sessions has reached a maximum, neither the MPLS Echo Reply nor the BFD control packet are sent. The ingress LSR will retry.

Because the source IP address cannot be changed for MPLS BFD session after the session has comes up, the LSR-ID is used as the source IP address for all MPLS BFD packets. This guarantees that the session does not go down when the LSP path switch occurs.

## *FRR LSP*

Only one BFD session is created for a FRR LSP. When a switchover from protected to detour path occurs, and when the detour is on a different LP, the BFD session is moved to the LP where the detour path resides. The BFD session can go down when the LP already has a maximum number of BFD sessions running. When the detour is on the same LP, the outgoing interface and label stack is updated on the existing LP.

A BFD session is not created for detour path originating on a transit LSR.

## *Redundant LSP*

One BFD session is created for a primary path of a redundant LSP. When the secondary path is hot-standby, a separate BFD session is created for it if and only if BFD is enabled for a secondary path. The two sessions operate independently.

## *Adaptive LSP*

When a new instance of an adaptive LSP comes UP, a BFD session automatically moves to a new LP when the new instance is created on a different LP. Otherwise, the local outgoing interface and label stack updates on the existing LP. When the BFD session needs to be moved to a different LP, it is possible that the BFD session may be down when the LP already has maximum number of BFD sessions running.

# BFD session deletion for RSVP-TE LSP

A BFD session is deleted when any of the following events happen:

- LSP goes down
- BFD is disabled for the LSP
- BFD is disabled for all LSP (example: through global MPLS configuration)

# BFD session modification for RSVP-TE LSP

You can change the BFD parameters globally or for an individual LSP without program shutdown without affecting the operational status of the LSP and the BFD session. When you change the global configuration, the change is applied to all egress MPLS BFD sessions and only to those ingress BFD sessions whose parameters are derived from the global configuration.

# BFD session down handling for RSVP-TE LSP

When a BFD session for a LSP goes down on ingress LSR because BFD detection time has expired, it may trigger path switchover if possible (protected to detour or primary to secondary path switchover). In the case where there is no alternative path, the LSP is brought down and the BFD session is deleted. The LSP goes through the normal retry mechanism in order to come back UP.

On egress LSR, BFD session down does not have any impact on the RSVP session.

# BFD for RSVP-TE LSPs configuration

BFD can be configured for use with RSVP-LSPs to detect data plane failures for MPLS LSPs. Although the LSP ping facility can also be used for this purpose, BFD provides the following advantages:

- BFD provides faster failure detection because it does not required control plane verification, which is required by LSP ping.
- BFD can be used to detect faults on a large number of LSPs without requiring manual interaction, which is required by LSP ping.

BFD configuration for RSVP-TE LSPs is performed at the global and LSP levels, as described:

- **BFD for RSVP-TE LSPs global configuration** - allows you to enable and disable BFD on all of the RSVP-TE LSPs that have been configured for BFD at the LSP level. In addition, use the global command to set revised default values for the transmit interval, receive interval, and for the detection time multiplier for all BFD sessions on RSVP-TEs. This configuration command can also be used as a convenient method to turn BFD for MPLS on or off.
- **BFD for RSVP-TE LSPs configuration at the LSP level** - allows you to enable and disable BFD for individual RSVP-TE LSPs. You can also change the values for the transmit interval, receive interval, and for the detection time multiplier from the default values. If these values are not specified at this level, they are obtained from the values configured at the global level.

BFD, which is disabled by default, can be enabled or disabled at the global MPLS level, or for each individual LSP without affecting the LSP operational status. BFD parameters can also be changed without changing the state of the BFD session.

BFD for RSVP-TE LSPs operates with Fast ReRoute (FRR), Redundant, and Adaptive LSPs as described:

- **FRR LSPs -** Only one BFD session is created for an FRR LSP. A separate BFD session is not created for the detour path. When a switchover from a protected to a detour path occurs, the detour path resides on another interface module, and the BFD session is moved to that interface module. The BFD session can go down if the interface module has already reached the maximum number of BFD sessions. If the detour path is on the same interface module, the outgoing interface and label stack are updated on that interface module. A BFD session is not created for a detour path originated on a transit LSR.
- **Redundant LSPs -** One BFD session is created for the primary path of a redundant LSP. If the secondary path is in the hot-standby condition, a separate BFD session is created for it, but only if BFD is enabled on the secondary path. The two sessions operate independently.
- **Adaptive LSPs -** If a new instance of an adaptive LSP comes up on a different interface module, its BFD session is automatically created on that module. Otherwise, the local outgoing interface and label stack are updated on the existing interface module. When a BFD session is moved to a different interface module, the BFD session may be brought down if the interface module has already reached the maximum number of BFD sessions allowed on it.

# BFD session support per-router and per-interface module

There is a limit to the number of BFD sessions available on a per-router and per-interface module basis as described:

- **per-router** - A maximum number of 250 BFD sessions are permitted per device
- **per-interface module** - A maximum number of 80 BFD sessions (Tx or Rx) are permitted per-interface module

These limitations are inclusive of any BFD sessions created for OSPFv2 or v3 and IS-IS. If creating a BFD session will exceed these limits, the session will be denied.

# BFD session down behavior

When a BFD session for an LSP goes down on an ingress LSR because the BFD detection time has expired, one of the following path switchovers will be triggered; from the protected path to the detour path, or from the primary path to the secondary path. In configurations with no alternative path, the LSP is brought down and the BFD session is deleted. The LSP then follows the normal retry procedures to come back up. On an egress LSR, a down BFD session does not have any impact on the RSVP session.

## *AdminDown State*

The AdminDown mechanism in BFD is intended to signal that the BFD session is being taken down for administrative purposes, and the session state is not indicative of the activity of the data path. Therefore, a system should not indicate a connectivity failure to a client if either the local session state or the remote session state (if known) transitions to AdminDown when that client has an independent means of activity detection (typically, control protocols).

If a client does not have any independent means of activity detection, a system should indicate a connectivity failure to a client, and assume the semantics of Down state, if either the local or remote session state transitions to AdminDown. Otherwise, the client will not be able to determine whether the path is viable, if not unfortunate results may occur.

## *Reaction to BFD Session State Changes*

If a BFD session transitions from Up state to AdminDown, or the session transitions from Up to Down because the remote system is indicating that the session is in state AdminDown, clients should not take any control protocol action.

# Enabling BFD for RSVP-TE LSPs at the global level

When using BFD for RSVP-TE LSPs, you must configure BFD globally at the **router mpls** level.

1. Enter the **configure terminal** command to access global configuration mode.

   ```
   device# configure terminal
   ```

2. Enter the **router mpls** command to enter MPLS configuration mode.

   ```
   device(config)# router mpls
   ```

3. Enter the **bfd** command to enable BFD globally.

   ```
   device(config-router-mpls)# bfd
   ```

4. Enter the **min-tx** command with the **min-rx** and **multiplier** keywords and enter a value to set new values for the transmit interval, receive interval, and for the detection time multiplier.

   ```
   device(config-mpls-bfd)# min-tx 500 min-rx 500 multiplier 5
   ```

The following example enables BFD globally for RSVP-TE LSPs and sets new values for the transmit interval, receive interval, and for the detection time multiplier.

```
device# configure terminal
device(config)# router mpls
device(config-router-mpls)# bfd
device(config-mpls-bfd)# min-tx 500 min-rx 500 multiplier 5
```

# Enabling BFD for a specific RSVP-TE-LSP

When you configure BFD globally, you must also configure it locally for the individual LSPs on which you want it to operate. You can also set separate values for the transmit interval, receive interval, and for the detection time multiplier for the specified LSP. The following example enables BFD for the LSP named “blue” and sets new parameter values.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **router mpls** command to enter MPLS configuration mode.

```
device(config)# router mpls
```

3. Enter the **lsp** command and specify a name to access LSP subconfiguration mode.

```
device(config-mpls)# lsp blue
```

4. Enter the **bfd** command to enable BFD.

```
device(config-mpls-lsp-blue)# bfd
```

5. Enter the **min-tx** command with the **min-rx** and **multiplier** keywords and enter a value to set new values for the transmit interval, receive interval, and for the detection time multiplier.

```
device(config-mpls-lsp-blue-bfd)# min-tx 500 min-rx 500 multiplier 5
```

The following example enables BFD for the LSP named “blue” and sets new parameter values.

```
device# configure terminal
device(config)# router mpls
device(config-mpls)# lsp blue
device(config-mpls-lsp-blue)# bfd
device(config-mpls-lsp-blue-bfd)# min-tx 500 min-rx 500 multiplier 5
```

# Configuring BFD for the secondary path of an LSP

You can configure BFD for the secondary path of an LSP.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **router mpls** command to enter MPLS configuration mode.

```
device(config)# router mpls
```

3. Enter the **lsp** command and specify a name to access LSP subconfiguration mode.

```
device(config-mpls)# lsp blue
```

4. Enter the **secondary-path** command and specify a name to set a secondary explicit path.

```
device(config-mpls-lsp-blue)# secondary-path alt_sf_to_sj
```

5. Enter the **bfd** command to enable BFD.

```
(config-mpls-lsp-blue-sec-path)# bfd
```

The following example configures BFD for the secondary path of an LSP.

```
device# configure terminal
device(config)# router mpls
device(config-mpls)# lsp blue
device(config-mpls-lsp-blue)# secondary-path alt_sf_to_sj
(config-mpls-lsp-blue-sec-path)# bfd
```

# Enabling the IP router alert option

You can set the IP router alert option for MPLS BFD packets sent from the ingress device to the egress device. NetIron OS devices support the IP router alert option as defined in RFC 2113.

This feature is supported only for XMR Series and MLX Series devices.

1. Enter the **configure terminal** command to access global configuration mode.

   ```
   device# configure terminal
   ```

2. Enter the **router mpls** command to enter MPLS configuration mode.

   ```
   device(config)# router mpls
   ```

3. Enter the **lsp** command and specify a name to access LSP subconfiguration mode.

   ```
   device(config-mpls)# lsp blue
   ```

4. Enter the **bfd** command to enable BFD.

   ```
   device(config-mpls-lsp-blue)# bfd
   ```

5. Enter the **set-router-alert-option** command to set the IP router alert option for MPLS BFD packets sent from the ingress device to the egress device.

   ```
   device (config-mpls-lsp-blue-bfd)#set-router-alert-option
   ```

The following example enables the router alert option under the LSP BFD.

```
device# configure terminal
device(config)# router mpls
device(config-mpls)# lsp blue
device(config-mpls-lsp-blue)# bfd
device (config-mpls-lsp-blue-bfd)# set-router-alert-option
```

# Displaying BFD information

Various **show** commands verify information about BFD configurations.

Use one or more of the following commands to verify BFD information. The commands do not have to be entered in this order.

1. Enter the **show bfd** command.

```
device> show bfd

BFD State: ENABLED Version: 1 Use PBIF Assist: Y
Current Registered Protocols: ospf/0  ospf6/0
All Sessions: Current: 4 Maximum Allowed: 100 Maximum Exceeded Count: 0
LP Sessions: Maximum Allowed on LP: 40  Maximum Exceeded Count for LPs: 0
LP Tx/Rx Sessions LP Tx/Rx Sessions LP Tx/Rx Sessions LP Tx/Rx Sessions
1   4/4           2   2/2            3  0/0            4  0/0
5   0/0           6   0/0            7  0/0            8  0/0
9   0/0           10  0/0            11 0/0            12 0/0
13  0/0           14  0/0            15 0/0            16 0/0
BFD Enabled ports count: 2
Port       MinTx       MinRx       Mult Sessions
eth 2/1    100         100         3    2
eth 3/1    100         100         3    2
```

   This example output displays BFD information.

2. Enter the **show bfd neighbors** command.

```
device> show bfd neighbors

Total number of Neighbor entries: 2
NeighborAddress          State   Interface Holddown  Interval  R/H
10.14.1.1                UP      eth 3/1   300000    100000    Y/S
10.2.1.1                 UP      eth 2/1   300000    100000    Y/S
```

   This example output displays BFD neighbor information for the default VRF.

3. Enter the **show bfd neighbors applications** command.

```
device> show bfd applications

Registered Protocols Count: 3
Protocol  VRFID     Parameter HoldoverInterval
isis      0         0         2
ospf6     0         1         10
ospf      0         0         5
```

   This example displays BFD registered protocol information for the device.

4. Enter the **show bfd neighbors interface** command with the **ethernet** keyword, and specify an interface

```
device> show bfd neighbors interface ethernet 1/1

 BFD State: ENABLED Version: 1 Use PBIF Assist: Y SH setup delay 180 MH setup delay 0
 Current Registered Protocols: mpls/0  ospf/2  ospf6/0  ospf/4  ospf/0
 All Sessions: Current: 0 Maximum Allowed: 250 Maximum Exceeded Count: 0
 Maximum TX/RX Sessions Allowed on LP: 80  Maximum Session Exceeded Count for LPs: 0
   LP Tx/Rx Sessions LP Tx/Rx Sessions LP Tx/Rx Sessions LP Tx/Rx Sessions
   1   0/0           2   0/0           3   0/0           4   0/0
 BFD Enabled ports count: 1
 Port       MinTx       MinRx       Mult Sessions
 eth 1/1    55          55          5    0
```

   This example shows detailed neighbor information about a specified Ethernet interface.

5. Enter the **show bfd neighbors isis** command.

```
device> show bfd neighbors isis

Total Entries:1 R:RxRemote(Y:Yes/N:No)H:Hop(S:Single/M:Multi)
NeighborAddress       State  Interface       Holddown   Interval  R/H
10.40.40.10           UP     eth 3/6         900000     300000    Y/S
```

This example shows BFD neighbor information for IS-IS.

6. Enter the **show bfd neighbors ospf** command.

```
device> show bfd neighbors ospf

Total Entries:1 R:RxRemote(Y:Yes/N:No)H:Hop(S:Single/M:Multi)
NeighborAddress               State  Interface       Holddown   Interval  R/H
1.1.1.1                       UP     eth 1/2         300000     100000    Y/S
```

This example shows BFD neighbor information for OSPFv2.

7. Enter the **show bfd neighbors ospf6** command.

```
device> show bfd neighbors ospf6

Total Entries:1 R:RxRemote(Y:Yes/N:No)H:Hop(S:Single/M:Multi)
NeighborAddress                      State  Interface       Holddown   Interval  R/H
fe80::21b:edff:fe3b:8601             UP     eth 1/2         300000     100000    Y/S
```

This example shows BFD neighbor information for OSPFv3.

8. Enter the **show bfd neighbors static** command.

```
device> show bfd neighbors static

Total Entries:1 R:RxRemote(Y:Yes/N:No)H:Hop(S:Single/M:Multi)
NeighborAddress               State  Interface       Holddown   Interval  R/H
1.1.1.1                       UP     eth 1/2         300000     100000    Y/S
```

This example shows BFD neighbor information for IP static routes.

# BGP4

# BGP4 overview

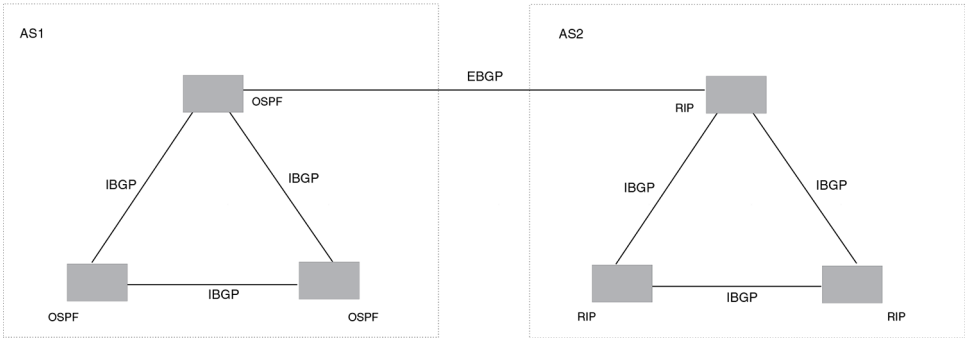
Border Gateway Protocol version 4 (BGP4) is an exterior gateway protocol that performs inter-autonomous system (AS) or inter-domain routing. It peers to other BGP-speaking systems over TCP to exchange network reachability and routing information. BGP primarily performs two types of routing: inter-AS routing, and intra-AS routing. BGP peers belonging to different autonomous systems use the inter-AS routing, referred as Exterior BGP (eBGP). On the other hand, within an AS BGP can be used to maintain a consistent view of network topology, to provide optimal routing, or to scale the network.

BGP is a path vector protocol and implements this scheme on large scales by treating each AS as a single point on the path to any given destination. For each route (destination), BGP maintains the AS path and uses this to detect and prevent loops between autonomous systems.

Devices within an AS can use different Interior Gateway Protocols (IGPs) such as RIP, IS-IS, and OSPF to communicate with one another. However, for devices in different autonomous systems to communicate, they need to use an EGP. BGP4 is the standard EGP used by Internet devices and therefore is the EGP implemented on NetIron devices.

This is a simple example of two BGP4 ASs. Each AS contains three BGP4 devices. All of the BGP4 devices within an AS communicate using iBGP. BGP4 devices communicate with other autonomous systems using eBGP. Notice that each of the devices also is running an Interior Gateway Protocol (IGP). The devices in AS1 are running OSPF and the devices in AS2 are running RIP. The device can be configured to redistribute routes among BGP4, IS-IS, RIP, and OSPF. They also can redistribute static routes.

**FIGURE 30** BGP4 autonomous systems



# BGP4 peering

Unlike OSPF or other IGP protocols, BGP4 does not have neighbor detection capability. BGP4 neighbors (or peers) must be configured manually. A device configured to run BGP4 is called a BGP "speaker." A BGP speaker connects to another speaker (either in the same or a different AS) by using a TCP connection to port 179 (the well-known BGP port), to exchange the routing information. The TCP connection is maintained throughout the peering session. While the connection between BGP peers is alive, two peers communicate by means of the following types of messages:

- OPEN
- UPDATE
- KEEPALIVE
- NOTIFICATION
- ROUTE REFRESH

BGP4 peering can be internal or external, depending on whether the two BGP peers belong to the same AS or different ASs. A BGP4 session between peers within a single AS is referred to as an Interior BGP (iBGP) session; a session between peers belonging to different ASs is referred to as an Exterior BGP (eBGP) session.

In order to establish a TCP connection between two iBGP peers, the IP reachability should be established either by means of the underlying IGP protocol (e.g. OSPF) or by means of static routes. When routes are advertised within iBGP peers, the following primary actions are taken in contrast to eBGP peering:

- Routes learned from an iBGP peer are not usually advertised to other iBGP peers, in order to prevent loops within an AS.
- Path attributes are not usually changed, in order to maintain the best path selection at other nodes within an AS.
- The AS path and next hop are not normally changed.

# BGP4 message types

All BGP4 messages use a common packet header, with the following byte lengths:

| Marker | Length | Type | Data |
|---|---|---|---|
| 16 | 2 | 1 | variable |

**NOTE**
All values in the following tables are in bytes.

Type can be OPEN, UPDATE, NOTIFICATION, KEEPALIVE, or ROUTE-REFRESH, as described below.

## OPEN message

After establishing TCP connection, BGP peers exchange OPEN message to identify each other.

| Version | Autonomous System | Hold-Time | BGP Identifier | Optional Parameter Len | Optional Parameters |
|---|---|---|---|---|---|
| 1 | 2 or 4 | 2 | 4 | 1 | 4 |

**Version**

Only BGP4 version 4 is supported.

**Autonomous System**

Both 2-byte and 4-byte AS numbers are supported.

**KEEPALIVE** and **HOLDTIME** messages

A BGP **timer** command specifies both **keep-alive** and **hold-time** operands that manage the intervals for BGP KEEPALIVE and HOLDTIME messages. The keep alive time specifies how frequently the device sends KEEPALIVE messages to its BGP4 neighbors. The hold time specifies how long the device waits for a KEEPALIVE or UPDATE message from a neighbor before concluding that the neighbor is dead. When two neighbors have different hold-time values, the lowest value is used. A hold-time value of 0 means "always consider neighbor to be active."

Refer to the *Extreme NetIron Command Reference* for more information.

**BGP Identifier**

Indicates the router (or device) ID of the sender. When router-id is not configured, device-id is taken from the loopback interface. Otherwise, the lowest IP address in the system is used.

**Parameter List**

Optional list of additional parameters used in peer negotiation.

# UPDATE message

The UPDATE message is used to advertise new routes, withdraw previously advertised routes, or both.

| WithdrawnRoutesLength | WithdrawnRoutes | Total PathAttributes Len | Path Attributes | NLRI |
|---|---|---|---|---|
| 2 | variable | 2 | variable | variable |

**Withdrawn Routes Length**

Indicates the length of next (withdrawn routes) field. It can be 0.

**Withdrawn Routes**

Contains list of routes (or IP-prefix/Length) to indicate routes being withdrawn.

Total Path Attribute Len

Indicates length of next (path attributes) field. It can be 0.

**Path Attributes**

Indicates characteristics of the advertised path. Possible attributes: Origin, AS Path, Next Hop, MED (Multi-Exit Discriminator), Local Preference, Atomic Aggregate, Aggregator, Community, extended-Communities.

**NLRI**

Network Layer Reachability Information — the set of destinations whose addresses are represented by one prefix. This field contains a list of IP address prefixes for the advertised routes.

# NOTIFICATION message

In case of an error that causes the TCP connection to close, the closing peer sends a notification message to indicate the type of error.

| Error Code | ErrorSubcode | Error Data |
|---|---|---|
| 1 | 1 | variable |

**Error Code**

Indicates the type of error, which can be one of following:

- Message header error
- Open message error
- Update message error
- Hold timer expired
- Finite state-machine error
- Cease (voluntarily)

**Error Subcode**

Provides specific information about the error reported.

**Error Data**

Contains data based on error code and subcode.

## KEEPALIVE message

Because BGP does not regularly exchanges route updates to maintain a session, KEEPALIVE messages are sent to keep the session alive. A KEEPALIVE message contains just the BGP header without data field. Default KEEPALIVE time is 60 seconds and is configurable.

## REFRESH message

A REFRESH message is sent to a neighbor requesting that the neighbor resend the route updates. This is useful when the inbound policy has been changed.

# BGP4 attributes

BGP4 attributes are passed in UPDATE messages to describe the characteristics of a BGP path by the advertising device. At a high level, there are only two types of attributes: well-known and optional. All of the well-known attributes, as described in RFC 4271, are supported.

# BGP4 best path selection algorithm

The BGP decision process is applied to the routes contained in the Routing Information Base, Incoming (RIB-In) which contains routes learned from inbound update messages. The output of the decision process is the set of routes that will be advertised to BGP speakers in local or remote autonomous systems and are stored in the Adjacency RIB, Outgoing (RIB-Out).

When multiple paths for the same route prefix are known to a BGP4 device, the device uses the following algorithm to weigh the paths and determine the optimal path for the route. The optimal path depends on various parameters, which can be modified.

Refer to the *Extreme NetIron Command Reference* for more information.

1. Verify that the next hop can be resolved by means of Interior Gateway Protocol (IGP).
2. Use the path with the largest weight.
3. If the weights are the same, prefer the path with the largest local preference.

4. Prefer the route that was self-originated locally.
5. If the local preferences are the same, prefer the path with the shortest AS-path. An AS-SET counts as 1. A confederation path length, if present, is not counted as part of the path length.

   The **as-path ignore** command disables the comparison of the AS path lengths of otherwise equal paths.

   NOTE
   This step can be skipped if the **as-path-ignore** command is configured.

6. If the AS-path lengths are the same, prefer the path with the lowest origin type. From low to high, route origin types are valued as follows:
   - IGP is lowest.
   - EGP is higher than IGP but lower than INCOMPLETE.
   - INCOMPLETE is highest.
7. If the paths have the same origin type, prefer the path with the lowest MED.

   The device compares the MEDs of two otherwise equivalent paths if and only if the routes were learned from the same neighboring AS. This behavior is called deterministic MED. Deterministic MED is always enabled and cannot be disabled.

   To ensure that the MEDs are always compared, regardless of the AS information in the paths, the **always-compare-med** command can be used. This option is disabled by default.

   The **med-missing-as-worst** command can be used to make the device regard a BGP4 route with a missing MED attribute as the least-favorable path when the MEDs of the route paths are compared.

   MED comparison is not performed for internal routes that originate within the local AS or confederation, unless the **compare-med-empty-aspath** command is configured.
8. Prefer paths in the following order:
   - Routes received through eBGP from a BGP4 neighbor outside of the confederation
   - Routes received through eBGP from a BGP4 device within the confederation *or* routes received through IBGP.
9. If all the comparisons above are equal, prefer the route with the lowest IGP metric to the BGP4 next hop. This is the closest internal path inside the AS to reach the destination.
10. If the internal paths also are the same and BGP4 load sharing is enabled, load-share among the paths. Otherwise go to Step 11.

    NOTE
    For eBGP routes, load sharing applies only when the paths are from neighbors within the same remote AS. eBGP paths from neighbors in different ASs are not compared, unless multipath multi-as is enabled.

11. If **compare-routerid** is enabled, prefer the path that comes from the BGP4 device with the lowest device ID. If a path contains originator ID attributes, then the originator ID is substituted for the router ID in the decision.
12. Prefer the path with the minimum cluster-list length.
13. Prefer the route that comes from the lowest BGP4 neighbor address.

# Implementation of BGP4

BGP4 is described in RFC 1771 and the latest BGP4 drafts. The Extreme BGP4 implementation fully complies with RFC 1771. Extreme BGP4 implementation also supports the following RFCs:

- RFC 1745 (OSPF Interactions)
- RFC 1997 (BGP Communities Attributes)
- RFC 2385 (TCP MD5 Signature Option)
- RFC 2439 (Route Flap Dampening)
- RFC 2796 (Route Reflection)
- RFC 2842 (Capability Advertisement)
- RFC 3065 (BGP4 Confederations)
- RFC 2858 (Multiprotocol Extensions)
- RFC 2918 (Route Refresh Capability)
- RFC 3392 (BGP4 Capability Advertisement)
- Draft-ietf-idr-restart-10.txt (restart mechanism for BGP4)

# Device ID

BGP automatically calculates the device identifier it uses to specify the origin in routes it advertises. If a router-id configuration is already present in the system, then device-id is used as the router-id. Otherwise, the device first checks for a loopback interface, and the IP address configured on that interface is chosen as the device-id. However, if a loopback interface is not configured, the device-id is chosen from lowest-numbered IP interface address configured on the device. Once device-id is chosen, the device identifier is not calculated unless the IP address configured above is deleted.

# BGP global mode

To enable BGP4, use the **router bgp** command in global configuration mode.

```
device(config)# router bgp
```

After using the **router bgp** command you enter into BGP global configuration mode.

Commands entered in BGP global configuration mode apply to the IPv4 unicast address family. Where relevant, this chapter discusses and provides IPv4-unicast-specific examples. You must first configure IPv4 unicast routing for any IPv4 routing protocol to be active.

```
Possible completions:

  additional-paths              Specify BGP  additional paths
  address-family                Enter Address Family command mode
  advertise-best-external       Advertise best external path to internal peers
  aggregate-address             Configure BGP aggregate entries
  always-compare-med            Allow comparing MED from different neighbors
  always-propagate              Allow readvertisement of best BGP routes not
                                in IP forwarding table
  as-path-ignore                Ignore AS_PATH length info for best route
                                selection
  auto-shutdown-new-neighbors   Automatically set new BGP neighbors to
                                shutdown state
  bfd                           Set BFD global parameters for BGP4
  bfd-enable                    Enable BFD for BGP4
  bgp-redistribute-internal     Allow redistribution of iBGP routes into IGPs
```

```
capability                      Set capability
clear                           Clear table/statistics/keys
client-to-client-reflection     Configure client to client route reflection
cluster-id                      Configure Route-Reflector Cluster-ID
compare-med-empty-aspath        Allow comparing MED from different neighbors
                                even with empty as-path attribute
compare-routerid                Compare router-id for identical BGP paths
confederation                   Configure AS confederation parameters
dampening                       Enable route-flap dampening
default-information-originat    Distribute a default route
default-local-preference        Configure default local preference value
default-metric                  Set metric of redistributed routes
distance                        Define an administrative distance
enforce-first-as                Enforce the first AS for EBGP routes
fast-external-fallover          Reset session if link to EBGP peer goes down
graceful-restart                Enables the BGP graceful restart capability
install-igp-cost                Install igp cost to nexthop instead of MED
                                value as BGP route cost
local-as                        Configure local AS number
log-dampening-debug             Log dampening debug messages
maxas-limit                     Impose limit on number of ASes in AS-PATH
                                attribute
maximum-paths                   Forward packets over multiple paths
med-missing-as-worst            Consider routes missing MED attribute as least
                                desirable
multipath                       Enable multipath for ibgp or ebgp neighbors
                                only
neighbor                        Specify a neighbor router
network                         Specify a network to announce via BGP
next-hop-enable-default         Enable default route for BGP next-hop lookup
next-hop-mpls                   Include MPLS routes for BGP next-hop lookup
next-hop-recursion              Perform next-hop recursive lookup for BGP route
redistribute                    Redistribute information from another routing
                                protocol
rib-route-limit                 Limit BGP rib count in routing table
show                            Display system information
static-network                  Special network that do not depends on IGP and
                                always treat as best route in BGP
table-map                       Map external entry attributes into routing
                                table
timers                          Adjust routing timers
update-time                     Configure igp route update interval
```

# Configuring a local AS number

The local AS number (ASN) identifies the AS in which the BGP device resides. The following task configures the local ASN in which the device resides.

NOTE

Use well-known private ASNs in the range from 64512 through 65535 if the AS number of the organization is not known.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **router bgp** command to enable BGP routing.

```
device(config)# router bgp
```

3. Enter the **local-as** command to configure the autonomous system number (ASN) in which your device resides.

```
device(config-bgp)# local-as 1000
```

The following example configures the local ASN for a device.

```
device# configure terminal
device(config)# router bgp
device(config-bgp)# local-as 1000
```

# IPv4 multicast address family

The BGP4 multicast address family configuration level provides access to commands that allow you to configure BGP4 multicast routes. The commands that you enter at this level apply only to the IPv4 multicast address family.

BGP4 supports the IPv4 address family configuration level.

You can generate a configuration for BGP4 multicast routes that is separate and distinct from configurations for BGP unicast routes and IPv6 BGP multicast routes.

The commands that you can access while at the IPv4 multicast address family configuration level are also available at the IPv6 multicast address family configuration levels. Each address family configuration level allows you to access commands that apply to that particular address family only.

The following configurations are allowed under BGP IPv4 address-family multicast mode:

```
device(config-bgp-ipv4m)# ?

Possible completions:

  additional-paths              Specify BGP  additional paths
  address-family                Enter Address Family command mode
  advertise-best-external       Advertise best external path to internal peers
  aggregate-address             Configure BGP aggregate entries
  always-propagate              Allow readvertisement of best BGP routes not
                                in IP forwarding table
  bfd                           Set BFD global parameters for BGP4
  bfd-enable                    Enable BFD for BGP4
  clear                         Clear table/statistics/keys
  client-to-client-reflection   Configure client to client route reflection
  dampening                     Enable route-flap dampening
  default-information-originate Distribute a default route
  default-metric                Set metric of redistributed routes
  log-dampening-debug           Log dampening debug messages
  maximum-paths                 Forward packets over multiple paths
  multipath                     Enable multipath for ibgp or ebgp neighbors
                                only
  neighbor                      Specify a neighbor router
  network                       Specify a network to announce via BGP
  next-hop-enable-default       Enable default route for BGP next-hop lookup
  redistribute                  Redistribute information from another routing
                                protocol
  rib-route-limit               Limit BGP rib count in routing table
  show                          Display system information
  table-map                     Map external entry attributes into routing
                                table
  update-time                   Configure igp route update interval
```

# Neighbor configuration

For each neighbor a device is going to peer with, there must be a neighbor configuration that specifies an IP address (which must be the primary IP address of interface connection to get established) and an AS number of the neighbor. For each neighbor, you can specify a set of attributes. However, in cases where a set of neighbors share the same set of attributes, it is advisable to create a peer-group.

Commands entered in BGP global configuration mode apply to the IPv4 unicast address family. Where relevant, this chapter discusses and provides IPv4-unicast-specific examples. You must first configure IPv4 unicast routing for any IPv4 routing protocol to be active.

The following neighbor configuration options are allowed under BGP global configuration mode:

```
device(config-bgp)# neighbor 10.1.1.1 ?
Possible completions:
   activate                 Allow exchange of route in the current family mode
  additional-paths          Specify bgp additional paths
  advertisement-interval    Minimum interval between sending BGP routing updates
  allowas-in                Accept as-path with my AS present in it
  as-override               Override matching AS-number while sending update
  bfd                       Set BFD parameters for BGP4 neighbor
  capability                Advertise capability to the peer
  default-originate         Originate default route to peer
  description               Neighbor by description
  distribute-list           Apply Address Filters to neighbor
  ebgp-btsh                 Enable EBGP TTL Security Hack Protection
  ebgp-multihop             Allow EBGP neighbors not on directly connected
                            networks
  enforce-first-as          Enforce the first AS for EBGP routes
  fail-over                 Set Failover source for BGP4 for this peer
  filter-list               Establish BGP filters
  local-as                  Assign local-as number to neighbor
  maxas-limit               Impose limit on number of ASes in AS-PATH attribute
  maximum-prefix            Maximum number of prefix accept from this peer
  next-hop-self             Disable the next hop calculation for this neighbor
  password                  Enable TCP-MD5 password protection
  peer-group                Assign peer-group to neighbor
  prefix-list               Prefix List for filtering routes
  remote-as                 Specify a BGP neighbor
  remove-private-as         Remove private AS number from outbound updates
  route-map                 Apply route map to neighbor
  route-reflector-client    Configure a neighbor as Route Reflector client
  send-community            Send community attribute to this neighbor
  shutdown                  Administratively shut down this neighbor
  soft-reconfiguration      Per neighbor soft reconfiguration
  static-network-edge       Neighbor as special service edge, static-network
                            shall not be advertised if installed as DROP
  timers                    BGP per neighbor timers
  unsuppress-map            Route-map to selectively unsuppress suppressed routes
  update-source             Source of routing updates
  weight                    Set default weight for routes from this neighbor
```

# Configuring BGP4 neighbors

BGP4 neighbors can be configured using this procedure.

1. Enter the **configure terminal** command to access global configuration mode.

   ```
   device# configure terminal
   ```

2. Enter the **router bgp** command to enable BGP routing.

   ```
   device(config)# router bgp
   ```

3. Enter the **local-as** command to configure the autonomous system number (ASN) in which your device resides.

   ```
   device(config-bgp)# local-as 1000
   ```

4. Enter the **neighbor remote-as** command, and specify an IP address, to specify the ASN in which the remote neighbor resides.

   ```
   device(config-bgp)# neighbor 10.1.1.1 remote-as 1001
   ```

The following example configures a BGP4 neighbor.

```
device# configure terminal
device(config)# router bgp
device(config-bgp)# local-as 1000
device(config-bgp)# neighbor 10.1.1.1 remote-as 1001
```

## Requiring the first AS to be the neighbor AS

By default, the Extreme device does not require the first AS listed in the AS_SEQUENCE field of an AS path update message from eBGP neighbors be the AS of the neighbor that sent the update.

You can configure the Extreme device to require that the first AS listed in the AS_SEQUENCE field of an AS path update message from eBGP neighbors is the AS of the neighbor that sent the update. You can enable this feature globally on the device, or for a specific neighbor or peer group. The device accepts the update only if the AS numbers match. If the AS numbers do not match, the Extreme device sends a notification message to the neighbor and closes the session. This requirement applies to all updates received from eBGP neighbors.

The hierarchy for enforcement of this feature is as follows:

- If the enforce-first-as value is configured, a neighbor attempts to use this value.
- If none is configured, the neighbor attempts to use the configured value for a peer group.
- If neither configuration exists, enforcement is the global configuration which is disabled by default.

The following example enables this feature globally.

```
device(config-bgp)# enforce-first-as
```

The following example enables this feature for a specific neighbor.

```
device(config-bgp)# neighbor 10.1.1.1 enforce-first-as enable
```

The following example enables this feature for a peer group.

```
device(config-bgp)# neighbor Peergroup1 enforce-first-as enable
```

The following example enables this feature globally, for a peer group, and a neighbor.

```
device(config)# router bgp
BGP4: Please configure 'local-as' parameter in order to enable BGP4.
device(config-bgp)# local-as 1

device(config-bgp)# enforce-first-as
device(config-bgp)# neighbor abc peer-group
device(config-bgp)# neighbor abc remote-as 2
device(config-bgp)# neighbor abc enforce-first-as disable
device(config-bgp)# neighbor 192.168.1.2 peer-group abc
device(config-bgp)# neighbor 192.168.1.2 enforce-first-as enable
```

For a neighbor or peer configuration, when the first-as requirement is enabled, its status appears in the output of the **show running configuration** command. The optional **enable** or **disable** lets you specify whether the output of the **show running configuration** command includes the configuration of the first-as requirement to show what you have actually configured.

# Peer groups

Neighbors having the same attributes and parameters can be grouped together by means of the **neighbor peer-group** command. You must first create a peer-group, after which you can associate neighbor IP addresses with the peer-group. All of the attributes that are allowed on a neighbor are also allowed on a peer-group.

The benefits of peer groups are:

- Simplified neighbor configuration - You can configure a set of neighbor parameters and then apply them to multiple neighbors. You do not need to configure the common parameters individually on each neighbor.
- Flash memory conservation - Using peer groups instead of individually configuring all the parameters for each neighbor requires fewer configuration commands in the startup configuration file.

You can perform the following tasks on a peer-group basis:

- Reset neighbor sessions
- Perform soft-outbound resets (the device updates outgoing route information to neighbors but does not entirely reset the sessions with those neighbors)
- Clear BGP4 message statistics
- Clear error buffers

An attribute value configured explicitly for a neighbor takes precedence over the attribute value configured for a peer-group. If neither the peer-group nor the individual neighbor has the attribute configured, the default value for the attribute is used.

For the parameters of a peer group to take effect, the peer group must be activated in the IPv4 or IPv6 address-family. By default, only IPv4 unicast address family is activated for a peer-group. A user needs to explicitly activate a peer-group in the IPv6 unicast address-family configuration mode when used with IPv6 peers.

# Configuring BGP4 peer groups

A peer group can be created and neighbor IPv4 addresses can be associated with the peer group.

1. Enter the **configure terminal** command to access global configuration mode.

   ```
   device# configure terminal
   ```

2. Enter the **router bgp** command to enable BGP routing.

   ```
   device(config)# router bgp
   ```

3. Enter the **local-as** command to configure the autonomous system number (ASN) in which your device resides.

   ```
   device(config-bgp)# local-as 1000
   ```

4. Enter the **neighbor** *peer-group-name* **peer-group** command to create a peer group.

   ```
   device(config-bgp)# neighbor mypeergroup1 peer-group
   ```

5. Enter the **neighbor** *peer-group-name* **remote-as** command to specify the ASN of the peer group.

   ```
   device(config-bgp)# neighbor mypeergroup1 remote-as 11
   ```

6. Enter the **neighbor** *ip-address* **peer-group** command to associate a neighbor with the peer group.

   ```
   device(config-bgp)# neighbor 10.2.2.2 peer-group mypeergroup1
   ```

7. Enter the **neighbor** *ip-address* **peer-group** command to associate another neighbor with the peer group.

   ```
   device(config-bgp)# neighbor 10.3.3.3 peer-group mypeergroup1
   ```

The following example creates a peer group and specifies two neighbors to belong to the peer group.

```
device# configure terminal
device(config)# router bgp
device(config-bgp)# local-as 1000
device(config-bgp)# neighbor mypeergroup1 peer-group
device(config-bgp)# neighbor mypeergroup1 remote-as 11
device(config-bgp)# neighbor 10.2.2.2 peer-group mypeergroup1
device(config-bgp)# neighbor 10.3.3.3 peer-group mypeergroup1
```

# Advertising the default BGP4 route

By default, a BGP device does not originate and advertise a default route using BGP4. A BGP4 default route is the IP address 0.0.0.0 and the route prefix 0 or network mask 0.0.0.0. For example, 0.0.0.0/0 is a default route. A BGP device can be configured to advertise the default IPv4 route to all BGP4 neighbors and to install that route in the local BGP4 route table.

The default route must be present in the local IPv4 route table.

1. Enter the **configure terminal** command to access global configuration mode.

   ```
   device# configure terminal
   ```

2. Enter the **router bgp** command to enable BGP routing.

   ```
   device(config)# router bgp
   ```

3. Enter the **default-information-originate** command to advertise the default IPv4 route to all BGP4 neighbors and to install that route in the local BGP4 route table.

   ```
   device(config-bgp)# default-information-originate
   ```

The following example enables a BGP4 device to advertise the default IPv4 route to all BGP4 neighbors and to install that route in the local BGP4 route table.

```
device# configure terminal
device(config)# router bgp
device(config-bgp)# default-information-originate
```

# Four-byte AS numbers

Four-byte autonomous system numbers (ASNs) can be optionally configured on a device, peer-group, or neighbor. If this is enabled, the device announces and negotiates "AS4" capability with its neighbors.

You can configure AS4 capability to be enabled or disabled either at the BGP global level or at the neighbor or peer-group level.

You can configure AS4 capability to be enabled for a neighbor while still keeping AS4 numbers disabled at the global level, or vice-versa. The neighbor AS4 capability configuration takes precedence. If AS4 capability is not configured on the neighbor, then the peer-group configuration takes effect. The global configuration is used if AS4 capability is configured neither at the neighbor nor at the peer-group level. If a device having a 4-byte ASN tries to connect to a device that does not have AS4 support, peering will not be established.

# Cooperative BGP4 route filtering

By default, the device performs all filtering of incoming routes locally, on the device itself. You can use cooperative BGP4 route filtering to cause the filtering to be performed by a neighbor before it sends the routes to the device. Cooperative filtering conserves resources by eliminating unnecessary route updates and filter processing. For example, the device can send a deny filter to a neighbor, which the neighbor uses to filter out updates before sending them to the device. The neighbor saves the resources it would otherwise use to generate the route updates, and the device saves the resources it would use to filter out the routes.

When you enable cooperative filtering, the device advertises this capability in its Open message to the neighbor when initiating the neighbor session. The Open message also indicates whether the device is configured to send filters, receive filters, or both, and the types of filters it can send or receive. The device sends the filters as Outbound Route Filters (ORFs) in route refresh messages.

To configure cooperative filtering, perform the following tasks on the device and on the BGP4 neighbor:

- Configure the filter.

  NOTE
  Cooperative filtering is currently supported only for filters configured using IP prefix lists.

- Apply the filter as an inbound filter to the neighbor.
- Enable the cooperative route filtering feature on the device. You can enable the device to send ORFs to the neighbor, to receive ORFs from the neighbor, or both. The neighbor uses the ORFs you send as outbound filters when it sends routes to the device. Likewise, the device uses the ORFs it receives from the neighbor as outbound filters when sending routes to the neighbor.
- Reset the BGP4 neighbor session to send and receive ORFs.
- Perform these steps on the other device.

  NOTE
  If the device has inbound filters, the filters are still processed even if equivalent filters have been sent as ORFs to the neighbor.

# BGP4 parameters

Some parameter changes take effect immediately while others do not take full effect until the device sessions with its neighbors are reset. Some parameters do not take effect until the device is rebooted.

The following parameter changes take effect immediately:

- Enable or disable BGP4.
- Set or change the local AS.
- Add neighbors.
- Change the update timer for route changes.
- Disable or enable fast external failover.
- Specify individual networks that can be advertised.
- Change the default local preference, default information originate setting, or administrative distance.
- Enable or disable use of a default route to resolve a BGP4 next-hop route.
- Enable or disable MED (metric) comparison.
- Require the first AS in an update from an EBGP neighbor to be the neighbor AS.

- Change MED comparison parameters.
- Disable comparison of the AS-Path length.
- Enable comparison of the device ID.
- Enable next-hop recursion.
- Change the default metric.
- Disable or re-enable route reflection.
- Configure confederation parameters.
- Disable or re-enable load sharing.
- Change the maximum number of load sharing paths.
- Change other load-sharing parameters.
- Define route flap dampening parameters.
- Add, change, or negate redistribution parameters (except changing the default MED).
- Add, change, or negate route maps (when used by the **network** command or a redistribution command).
- Apply maximum AS path limit settings for UPDATE messages.
- Aggregate routes

The following parameter changes take effect only after the BGP4 sessions on the device are cleared, or reset using the "soft" clear option:

- Change the Hold Time or Keep Alive Time.
- Aggregate routes
- Add, change, or negate filter tables that affect inbound and outbound route policies.
- Apply maximum AS path limit settings to the RIB.

The following parameter change takes effect only after you disable and then re-enable redistribution:

- Change the default MED (metric).

# Route redistribution

The redistribution of static, connected, RIP, ISIS, and OSPF routes into BGP is supported. Similarly, routes learned through BGP can also be redistributed into OSPF.

An optional route-map can be specified, and this map will be consulted before routes are added to BGP. Management routes are not redistributed.

## Redistributing routes into BGP4

Various routes can be redistributed into BGP. This tasks redistributes connected routes into BGP4.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **router bgp** command to enable BGP routing.

```
device(config)# router bgp
```

3. Enter the **redistribute** command using the **connected** keyword to redistribute connected routes.

```
device(config-bgp)# redistribute connected
```

The following example redistributes connected routes into BGP4.

```
device# configure terminal
device(config)# router bgp
device(config-bgp)# redistribute connected
```

# Advertised networks

As previously described, you can advertise routes into BGP by redistributing static, connected, RIP, ISIS, or OSPF routes.

However, you can explicitly specify routes to be advertised by BGP4 by using the **network** command in BGP global configuration mode or IPv4 address-family multicast configuration mode.

With the exception of static network routes, the routing table must have this route already installed before BGP4 can advertise this route. You can also specify a route to be local. If the same route is received by means of eBGP, the local IGP route will be preferred. You can also specify a weight that the device adds to routes that are received from the specified BGP neighbor. BGP4 prefers larger weights over smaller weights.

Refer to the *Extreme NetIron Command Reference* for configuration examples and more information.

# Importing routes into BGP4

Routes can be explicitly specified for advertisement by BGP.

With the exception of static network routes, the routes imported into BGP4 must first exist in the IPv4 multicast or unicast route table.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **router bgp** command to enable BGP routing.

```
device(config)# router bgp
```

3. Enter the **neighbor remote-as** command to specify the ASN in which the remote neighbor resides.

```
device(config-bgp)# neighbor 10.2.2.2 remote-as 1001
```

4. Enter the **network** command and specify a *network/mask* to import the specified prefix into the BGP4 database.

```
device(config-bgp)# network 10.1.1.1/32
```

The following example imports the 10.1.1.1/32 prefix in to the BGP4 database for advertising.

```
device# configure terminal
device(config)# router bgp
device(config-bgp)# 10.2.2.2 remote-as 1001
device(config-bgp)# network network 10.1.1.1/32
```

# Static networks

Before advertising any route, BGP checks for its existence in the routing table. BGP can be configured to advertise a stable route that does not depend on its existence in the routing table.

This allows you to configure a static network in BGP4, creating a stable BGP4 network in the core. While a route configured with this feature will never flap unless it is manually deleted, a "static" BGP4 network will not interrupt the normal BGP4 decision process on other learned routes being installed into the Routing Table Manager (RTM). Consequently, when there is a route that can be resolved, it will be installed into the RTM.

When the configured route is lost, BGP installs the "null0" route in the routing table. Later, when the route is resolved, the null0 route is removed.

NOTE
The BGP4 network route and the BGP4 static network route are mutually exclusive. They cannot be configured with the same prefix and mask.

## Configuring a static network

BGP can be configured to advertise a stable route that does not depend on its existence in the routing table. The following task configures a static network and sets the administrative distance.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **router bgp** command to enable BGP routing.

```
device(config)# router bgp
```

3. Enter the **local-as** command to configure the autonomous system number (ASN) in which your device resides.

```
device(config-bgp)# local-as 1000
```

4. Enter the **static-network** command with the **distance** parameter, and specify a value, to configure a static network and set an administrative distance.

```
device(config-bgp)# static-network 10.11.12.0/32 distance 300
```

The following example configures a static network and sets an administrative distance of 300.

```
device# configure terminal
device(config)# router bgp
device(config-bgp)# local-as 1000
device(config-bgp)# static-network 10.11.12.0/32 distance 300
```

# Route reflection

A BGP device can act as a route-reflector client or as a route reflector. You can configure a BGP peer as a route-reflector client from the device that is going to reflect the routes and act as the route reflector using the **neighbor route-reflector-client** command.

When there is more than one route reflector, they should all belong to the same cluster. By default, the value for **cluster-id** is used as the device ID. The device ID can be changed using the **cluster-id** command.

The route-reflector server reflects the routes as follows:

- Routes from the client are reflected to the client as well as to nonclient peers.
- Routes from nonclient peers are reflected only to client peers.

If route-reflector clients are connected in a full iBGP mesh, you can disable client-to-client reflection on the route reflector using the **no client-to-client-reflection** command.

A BGP device advertises only those routes that are preferred ones and are installed into the Routing Table Manager (RTM). When a route cannot be installed into the RTM because the routing table is full, the route reflector may not reflect that route. In cases where the route reflector is not placed directly in the forwarding path, you can configure the route reflector to reflect routes even though those routes are not in the RTM using the **always-propagate** command.

# Configuring a cluster ID for a route reflector

The cluster ID can be changed if there is more than one route reflector, so that all route reflectors belong to the same cluster.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **router bgp** command to enable BGP routing.

```
device(config)# router bgp
```

3. Enter the **local-as** command to configure the autonomous system number (ASN) in which your device resides.

```
device(config-bgp)# local-as 1000
```

4. Enter the **cluster-id** command and specify a value to change the cluster ID of a device from the default device ID.

```
device(config-bgp)# cluster-id 321
```

The following example changes the cluster ID of a device from the default device ID to 321.

```
device# configure terminal
device(config)# router bgp
device(config-bgp)# cluster-id 321
```

# Configuring a route reflector client

A BGP peer can be configured as a route reflector client.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **router bgp** command to enable BGP routing.

```
device(config)# router bgp
```

3. Enter the **local-as** command to configure the autonomous system number (ASN) in which your device resides.

```
device(config-bgp)# local-as 1000
```

4. Enter the **neighbor route-reflector-client** command to configure a specified neighbor to be a route reflector client.

```
device(config-bgp)# neighbor 10.1.1.1 route-reflector-client
```

The following example configures a neighbor with the IPv4 address 10.1.1.1 to be a route reflector client.

```
device# configure terminal
device(config)# router bgp
device(config-bgp)# local-as 1000
device(config-bgp)# neighbor 10.1.1.1 route-reflector-client
```

# Route flap dampening

A route flap is a change in the state of a route, from up to down or down to up. A route state change causes changes in the route tables of the devices that support the route.

Frequent route state changes can cause Internet instability and add processing overhead to the devices that support the route. Route flap dampening helps reduce the impact of route flap by changing the way a BGP4 device responds to route state changes. When route flap dampening is configured, the device suppresses unstable routes until the number of route state changes drops enough to meet an acceptable degree of stability.

Route flap dampening is disabled by default. You can enable the feature globally or on an individual route basis using route maps.

> NOTE
> The device applies route flap dampening only to routes learned from eBGP neighbors.

The route flap dampening mechanism is based on penalties. When a route exceeds a configured penalty value, the device stops using that route and stops advertising it to other devices. The mechanism also allows route penalties to reduce over time if route stability improves.

# Aggregating routes advertised to BGP neighbors

A device can be configured to aggregate routes in a range of networks into a single IP prefix.

1. Enter the **configure terminal** command to access global configuration mode.

   ```
   device# configure terminal
   ```

2. Enter the **router bgp** command to enable BGP routing.

   ```
   device(config)# router bgp
   ```

3. Enter the **aggregate-address** command to aggregate the routes from a range of networks into a single network prefix.

   ```
   device(config-bgp)# aggregate-address 10.1.1.1/32
   ```

The following example enables a BGP4 device to advertise the default route and send the default route to a specified neighbor.

```
device# configure terminal
device(config)# router bgp
device(config-bgp)# aggregate-address 10.1.1.1/32
```

# Advertising the default BGP4 route

By default, a BGP device does not originate and advertise a default route using BGP4. A BGP4 default route is the IP address 0.0.0.0 and the route prefix 0 or network mask 0.0.0.0. For example, 0.0.0.0/0 is a default route. A BGP device can be configured to advertise the default IPv4 route to all BGP4 neighbors and to install that route in the local BGP4 route table.

The default route must be present in the local IPv4 route table.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **router bgp** command to enable BGP routing.

```
device(config)# router bgp
```

3. Enter the **default-information-originate** command to advertise the default IPv4 route to all BGP4 neighbors and to install that route in the local BGP4 route table.

```
device(config-bgp)# default-information-originate
```

The following example enables a BGP4 device to advertise the default IPv4 route to all BGP4 neighbors and to install that route in the local BGP4 route table.

```
device# configure terminal
device(config)# router bgp
device(config-bgp)# default-information-originate
```

# Advertising the default BGP4 route to a specific neighbor

A BGP device can be configured to advertise the default IPv4 route to a specific neighbor.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **router bgp** command to enable BGP routing.

```
device(config)# router bgp
```

3. Enter the **local-as** command to configure the autonomous system number (ASN) in which your device resides.

```
device(config-bgp)# local-as 1000
```

4. Enter the **neighbor default-originate** command and specify an IP address to enable the BGP4 device to advertise the default IPv4 route to a specific neighbor.

```
device(config-bgp)# neighbor 10.4.4.4 default-originate
```

The following example enables a BGP4 device to advertise the default IPv4 route to a specific neighbor.

```
device# configure terminal
device(config)# router bgp
device(config-bgp)# local-as 1000
device(config-bgp)# neighbor 10.4.4.4 default-originate
```

# Multipath load sharing

Unlike IGP, BGP does not perform multipath load sharing by default. Therefore, the maximum number of paths across which BGP can balance the traffic is set to 1 by default. You can change this value by using the **maximum-paths** command.

By default, when BGP4 multipath load sharing is enabled, both iBGP and eBGP paths are eligible for load sharing, while paths from different neighboring autonomous systems are not eligible. You can change load sharing to apply only to iBGP or eBGP paths, or to support load sharing among paths from different neighboring autonomous systems.

# Specifying the weight added to received routes

The weight that the device adds to received routes can be specified. The following task changes the weight from the default for routes that are received from a specified BGP neighbor.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **router bgp** command to enable BGP routing.

```
device(config)# router bgp
```

3. Enter the **local-as** command to configure the autonomous system number (ASN) in which your device resides.

```
device(config-bgp)# local-as 65520
```

4. Enter the **neighbor weight** command and specify an *ip address* and a weight value to specify a weight that the device adds to routes that are received from the specified BGP4 neighbor.

```
device(config-bgp)# neighbor 10.11.12.13 weight 100
```

The following example specifies a weight of 100 that the device adds to routes that are received from the specified BGP4 neighbor.

```
device# configure terminal
device(config)# router bgp
device(config-bgp)# local-as 65520
device(config-bgp)# neighbor 10.11.12.13 weight 100
```

# Using the IPv4 default route as a valid next hop for a BGP4 route

In certain cases, such as when a device is acting as an edge device, it can be configured to use the default route as a valid next hop.

By default, a device does not use a default route to resolve a BGP4 next-hop route. If the IPv4 route lookup for the BGP4 next-hop does not result in a valid IGP route (including static or direct routes), the BGP4 next-hop is considered to be unreachable and the BGP4 route is not used. You can configure the device to use the default route as a valid next hop.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **router bgp** command to enable BGP routing.

```
device(config)# router bgp
```

3. Enter the **next-hop-enable-default** command to configure the device to use the default route as a valid next hop.

```
device(config-bgp)# next-hop-enable-default
```

The following example configures a BGP4 device to use the default route as a valid next hop.

```
device# configure terminal
device(config)# router bgp
device(config-bgp)# next-hop-enable-default
```

# Adjusting defaults to improve routing performance

The following examples illustrate a variety of options for enabling and fine-tuning route flap dampening.

The following example enables default dampening as an address-family function.

```
device# configure terminal
device(config)# router bgp
device(config-bgp)# dampening
```

The following example changes all dampening values.

```
device# configure terminal
device(config)# router bgp
device(config-bgp)# dampening 20 200 2500 40
```

# Next-hop recursion

For each BGP4 route learned, the device performs a route lookup to obtain the IPv4 address of the next hop for the route. A BGP4 route is eligible for addition in the IPv4 route table only if the following conditions are true:

- The lookup succeeds in obtaining a valid next-hop IPv4 address for the route.
- The path to the next-hop IP address is an IGP path or a static route path.

By default, only one lookup is performed for the next-hop IPv4 address for the BGP4 route. If the next hop lookup does not result in a valid next hop IPv4 address, or the path to the next hop IPv4 address is a BGP4 path, the BGP4 route destination is considered unreachable. The route is not eligible to be added to the IPv4 route table.

The BGP4 route table can contain a route with a next hop IPv4 address that is not reachable through an IGP route, even though the device can reach a hop farther away through an IGP route. This can occur when the IGPs do not learn a complete set of IGP routes, so the device learns about an internal route through iBGP instead of through an IGP. In this case, the IPv4 route table does not contain a route that can be used to reach the BGP4 route destination.

When next-hop recursion is enabled, if the lookup for the next hop IP address results in an iBGP path that originated in the same AS, then the next hop is considered as resolved and BGP4 depended routes are eligible for addition in the IPV4 route table.

## Enabling next-hop recursion

Next hop recursion can be enabled so that a device can find the IGP route to the next hop gateway for a BGP4 route.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **router bgp** command to enable BGP routing.

```
device(config)# router bgp
```

3. Enter the **next-hop-recursion** command to enable recursive next hop lookups.

```
device(config-bgp)# next-hop-recursion
```

The following example enables recursive next hop lookups.

```
device# configure terminal
device(config)# router bgp
device(config-bgp)# next-hop-recursion
```

## Enabling next-hop recursion in the IPv4 multicast address family

Next hop recursion can be enabled so that a device can find the IGP route to the next hop gateway for a BGP4 route.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **router bgp** command to enable BGP routing.

```
device(config)# router bgp
```

3. Enter the **address-family ipv4 multicast** command to enter address family IPv4 multicast configuration mode.

```
device(config-bgp)# address-family ipv4 multicast
```

4. Enter the **next-hop-recursion** command to enable recursive next hop lookups.

```
device(config-bgp-ipv4m)# next-hop-recursion
```

The following example enables recursive next hop lookups.

```
device# configure terminal
device(config)# router bgp
device(config-bgp)# address-family ipv4 multicast
device(config-bgp-ipv4m)# next-hop-recursion
```

# Route filtering

The following route filters are supported:

- AS-path filter
- Community filter
- Prefix list
- Route map

- Table map

NOTE
Support for access lists in route filtering is not available, and has been replaced by prefix-list filtering. BGP does not use community and extended-community filters directly. Rather, it uses them indirectly through route-map filtering by means of the **route-map** command.

# BGP regular expression pattern-matching characters

The following table illustrates the functions of BGP regular expression pattern-matching characters and illustrates their use.

NOTE
The **ip-extcommunity-list** command now supports a range of extended instances, from 100 through 500, beyond the standard range of 1 through 99.

**TABLE 36** BGP regular expression pattern-matching characters

| Regular expression character | Function | Examples |
|---|---|---|
| . | Matches any single character. | 0.0 matches 0x0 and 020 t..t matches strings such as test, text, and tart |
| \ | Matches the character following the backslash. Also matches (escapes) special characters. | 172\.1\.. matches 172.1.10.10 but not 172.12.0.0 \. allows a period to be matched as a period |
| [ ] | Matches the characters or a range of characters separated by a hyphen, within left and right square brackets. | [02468a-z] matches 0, 4, and w, but not 1, 9, or K |
| ^ | Matches the character or null string at the beginning of an input string. | ^123 matches 1234, but not 01234 |
| ? | Matches zero or one occurrence of the pattern. (Precede the question mark with Ctrl-V sequence to prevent it from being interpreted as a help command.) | ba?b matches bb and bab |
| $ | Matches the character or null string at the end of an input string. | 123$ matches 0123, but not 1234 |
| * | Matches zero or more sequences of the character preceding the asterisk. Also acts as a wildcard for matching any number of characters. | 5* matches any occurrence of the number 5 including none 18\..* matches the characters 18. and any characters that follow 18. |
| + | Matches one or more sequences of the character preceding the plus sign. | 8+ requires there to be at least one number 8 in the string to be matched |
| () [] | Nest characters for matching. Separate endpoints of a range with a dash (-). | (17)* matches any number of the two-character string 17 ([A-Za-z][0-9])+ matches one or more instances of letter-digit pairs: b8 and W4, as examples |
| \| | Concatenates constructs. Matches one of the characters or character patterns on either side of the vertical bar. | A(B\|C)D matches ABD and ACD, but not AD, ABCD, ABBD, or ACCD |
| _ | Replaces a long regular expression list by matching a comma (,), left brace ({), right brace (}), the beginning of the input string, the end of the input string, or a space. | The characters _1300_ can match any of the following strings: ^1300$ ^1300space space1300 {1300, ,1300, {1300} ,1300, |

# Timers

The keep alive time specifies how frequently the device sends KEEPALIVE messages to its BGP4 neighbors. The hold time specifies how long the device waits for a KEEPALIVE or UPDATE message from a neighbor before concluding that the neighbor is dead. When the device concludes that a BGP4 neighbor is dead, the device ends the BGP4 session and closes the TCP connection to the neighbor.

A hold-time value of 0 means that the device waits indefinitely for messages from a neighbor without tearing down the session.

NOTE
Generally, you should set the hold time to three times the value of the keep alive time.

NOTE
You can override the global keep alive time and hold time on individual neighbors.

# Enabling BGP4 in a non-default VRF

When BGP4 is enabled in a non-default VRF instance, the device enters BGP address-family IPv4 unicast VRF configuration mode. Several commands can then be accessed that allow the configuration of BGP4 for the non-default VRF instance.

A non-default VRF instance has been configured.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **router bgp** command to enable BGP routing globally.

```
device(config)# router bgp
```

3. Enter the **address-family ipv4 unicast** command with the **vrf** parameter, and specify a VRF name, to enter BGP address-family IPv4 unicast VRF configuration mode.

```
device(config-bgp)# address-family ipv4 unicast vrf red
```

The following example enables BGP address-family IPv4 unicast VRF configuration mode where several commands can be accessed that allow the configuration of BGP4 for the non-default VRF instance..

```
device# configure terminal
device(config)# router bgp
device(config-bgp)# address-family ipv4 unicast vrf red
device(config-bgp-ipv4u-vrf)#
```

# BGP4 outbound route filtering

The BGP4 Outbound Route Filtering Capability (ORF) feature is used to minimize the number of BGP updates sent between BGP peers.

When the ORF feature is enabled, unwanted routing updates are filtered out, reducing the amount of system resources required for generating and processing routing updates. The ORF feature is enabled through the advertisement of ORF capabilities to peer routers. The locally configured BGP4 inbound prefix filters are sent to the remote peer so that the remote peer applies the filter as an outbound filter for the neighbor.

The ORF feature can be configured with send and receive ORF capabilities. The local peer advertises the ORF capability in send mode, indicating that it will accept a prefix list from a neighbor and apply the prefix list to locally configured ORFs. The local peer exchanges the

ORF capability in send mode with a remote peer for a prefix list that is configured as an inbound filter for that peer locally. The remote peer only sends the first update once it receives a ROUTEREFRESH request or BGP ORF with IMMEDIATE from the peer. The local and remote peers exchange updates to maintain the ORF on each router.

# Configuring BGP4 outbound route filtering

The BGP4 Outbound Route Filtering (ORF) prefix list capability can be configured in receive mode, send mode, or both send and receive modes, minimizing the number of BGP updates exchanged between BGP peers.

1. Enter the **configure terminal** command to access global configuration mode.

   ```
   device# configure terminal
   ```

2. Enter the **router bgp** command to enable BGP routing.

   ```
   device(config)# router bgp
   ```

3. Enter the **neighbor prefix-list** command and specify the **in** keyword to filter the incoming route updates from a specified BGP neighbor.

   ```
   device(config-bgp)# neighbor 10.1.2.3 prefix-list myprefixlist in
   ```

4. Do one of the following:
   - Enter the **neighbor capability orf prefixlist** command and specify the **send** keyword to advertise ORF send capabilities.

     ```
     device(config-bgp)# neighbor 10.1.2.3 capability orf prefixlist send
     ```

   - Enter the **neighbor capability orf prefixlist** command and specify the **receive** keyword to advertise ORF receive capabilities.

     ```
     device(config-bgp)# neighbor 10.1.2.3 capability orf prefixlist receive
     ```

   - Enter the **neighbor capability orf prefixlist** command to configure ORF capability in both send and receive modes.

     ```
     device(config-bgp)# neighbor 10.1.2.3 capability orf prefixlist
     ```

The following example configures ORF in receive mode.

```
device# configure terminal
device(config)# router bgp
device(config-bgp)# neighbor 10.1.2.3 capability orf prefixlist receive
```

The following example configures ORF in send mode.

```
device# configure terminal
device(config)# router bgp
device(config-bgp)# neighbor 10.1.2.3 prefix-list myprefixlist in
device(config-bgp)# neighbor 10.1.2.3 capability orf prefixlist send
```

The following example configures ORF in both send and receive modes.

```
device# configure terminal
device(config)# router bgp
device(config-bgp)# neighbor 10.1.2.3 prefix-list myprefixlist in
device(config-bgp)# neighbor 10.1.2.3 capability orf prefixlist
```

## Enabling BGP4 cooperative route filtering

You can use cooperative BGP4 route filtering to cause the filtering to be performed by a neighbor before it sends the routes to the device, conserving resources by eliminating unnecessary route updates and filter processing. The following task enables cooperative route filtering.

NOTE
The current release supports cooperative filtering only for filters configured using IP prefix lists.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **ip prefix-list** command to to configure the IP prefix list instance.

```
device(config)# ip prefix-list Routesfrom10234 deny 10.20.0.0/24
device(config)# ip prefix-list Routesfrom10234 permit 10.0.0.0/0 le 32
```

3. Enter the **router bgp** command to enable BGP routing.

```
device(config)# router bgp
```

4. Enter the **neighbor prefix-list** command with the **in** parameter and specify a prefix-list to filter the incoming route updates from the specified BGP neighbor.

```
device(config-bgp)# neighbor 10.2.3.4 prefix-list Routesfrom1234 in
```

5. Enter the **capability orf prefixlist** command with the **send** parameter to enable the ORF prefix list capability in send mode.

```
device(config-bgp)# neighbor 10.2.3.4 capability orf prefixlist send
```

The following example enables BGP4 cooperative route filtering.

```
device# configure terminal
device(config)# ip prefix-list Routesfrom10234 deny 10.20.0.0/24
device(config)# ip prefix-list Routesfrom10234 permit 10.0.0.0/0 le 32
device(config)# router bgp
device(config-bgp)# neighbor 10.2.3.4 prefix-list Routesfrom1234 in
device(config-bgp)# neighbor 10.2.3.4 capability orf prefixlist send
```

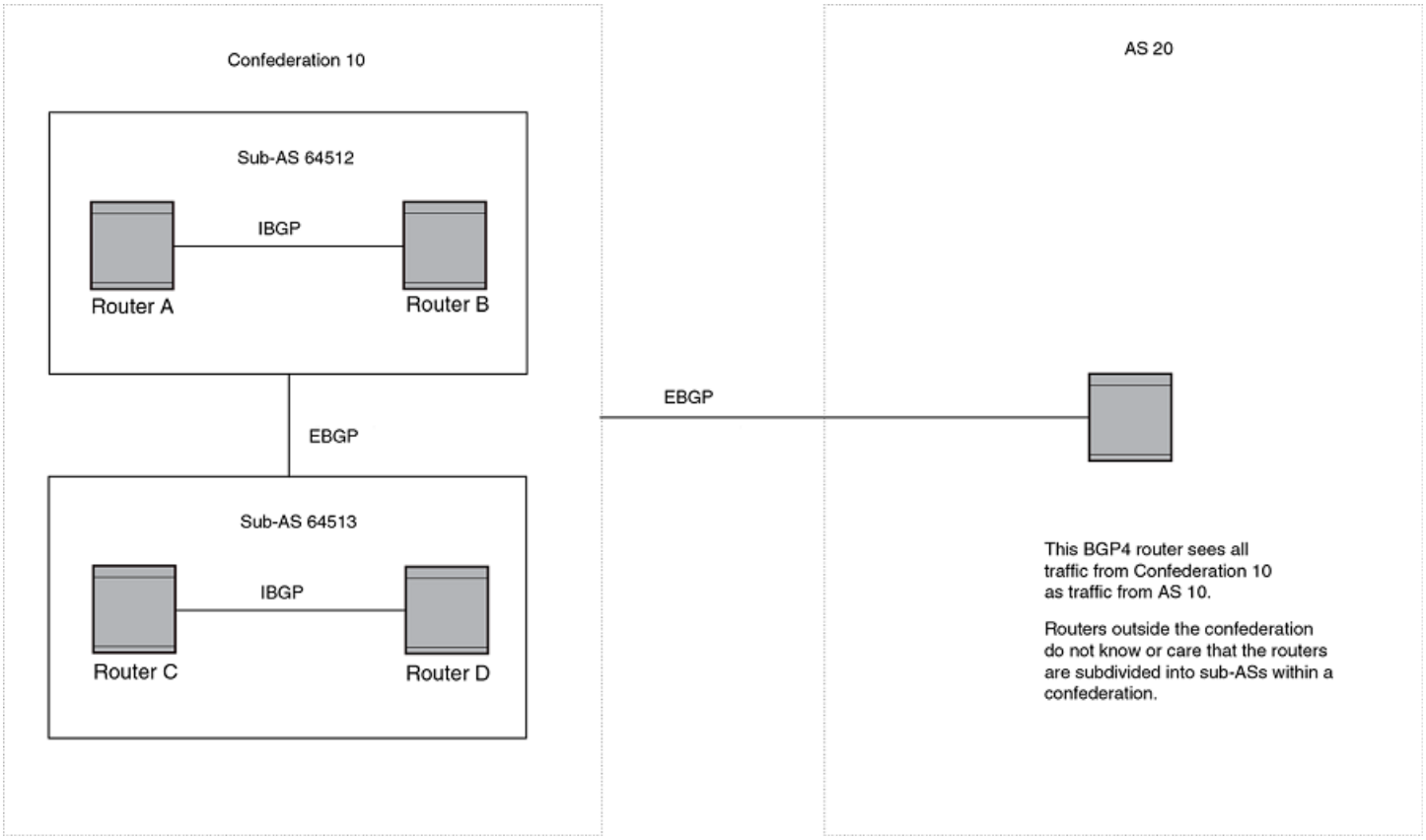
# BGP4 confederations

A large autonomous system (AS) can be divided into multiple subautonomous systems and grouped into a single BGP4 confederation.

Each subautonomous system must be uniquely identified within the confederation AS by a subautonomous system number. Within each subautonomous system, all the rules of internal BGP (iBGP) apply. For example, all BGP routers inside the subautonomous system must be fully meshed. Although eBGP is used between subautonomous systems, the subautonomous systems within the confederation exchange routing information like iBGP peers. Next hop, Multi Exit Discriminator (MED), and local preference information is preserved when crossing subautonomous system boundaries. To the outside world, a confederation looks like a single AS.

The AS path list is a loop-avoidance mechanism used to detect routing updates leaving one subautonomous system and attempting to re-enter the same subautonomous system. A routing update attempting to re-enter a subautonomous system it originated from is detected because the subautonomous system sees its own subautonomous system number listed in the update's AS path.

**FIGURE 31** Example BGP4 confederation




In this example, four devices are configured into two sub-autonomous systems, each containing two of the devices. The sub-autonomous systems are members of confederation 10. Devices within a sub-AS must be fully meshed and communicate using iBGP. In this example, devices A and B use iBGP to communicate. devices C and D also use IBGP. However, the sub-autonomous systems communicate with one another using eBGP. For example, device A communicates with device C using eBGP. The devices in the confederation communicate with other autonomous systems using eBGP.

Devices in other autonomous systems are unaware that devices A through D are configured in a confederation. In fact, when devices in confederation 10 send traffic to devices in other autonomous systems, the confederation ID is the same as the AS number for the devices in the confederation. Thus, devices in other autonomous systems see traffic as coming from AS 10 and are unaware that the devices in AS 10 are subdivided into sub-autonomous systems within a confederation.

# Configuring BGP4 confederations

BGP4 confederations, composed of multiple subautonomous systems, can be created.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **router bgp** command to enable BGP routing.

```
device(config)# router bgp
```

3. Enter the **local-as** command to configure the autonomous system number (ASN) in which your device resides.

```
device(config-bgp)# local-as 65520
```

4. Enter the **confederation identifier** command and specify an ASN to configure a BGP confederation identifier.

```
device(config-bgp)# confederation identifier 100
```

5. Enter the **confederation peers** command and specify as many ASNs as needed to list all BGP peers that will belong to the confederation.

```
device(config-bgp)# confederation peers 65520 65521 65522
```

The following example creates a confederation with the confederation ID "100" and adds three subautonomous systems to the confederation.

```
device# configure terminal
device(config)# router bgp
device(config-bgp)# local-as 65520
device(config-bgp)# confederation identifier 100
device(config-bgp)# confederation peers 65520 65521 65522
```

# BGP community and extended community

A BGP community is a group of destinations that share a common property. Community information identifying community members is included as a path attribute in BGP UPDATE messages. You can perform actions on a group using community and extended community attributes to trigger routing decisions.

All communities of a particular type can be filtered out, or certain values can be specified for a particular type of community. You can also specify whether a particular community is transitive or non-transitive across an autonomous system (AS) boundary.

An extended community is an 8-octet value and provides a larger range for grouping or categorizing communities. BGP extended community attributes are specified in RFC 4360.

You define the extended community list using the **ip extcommunity-list** command. The extended community can then be matched or applied to the neighbor through the route map. The route map must be applied on the neighbor to which routes need to carry the extended community attributes. The "send-community" should be enabled for the neighbor configuration to start including the attributes while sending updates to the neighbor.

## Applying a BGP4 extended community filter

A BGP4 extended community filter can be applied.

BGP4 communities must already be defined.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **ip extcommunity-list** command and specify a number to set a BGP extended community filter.

```
device(config)# ip extcommunity-list 1 permit rt 123:2
```

3. Enter the **router bgp** command to enable BGP routing.

```
device(config)# router bgp
```

4. Enter the **local-as** command to configure the autonomous system number (ASN) in which your device resides.

```
device(config-bgp)# local-as 1000
```

5. Enter the **neighbor** *ip-address* **remote-as** command to specify the ASN in which the remote neighbor resides.

```
device(config-bgp)# neighbor 10.1.2.3 remote-as 1001
```

6. Enter the **neighbor** *ip-address* **activate** command to enable the exchange of information with the neighbor.

```
device(config-bgp)# neighbor 10.1.2.3 activate
```

7. Enter the **neighbor** *ip-address* **route-map** command and specify the **in** keyword to apply a route map to incoming routes.

```
device(config-bgp)# neighbor 10.1.2.3 route-map in extComRmapt
```

8. Enter the **neighbor** *ip-address* **route-map** command and specify the **out** keyword to apply a route map to outgoing routes.

```
device(config-bgp)# neighbor 10.1.2.3 route-map out sendExtComRmap
```

9. Enter the **neighbor** *ip-address* **send-community** command and specify the **both** keyword to enable the sending of standard and extended attributes in updates to the specified BGP neighbor.

```
device(config-bgp)# neighbor 10.1.2.3 send-community both
```

The following example applies a BGP4 extended community filter.

```
device# configure terminal
device(config)# ip extcommunity-list 1 permit rt 123:2
device(config)# router bgp
device(config-bgp)# local-as 1000
device(config-bgp)# neighbor 10.1.2.3 remote-as 1001
device(config-bgp)# neighbor 10.1.2.3 activate
device(config-bgp)# neighbor 10.1.2.3 route-map in extComRmapt
device(config-bgp)# neighbor 10.1.2.3 route-map out sendExtComRmap
device(config-bgp)# neighbor 10.1.2.3 send-community both
```

# BGP Large Communities

The BGP Large Communities attribute (RFC 8092) supports an optional transitive path attribute of variable length, overcoming previous length limitations. All routes with this attribute belong to the communities specified in the attribute.

## Introduction

BGP [RFC 4271] implementations typically support a routing policy language to control the distribution of routing information. Network operators attach BGP communities to routes to associate particular properties with these routes.

A BGP Communities attribute (RFC 1997) is a variable-length attribute consisting of a set of one or more four-octet values, each of which specifies a community. A common use of the individual values of this attribute type splits this single 32-bit value into two 16-bit values. The most significant word is interpreted as an Autonomous System Number (ASN), and the least significant word is a locally defined value whose meaning is assigned by the operator of the AS in the most significant word.

Since the adoption of four-octet ASNs, the BGP Communities attribute can no longer accommodate the above encoding, as a two-octet word cannot fit within a four-octet ASN.

To address these shortcomings, a BGP Large Communities attribute is encoded as an unordered set of one or more 12-octet values, each consisting of a four-octet Global Administrator field and two four-octet operator-defined fields, each of which can be used to denote properties or actions significant to the operator of the AS assigning the values.

# BGP Large Communities attribute details

Each BGP Large Community value is encoded as a 12-octet quantity, as follows:

```
0 1 2 3 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          Global Administrator                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           Local Data Part 1                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           Local Data Part 2                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

| Field | Description |
|---|---|
| Global Administrator | A four-octet namespace identifier. Allows different ASes to define BGP Large Communities without collision. This field SHOULD be an ASN, in which case the Local Data Parts are to be interpreted as defined by the owner of the ASN. |
| Local Data Part 1 | A four-octet operator-defined value. Represents a function identifier. |
| Local Data Part 2 | A four-octet operator-defined value. Represents a parameter value. |

**ATTENTION**
The use of Reserved ASNs (0 [RFC 7607], 65535, and 4294967295 [RFC 7300] is NOT RECOMMENDED.

There is no significance to the order in which twelve-octet Large Community Attribute values are encoded in a Large Communities attribute, A BGP speaker can transmit them in any order.

In canonical notation, each Large Community value can be summarized as "ASN:Function:Parameter". The function of a community can be divided into two categories for the treatment of routes: Informational Communities and Action Communities.

Informational Communities are labels for attributes such as the origin of the route announcement, the nature of the relation with an External BGP (EBGP) neighbor, or the intended propagation audience.

Action Communities are added as labels to request that a route be treated in a particular way within an AS.

More information on the usage perspective of BGP Large Communities can be obtained from RFC 8195 (Informational RFC) which could serve as an informational reference to real-world applications for BGP Large Communities.

# BGP Large Community configuration examples

This section presents a variety of configuration examples.

## *Standard community list*

The following example creates a standard large-community list.

```
device# configure terminal
device(config)# ip large-community-list standard my_std_list seq 10 permit 64497:1:528
```

## Extended community list

The following example creates an extended community list with a regular expression.

```
device# configure terminal
device(config)# ip large-community-list extended my_ext_list seq 10 permit 64497:*:*
```

## ACL matching for a route map

The following example shows how to configure matching based on a large-community access list named ABCPath for a route map named myroutes.

```
device# config terminal
device(config)# route-map myroutes permit 10
device(config-route-map myroutes/permit/10)# match large-community-list ABCPath
```

The following example shows how to configure matching based on a large-community access list named lcstdacl1 with an exact match for a route map named myroutes.

```
device# config terminal
device(config)# route-map myroutes permit 10
device(config-route-map myroutes/permit/10)# match large-community-list lcstdacl1 exact-match
```

## Setting a large community list

The following example sets a large-community list in a route-map instance.

```
device# configure terminal
device(config)# route-map myroutes permit 10
device(config-route-map-myroutes/permit/10)# set large-community 64497:1:528
```

The following example sets a large-community list in a route-map instance and appends updates to existing attributes.

```
device# configure terminal
device(config)# route-map myroutes permit 10
device(config-route-map-myroutes/permit/10)# set large-community 64497:1:528 additive
```

## Setting a large community list for deletion

The following example sets a community list for deletion in a route-map instance.

```
device# configure terminal
device(config)# route-map myroutes permit 10
device(config-route-map-myroutes/permit/10)# set large-community-list mylargecommunitylist delete
```

## Sending large community attributes

The following example sends standard community attributes to a neighbor.

```
device# configure terminal
device(config)# router bgp
device(config-bgp)# neighbor 10.11.12.13 send-community large
```

The following example adds the configured route-map for a peer route-map out.

```
device# configure terminal
device(config)# router bgp
device(config-bgp)# neighbor 10.11.12.13 route-map out myroutes
```

# BGP4 graceful restart

BGP4 graceful restart (GR) allows for restarts where BGP neighboring devices participate in the restart, helping to ensure that no route and topology changes occur in the network for the duration of the restart.

The GR feature provides a routing device with the capability to inform its neighbors when it is performing a restart.

When a BGP session is established, GR capability for BGP is negotiated by neighbors through the BGP OPEN message. If the neighbor also advertises support for GR, GR is activated for that neighbor session. If neither peer exchanges the GR capability, the session is not GR-capable. If the BGP session is lost, the BGP peer router, known as a GR helper, marks all routes associated with the device as "stale" but continues to forward packets to these routes for a set period of time. The restarting device also continues to forward packets for the duration of the graceful restart. When the graceful restart is complete, routes are obtained from the helper so that the device is able to quickly resume full operation.

When the GR feature is configured on a device, both helper router and restarting router functionalities are supported. It is not possible to disable helper functionality explicitly.

GR is disabled by default and can be enabled for both IPv4 and IPv6 address families. When the GR timer expires, the BGP RASlog message is triggered.

> NOTE
> BGP4 GR can be configured for a global routing instance or for a specified VRF instance.

> NOTE
> BGP4 GR is fully supported by MLX Series and XMR Series devices. The CER 2000 Series and CES 2000 Series devices only support GR helper mode.

## Configuring BGP4 graceful restart

The graceful restart (GR) feature can be configured on a routing device, providing it with the capability to inform its neighbors when it is performing a restart.

> NOTE
> High availability (HA) requires GR to be enabled.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **router bgp** command to enable BGP routing.

```
device(config)# router bgp
```

3. Enter the **local-as** command to configure the autonomous system number (ASN) in which your device resides.

```
device(config-bgp)# local-as 1000
```

4. Enter the **neighbor** *ip address* **remote-as** command to add a neighbor.

```
device(config-bgp)# neighbor 10.11.12.13 remote-as 2
```

5. Enter the **graceful-restart** command to enable the graceful restart feature.

```
device(config-bgp)# neighbor 10.11.12.13 remote-as 2
```

6. Do any of the following:
   - Enter the **graceful-restart** command and use **purge-time** parameter to overwrite the default purge-time value.

     ```
     device(config-bgp)# graceful-restart purge-time 300
     ```

   - Enter the **graceful-restart** command and use **restart-time** parameter to overwrite the default restart-time advertised to graceful restart-capable neighbors.

     ```
     device(config-bgp)# neighbor graceful-restart restart-time 180
     ```

   - Enter the **graceful-restart** command and use **stale-routes-time** parameter to overwrite the default amount of time that a helper device will wait for an EOR message from a peer.

     ```
     device(config-bgp)# graceful-restart stale-routes-time 100
     ```

The following example enables the GR feature.

```
device# configure terminal
device(config)# router bgp
device(config-bgp)# local-as 1
device(config-bgp)# neighbor 10.11.12.13 remote-as 2
device(config-bgp)# graceful-restart
```

The following example enables the GR feature and sets the purge time to 300 seconds, over-writing the default value.

```
device# configure terminal
device(config)# router bgp
device(config-bgp)# local-as 1
device(config-bgp)# neighbor 10.11.12.13 remote-as 2
device(config-bgp)# graceful-restart purge-time 180
```

The following example enables the GR feature and sets the restart time to 180 seconds, over-writing the default value.

```
device# configure terminal
device(config)# router bgp
device(config-bgp)# local-as 1
device(config-bgp)# neighbor 10.11.12.13 remote-as 2
device(config-bgp)# graceful-restart restart-time 180
```

The following example enables the GR feature and sets the stale-routes time to 100 seconds, over-writing the default value.

```
device# configure terminal
device(config)# router bgp
device(config-bgp)# local-as 1
device(config-bgp)# neighbor 10.11.12.13 remote-as 2
device(config-bgp)# graceful-restart stale-routes-time 100
```

Use the **clear ip bgp neighbor** command with the **all** parameter for the changes to the GR parameters to take effect immediately.

# BGP additional-paths overview

BGP additional-paths provides the ability for multiple paths for the same prefix to be advertised without the new paths implicitly replacing the previous paths. Path diversity is achieved rather than path hiding.

BGP devices generally advertise only their best path to neighboring devices, even when multiple paths exist. The advertisement of the same prefix from the same neighbor replaces the previous announcement of that prefix. This is known as an implicit withdraw, behavior that achieves better scaling but at the cost of path diversity.

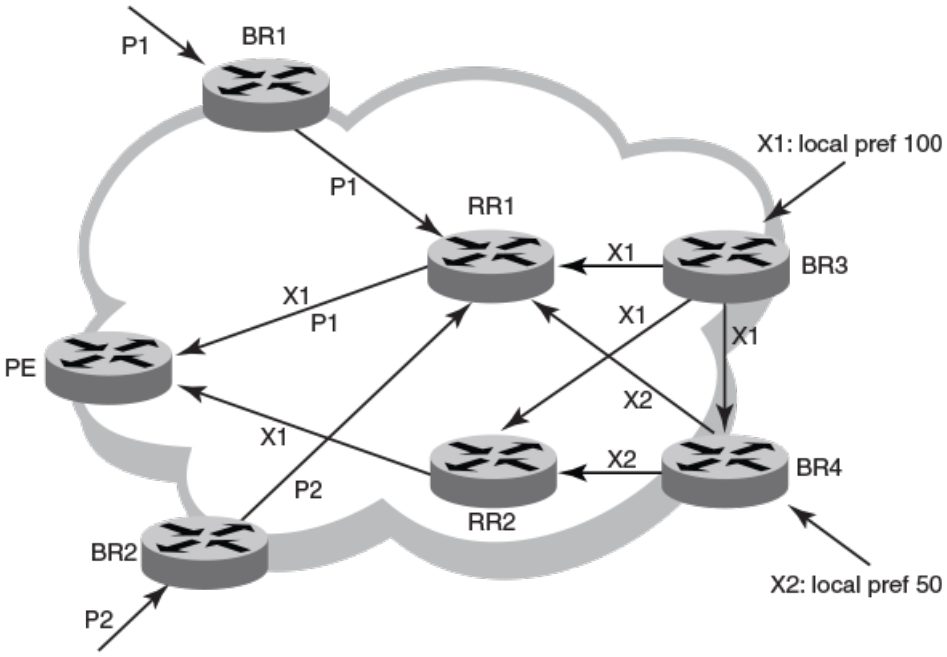
Path hiding can affect the efficient use of BGP multipath and path diversity, and prevent hitless planned maintenance. Upon next hop failures, path hiding also inhibits fast and local recovery because the network must wait for BGP control plane convergence to restore

traffic. BGP additional-paths enables BGP to advertise even the secondary best routes so that multiple paths for the same prefix can be advertised without the new paths implicitly replacing previous paths. BGP additional-paths provides a generic way of offering path diversity.

In the following figure, path hiding occurs in two ways:

- Prefix P has paths P1 and P2 advertised from BR1 and BR2 to RR1. RR1 selects P1 as the best path and advertises only P1 to PE.
- Prefix X has path X1 advertised from BR3 to BR4 with local preference 100. BR4 also has path X2. However, only the best path, X1, is selected. BR3 advertises X1 to the RRs and X2 is suppressed.

**FIGURE 32** BGP path hiding




## Advantages of BGP additional-paths

- Fast convergence and fault tolerance: When BGP additional-paths is enabled, more than one path to a destination is advertised. If one of the paths goes down, connectivity is easily restored due to the availability of backup paths. If the next hop for the prefix becomes unreachable, the device can switch to the backup route immediately without having to wait for BGP control plane messages.
- Enhanced load balancing capabilities: Traditionally with RRs in an iBGP domain, only the best path is given to the clients, even if ECMP paths exist. This affects load balancing. With additional paths advertised by RRs, the clients have more effective load balancing.

## Considerations and limitations for BGP additional-paths RIB-in

- When BGP additional-paths is not configured, only one NLRI per prefix per peer is supported. Any additional NLRI update for the same prefix from the same peer replaces the existing one.

- When BGP additional-paths is configured, a device can receive multiple NLRI advertisements for the same prefix from the same peer that are uniquely identified by NLRI path identifiers.
- For MLX Series and XMR Series devices, the maximum number of additional paths per peer per prefix is 128.
- For CER 2000 Series and CES 2000 Series devices, the maximum number of additional paths per peer per prefix is 64.
- Changes in the capability of sending or receiving additional paths are reflected only after the BGP session is restarted.

## Considerations and limitations for BGP additional-paths RIB-out

- Changes in the capability of sending or receiving additional paths are reflected only after the BGP session is restarted.
- The maximum number of paths that can be advertised per prefix is 16. If there are more than 16 paths for a prefix in the RIB-in, only 16 can be advertised.
- You should maintain the number of RIB-in paths for any prefix in the range of 16 for smooth RIB-out processing. Otherwise the RIB-out processing time increases exponentially in the scaled scenarios.

## Upgrade and downgrade considerations

If BGP additional-paths is enabled and the configuration saved, an error message occurs if the software is downgraded to an earlier version. BGP additional-paths should be unconfigured before a downgrade take place.

## BGP additional-paths functionality

BGP additional-paths is implemented by including an additional four-octet value known as a path identifier (ID) for each path in the NLRI. Path IDs are unique to a peering session and are generated for each network. A generated Path ID is unique per peer per prefix. A BGP device can receive the same path ID for the same prefix from two different peers, or it can receive the same path ID from the same peer for two different prefixes.

A path ID can apply to the IPv4 or IPv6 unicast or multicast address family.

Therefore, when the same prefix is received with the same path ID from the same peer, it is considered as a replacement route or a duplicate route. When the same prefix is received with a different path ID from the same peer, it is considered as an additional path to the prefix.

To send or receive additional paths, the additional-paths capability must be negotiated. If it is not negotiated, only the best path can be sent. BGP updates carry the path ID once the additional-paths capability is negotiated. In order to carry the path ID in an update message, the existing NLRI encodings are extended by prepending the path ID field, which consists of four octets.

The assignment of the path ID for a path by a BGP device occurs in such a way that the BGP device is able to use the prefix and path ID to uniquely identify a path advertised to a neighbor so as to continue to send further updates for that path. The receiving BGP neighbor that re-advertises a route regenerates its own path ID to be associated with the re-advertised route.

The set of additional paths advertised to each neighbor can be different, and advertisement filters are provided on a per-neighbor basis.

NOTE
BGP additional-paths is supported for the BGP IPv4 and IPv6 unicast address families and the BGP IPv4 and IPv6 multicast address families.

NOTE
BGP additional-paths is not supported for the BGP L2VPN VPLS, BGP VPNv4 unicast, and BGP VPNv6 address families.

There are three basic steps involved in configuring BGP additional-path:

- **Capability Negotiation**: Specify whether the device can send, receive, or send and receive additional paths. This is done at the address family level or peer-group level or the neighbor level. Refer to the sections and the NetIron Command Reference for more information.
- **Select Candidate paths**: Select a set or sets of candidate paths for advertisement by specifying selection criteria. This is done at the address family level.
- **Advertise additional paths from the candidate set**: Advertise to a neighbor a set or sets of additional paths from the candidate paths marked. This is done at the neighbor level or peer-group level.

# Configuring BGP4 additional-paths and additional-path selection for the default VRF

You can enable BGP additional-paths send and receive capability for the IPv4 address family and select a set or sets of candidate paths for advertisement by specifying the selection criteria. This task specifies that all BGP paths are eligible to be selected as additional paths under the IPv4 unicast address family for the default VRF and enables BGP additional-paths.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **router bgp** command to enable BGP routing.

```
device(config)# router bgp
```

3. Enter the **local-as** command to configure the autonomous system number (ASN) in which your device resides.

```
device(config-bgp)# local-as 65520
```

4. Enter the **additional-paths** command, using the **receive** parameter, to enable additional-paths receive capability under the IPv4 unicast address family.

```
device(config-bgp)# additional-paths receive
```

5. Enter the **additional-paths** command, using the **send** parameter, to enable additional-paths send capability under the IPv4 unicast address family.

```
device(config-bgp)# additional-paths send
```

6. Enter the **additional-paths select** command, using the **all** parameter, to specify that all BGP paths are eligible to be selected as additional paths under the IPv4 unicast address family.

```
device(config-bgp)# additional-paths select all
```

7. Enter the **neighbor additional-paths advertise** command, specifying an IP address and using the **all** parameter, to configure BGP to advertise all BGP additional paths to a neighbor.

```
device(config-bgp)# neighbor 10.11.12.13 additional-paths advertise all
```

The following example enables BGP additional-paths send and receive capability and specifies that all BGP paths are eligible to be selected as additional paths under the IPv4 unicast address family. The additional-paths feature is enabled and all paths can be advertised to a BGP neighbor.

```
device# configure terminal
device(config)# router bgp
device(config-bgp)# local-as 65520
device(config-bgp)# additional-paths receive
device(config-bgp)# additional-paths send
device(config-bgp)# additional-paths select all
device(config-bgp)# neighbor 10.11.12.13 additional-paths advertise all
```

# Configuring BGP4 additional-paths and additional-path selection for a non-default VRF instance

You can enable BGP additional-paths send and receive capability under the configured IPv4 address family for a non-default VRF instance and select a set or sets of candidate paths for advertisement by specifying the selection criteria. This task specifies that all BGP paths are eligible to be selected as additional paths under the IPv4 unicast address family for VRF green and enables BGP additional-paths.

1. Enter the **configure terminal** command to access global configuration mode.

   ```
   device# configure terminal
   ```

2. Enter the **router bgp** command to enable BGP routing.

   ```
   device(config)# router bgp
   ```

3. Enter the **local-as** command to configure the autonomous system number (ASN) in which your device resides.

   ```
   device(config-bgp)# local-as 65520
   ```

4. Enter the **address-family unicast ipv4** command, using the **vrf** parameter, and specifying a VRF, to enter BGP address-family IPv4 unicast VRF configuration mode.

   ```
   device(config-bgp)# address-family ipv4 unicast vrf green
   ```

5. Enter the **additional-paths** command, using the **receive** parameter, to enable additional-paths receive capability under the configured IPv4 address family for VRF instance green.

   ```
   device(config-bgp)# additional-paths receive
   ```

6. Enter the **additional-paths** command, using the **send** parameter, to enable additional-paths send capability under the configured IPv4 address family for VRF instance green.

   ```
   device(config-bgp)# additional-paths send
   ```

7. Enter the **additional-paths select** command, using the **all** parameter, to specify that all BGP paths are eligible to be selected as additional paths under the IPv4 unicast address family for VRF instance green.

   ```
   device(config-bgp-ipv4u-vrf)# additional-paths select all
   ```

8. Enter the **neighbor additional-paths advertise** command, specifying an IP address and using the **all** parameter, to configure BGP to advertise all BGP additional paths to a neighbor for VRF instance green.

   ```
   device(config-bgp)# neighbor 10.11.12.13 additional-paths advertise all
   ```

The following example enables BGP additional-paths send and receive capability and specifies that all BGP paths are eligible to be selected as additional paths under the IPv4 unicast address family for VRF instance green. The additional-paths feature is enabled and all paths can be advertised to a BGP neighbor.

```
device# configure terminal
device(config)# router bgp
device(config-bgp)# local-as 65520
device(config-bgp)# address-family ipv4 unicast vrf green
device(config-bgp-ipv4u-vrf)# additional-paths receive
device(config-bgp-ipv4u-vrf)# additional-paths send
device(config-bgp-ipv4u-vrf)# additional-paths select all
device(config-bgp-ipv4u-vrf)# neighbor 10.11.12.13 additional-paths advertise all
```

# Configuring BGP4 additional-paths for a specified neighbor

You can apply filters for the advertisement of additional paths for specified BGP neighbors.

1. Enter the **configure terminal** command to access global configuration mode.

   ```
   device# configure terminal
   ```

2. Enter the **router bgp** command to enable BGP routing.

   ```
   device(config)# router bgp
   ```

3. Enter the **local-as** command to configure the autonomous system number (ASN) in which your device resides.

   ```
   device(config-bgp)# local-as 65520
   ```

4. Enter the **address-family ipv4 unicast** command to enter BGP address-family IPv4 multicast configuration mode.

   ```
   device(config-bgp)# address-family ipv4 multicast
   ```

5. Enter the **neighbor additional-paths** command, specifying an IP address and using the **receive** parameter, to enable additional-paths receive capability from a specified BGP neighbor.

   ```
   device(config-bgp-ipv4m)# neighbor 10.11.12.13 additional-paths receive
   ```

6. Enter the **neighbor additional-paths** command, specifying an IP address and using the **send** parameter, to enable additional-paths send capability to a specified BGP neighbor.

   ```
   device(config-bgp-ipv4m)# neighbor 10.11.12.13 additional-paths send
   ```

7. Enter the **additional-paths select** command, using the **all** parameter, to specify that all BGP paths are eligible to be selected as additional paths under the configured IPv4 address family.

   ```
   device(config-bgp-ipv4m)# additional-paths select all
   ```

8. Enter the **neighbor additional-paths advertise** command, specifying an IP address and using the **all** parameter, to configure BGP to advertise all BGP additional paths.

   ```
   device(config-bgp-ipv4m)# neighbor 10.11.12.13 additional-paths advertise all
   ```

The following example specifies that all BGP paths are eligible to be selected as additional paths under the IPv4 multicast address family and enables the capability to send and receive additional paths for a specified neighbor. The additional-paths feature is enabled and all paths can be advertised to the BGP neighbor.

```
device# configure terminal
device(config)# router bgp
device(config-bgp)# local-as 65520
device(config-bgp)# address-family ipv4 multicast
device(config-bgp-ipv4m)# neighbor 10.11.12.13 additional-paths receive
device(config-bgp-ipv4m)# neighbor 10.11.12.13 additional-paths send
device(config-bgp-ipv4m)# additional-paths select all
device(config-bgp-ipv4m)# neighbor 10.11.12.13 additional-paths advertise all
```

# Configuring BGP additional-paths for a specified BGP4 neighbor for a non-default VRF instance

You can enable the advertisement of additional paths for specified BGP4 neighbors and apply filters for the advertisement of additional paths for BGP neighbors for a non-default VRF instance.

1. Enter the **configure terminal** command to access global configuration mode.

   ```
   device# configure terminal
   ```

2. Enter the **router bgp** command to enable BGP routing.

   ```
   device(config)# router bgp
   ```

3. Enter the **local-as** command to configure the autonomous system number (ASN) in which your device resides.

   ```
   device(config-bgp)# local-as 65520
   ```

4. Enter the **address-family unicast ipv4** command, using the **vrf** parameter and specifying a VRF, to enter BGP address-family IPv4 unicast VRF configuration mode.

   ```
   device(config-bgp)# address-family ipv4 unicast vrf green
   ```

5. Enter the **neighbor additional-paths** command, specifying an IP address and using the **receive** parameter, to enable additional-paths receive capability from a specified BGP neighbor for VRF green.

   ```
   device(config-bgp-ipv4u-vrf)# neighbor 10.11.12.13 additional-paths receive
   ```

6. Enter the **neighbor additional-paths** command, specifying an IP address and using the **send** parameter, to enable additional-paths send capability to a specified BGP neighbor for VRF green.

   ```
   device(config-bgp-ipv4u-vrf)# neighbor 10.11.12.13 additional-paths send
   ```

7. Enter the **additional-paths select** command, using the **all** parameter, to specify that all BGP paths are eligible to be selected as additional paths under the IPv4 address unicast family for VRF green.

   ```
   device(config-bgp-ipv4u-vrf)# additional-paths select all
   ```

8. Enter the **neighbor additional-paths advertise** command, specifying an IP address and using the **all** parameter, to configure BGP to advertise all BGP additional paths for VRF green.

   ```
   device(config-bgp-ipv4u-vrf)# neighbor 10.11.12.13 additional-paths advertise all
   ```

The following example specifies that all BGP paths are eligible to be selected as additional paths under the IPv4 unicast address family for VRF green, and enables BGP additional-paths send and receive capability for a specified neighbor. The additional-paths feature is enabled and all paths can be advertised to the BGP neighbor.

```
device# configure terminal
device(config)# router bgp
device(config-bgp)# local-as 65520
device(config-bgp)# address-family ipv4 unicast vrf green
device(config-bgp-ipv4u-vrf)# neighbor 10.11.12.13 additional-paths receive
device(config-bgp-ipv4u-vrf)# neighbor 10.11.12.13 additional-paths send
device(config-bgp-ipv4u-vrf)# additional-paths select all
device(config-bgp-ipv4u-vrf)# neighbor 10.11.12.13 additional-paths advertise all
```

# Disabling BGP additional-paths for a specified BGP4 neighbor

You can disable the BGP additional-paths capability for a specific BGP neighbor after the capability has been enabled at the peer-group or address-family level.

1. Enter the **configure terminal** command to access global configuration mode.

   ```
   device# configure terminal
   ```

2. Enter the **router bgp** command to enable BGP routing.

   ```
   device(config)# router bgp
   ```

3. Enter the **local-as** command to configure the autonomous system number (ASN) in which your device resides.

   ```
   device(config-bgp)# local-as 65520
   ```

4. Enter the **neighbor additional-paths disable** command, specifying an IP address, to disable the sending of additional paths by BGP4 to the specified BGP neighbor.

   ```
   device(config-bgp)# neighbor 10.11.12.13 additional-paths disable
   ```

The following example disables the sending of additional paths by BGP4 to the specified neighbor in address-family IPv4 unicast configuration mode.

```
device# configure terminal
device(config)# router bgp
device(config-bgp)# local-as 65520
device(config-bgp)# neighbor 10.11.12.13 additional-paths disable
```

# Configuring BGP additional-paths for a BGP peer group

You can enable the advertisement of additional paths for specified BGP peer groups and apply filters for the advertisement of additional paths for BGP peer groups.

1. Enter the **configure terminal** command to access global configuration mode.

   ```
   device# configure terminal
   ```

2. Enter the **router bgp** command to enable BGP routing.

   ```
   device(config)# router bgp
   ```

3. Enter the **local-as** command to configure the autonomous system number (ASN) in which your device resides.

   ```
   device(config-bgp)# local-as 65520
   ```

4. Enter the **neighbor** *peer-group-name* **peer-group** command to create a peer group.

```
device(config-bgp)# neighbor mypeergroup1 peer-group
```

5. Enter the **neighbor** *peer-group-name* **remote-as** command to specify the ASN of the peer group.

```
device(config-bgp)# neighbor mypeergroup1 remote-as 11
```

6. Enter the **neighbor** *ip-address* **peer-group** command to associate a neighbor with the peer group.

```
device(config-bgp)# neighbor 10.11.12.13 peer-group mypeergroup1
```

7. Enter the **neighbor** *ip-address* **peer-group** command to associate a neighbor with the peer group.

```
device(config-bgp)# neighbor 10.11.13.15 peer-group mypeergroup1
```

8. Enter the **neighbor additional-paths** command, specifying a peer group name and using the **receive** parameter, to enable additional-paths receive capability from a specified BGP peer group.

```
device(config-bgp)# neighbor mypeergroup1 additional-paths receive
```

9. Enter the **neighbor additional-paths** command, specifying a peer group name and using the **send** parameter, to enable additional-paths send capability to a specified BGP peer group.

```
device(config-bgp)# neighbor mypeergroup1 additional-paths send
```

10. Enter the **additional-paths select** command, using the **all** parameter, to specify that all BGP paths are eligible to be selected as additional paths.

```
device(config-bgp)# additional-paths select all
```

11. Enter the **neighbor additional-paths advertise** command, specifying a peer group name and using the **all** parameter, to configure BGP to advertise all BGP additional paths.

```
device(config-bgp)# neighbor mypeergroup1 additional-paths advertise all
```

The following example creates a peer group, enables BGP4 additional-paths send and receive capability for the specified BGP peer group, and specifies that all BGP paths are eligible to be selected as additional paths. The additional-paths feature is enabled and all paths can be advertised.

```
device# configure terminal
device(config)# router bgp
device(config-bgp)# local-as 65520
device(config-bgp)# neighbor mypeergroup1 peer-group
device(config-bgp)# neighbor mypeergroup1 remote-as 11
device(config-bgp)# neighbor 10.11.12.13 peer-group mypeergroup1
device(config-bgp)# neighbor 10.11.13.15 peer-group mypeergroup1
device(config-bgp)# neighbor mypeergroup1 additional-paths receive
device(config-bgp)# neighbor mypeergroup1 additional-paths send
device(config-bgp)# additional-paths select all
device(config-bgp)# neighbor mypeergroup1 additional-paths advertise all
```

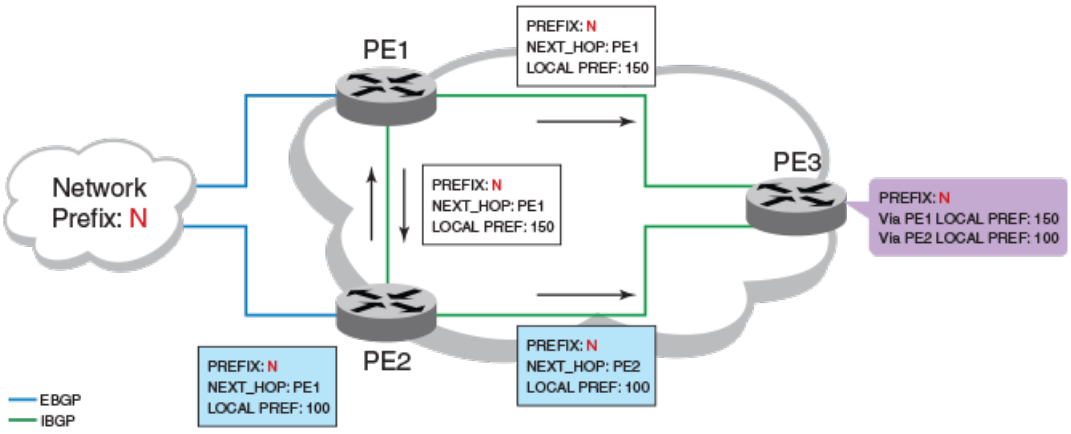
# BGP best external overview

BGP best external enables a device to advertise the most preferred route among those received from external neighbors as a backup route.

In active-backup topologies, service providers use routing policies that cause a border router to choose a path received over an Interior Border Gateway Protocol (iBGP) session as the best path for a prefix. This path is chosen even if an Exterior Border Gateway Protocol (eBGP) learned path exists. BGP best external is beneficial in such a topology. In such a topology, one exit or egress point for the prefix in the autonomous system is defined, and the other points are used as backups if the primary link or eBGP peering is unavailable. The border router does not advertise any path for such prefixes, and the paths learned over its eBGP sessions from the autonomous system (AS) are hidden. To cope with this situation, a device can advertise the best external path.

In the following figure, PE1 is the primary path to network N, and PE2 is the backup path. If BGP best external is not configured, PE2 does not advertise prefix N to its iBGP peers because PE2 prefers the iBGP route from PE1 as the best route compared to its best eBGP route. However, if BGP best external is configured, PE2 propagates its best external path to its iBGP peers so that PE3 has two paths for prefix N.

**FIGURE 33** BGP best external




NOTE
BGP best external is supported for the BGP IPv4 and IPv6 unicast address families and the BGP IPv4 and IPv6 multicast address families.

NOTE
BGP best external is not supported for the BGP L2VPN VPLS, BGP VPNv4 unicast, and BGP VPNv6 address families.

## Limitations of BGP best external

- BGP best external advertises the best external path to iBGP peers only.
- When BGP best external is configured on a route reflector (RR), the best external path is not advertised.

## Upgrade and downgrade considerations

If BGP best external is enabled and the configuration saved, an error message occurs if the software is downgraded to an earlier version. BGP best external should be unconfigured before a downgrade takes place.

## Configuring BGP4 best external

You can enable BGP4 to calculate the best external path and to advertise this path to its neighbors.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **router bgp** command to enable BGP routing.

```
device(config)# router bgp
```

3. Enter the **local-as** command to configure the autonomous system number (ASN) in which your device resides.

```
device(config-bgp)# local-as 65520
```

4. Enter the **advertise-best-external** command to configure BGP4 to calculate the best external path and to advertise this path to its neighbors.

```
device(config-bgp)# advertise-best-external
```

The following example configures BGP4 to calculate the best external path and to advertise this path to its neighbors under the IPv4 unicast address family .

```
device# configure terminal
device(config)# router bgp
device(config-bgp)# local-as 65520
device(config-bgp)# advertise-best-external
```

# Auto shutdown of BGP neighbors on initial configuration

The auto shutdown of BGP neighbors on initial configuration feature prevents a BGP peer from attempting to establish connections with remote peers when the BGP peer is first configured. Sessions are only initiated when the entire configuration for the BGP peer is complete.

When the auto shutdown of BGP neighbors on initial configuration feature is enabled, the value of the shutdown parameter for any existing configured neighbor is not changed. Any new BGP neighbor configured after the feature is enabled has the shutdown state set to the configured value. When the feature is enabled and a new BGP neighbor is configured, the shutdown parameter of the BGP neighbor is automatically set to enabled. When all the BGP neighbor parameters are configured and it is ready to start the establishment of BGP session with the remote peer, the shutdown parameter of the BGP neighbor is then disabled.

Any new neighbor configured and added to a peer group has the shutdown flag enabled by default. Additionally, any new neighbor configured with the autonomous system (AS) in which that remote neighbor resides specified using the **neighbor remote-as** command has the shutdown flag enabled by default.

## Configuring auto shutdown of BGP neighbors on initial configuration

Follow this procedure to enable auto shutdown of BGP neighbors on initial configuration.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **router bgp** command to enable BGP routing.

```
device(config)# router bgp
```

3. Enter the **local-as** command to configure the autonomous system number (ASN) in which your device resides.

```
device(config-bgp)# local-as 65520
```

4. Enter the **auto-shutdown-new-neighbors** command to prevent the BGP peer from attempting to establish connections with remote peers until all BGP neighbor parameters are configured.

```
device(config-bgp)# auto-shutdown-new-neighbors
```

The following example enables auto shutdown of BGP neighbors on initial configuration.

```
device# configure terminal
device(config)# router bgp
device(config-bgp)# local-as 65520
device(config-bgp)# auto-shutdown-new-neighbors
```

## Disabling the BGP4 peer shutdown state

When the auto shutdown of BGP4 neighbors on initial configuration feature is enabled, a BGP4 peer is prevented from attempting to establish connections with remote peers when the BGP4 peer is first configured. Once all of the configuration parameters for the peer are complete, you can start the BGP4 session establishment process and disable the peer shutdown state. Follow this procedure to disable the peer shutdown state.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **router bgp** command to enable BGP routing.

```
device(config)# router bgp
```

3. Enter the **local-as** command to configure the autonomous system number (ASN) in which your device resides.

```
device(config-bgp)# local-as 65520
```

4. Enter the **no neighbor shutdown** command, specifying an IP address, to disable the peer shutdown state and begin the BGP4 session establishment process.

```
device(config-bgp)# no neighbor 10.1.1.1 shutdown
```

The following example disables the peer shutdown state and begins the BGP4 session establishment process.

```
device# configure terminal
device(config)# router bgp
device(config-bgp)# local-as 65520
device(config-bgp)# no neighbor 10.1.1.1 shutdown
```

# Generalized TTL Security Mechanism support

Generalized TTL Security Mechanism (GTSM) is a lightweight security mechanism that protects external Border Gateway Protocol (eBGP) peering sessions from CPU utilization-based attacks using forged IP packets. GTSM prevents attempts to hijack the eBGP peering session by a host on a network segment that is not part of either BGP network, or by a host on a network segment that is not between the eBGP peers.

GTSM is enabled by configuring a minimum Time To Live (TTL) value for incoming IP packets received from a specific eBGP peer. BGP establishes and maintains the session only if the TTL value in the IP packet header is equal to or greater than the TTL value configured for the peering session. If the value is less than the configured value, the packet is silently discarded and no Internet Control Message Protocol (ICMP) message is generated.

When GTSM protection is enabled, BGP control packets sent by the device to a neighbor have a Time To Live (TTL) value of 255. In addition, the device expects the BGP control packets received from the neighbor to have a TTL value of either 254 or 255. For multihop peers, the device expects the TTL for BGP control packets received from the neighbor to be greater than or equal to 255, minus the configured number of hops to the neighbor. If the BGP control packets received from the neighbor do not have the anticipated value, the device drops them.

For more information on GTSM protection, refer to RFC 3682.

## Assumptions and limitations

- GTSM is supported for both directly connected peering sessions and multihop eBGP peering sessions.
- GTSM is supported for eBGP only.
- GTSM does not protect the integrity of data sent between eBGP peers and does not validate eBGP peers through any authentication method.
- GTSM validates only the locally configured TTL count against the TTL field in the IP packet header.
- GTSM should be configured on each participating device to maximize the effectiveness of this feature.
- When GTSM is enabled, the eBGP session is secured in the incoming direction only and has no effect on outgoing IP packets or the remote device.

## Configuring GTSM for BGP4

Generalized TTL Security Mechanism (GTSM) can be configured to protect external Border Gateway Protocol (eBGP) peering sessions.

1. Enter the **configure terminal** command to access global configuration mode.

   ```
   device# configure terminal
   ```

2. Enter the **router bgp** command to enable BGP routing.

   ```
   device(config)# router bgp
   ```

3. Enter the **local-as** command to configure the autonomous system number (ASN) in which your device resides.

   ```
   device(config-bgp)# local-as 65520
   ```

4. Enter the **neighbor remote-as** command to add a neighbor.

   ```
   device(config-bgp)# neighbor 10.10.10.1 remote-as 2
   ```

5. Enter the **neighbor ebgp-btsh** command, specifying an IP address, to enable GTSM.

   ```
   device(config-bgp)# neighbor 10.10.10.1 ebgp-btsh
   ```

The following example enables GTSM between a device and a neighbor with the IP address 10.10.10.1.

```
device# configure terminal
device(config)# router bgp
device(config-bgp)# local-as 65520
device(config-bgp)# neighbor 10.10.10.1 remote-as 2
device(config-bgp)# neighbor 10.10.10.1 ebgp-btsh
```

# Disabling the BGP AS_PATH check function

A device can be configured so that the AS_PATH check function for routes learned from a specific location is disabled, and routes that contain the recipient BGP speaker's AS number are not rejected.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **router bgp** command to enable BGP routing.

```
device(config)# router bgp
```

3. Enter the **neighbor allowas-in** command and specify a **number** to disable the BGP AS_PATH check function, and specify the number of times that the AS path of a received route may contain the recipient BGP speaker's AS number and still be accepted.

```
device(config-bgp)# neighbor 10.1.1.1 allowas-in 3
```

The following example specifies that the AS path of a received route may contain the recipient BGP speaker's AS number three times and still be accepted.

```
device# configure terminal
device(config)# router bgp
device(config-bgp)# neighbor 10.1.1.1 allowas-in 3
```

# Matching on an AS-path

A route map that matches on a specified AS-path access control list (ACL) can be configured.

An AS-path ACL must be configured using the **ip as-path access-list** command.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **route-map** command using the **permit** parameter and specifying a route map name to create a route map instance and allow a matching pattern.

```
device(config)# route-map myaclroutemap1 permit 10
```

3. Enter the **match** command and specify an ACL name to configure the route map to match on the specified AS-path ACL.

```
device(config-routemap myaclroutemap1)# match as-path myaspath
```

The following example configures a route map instance that matches on AS-path ACL "myaspath".

```
device# configure terminal
device(config)# route-map myaclroutemap1 permit 10
device(config-routemap myaclroutemap1)# match as-path myaspath
```

# Matching on a community ACL

A route map instance that matches on a specified community access control list (ACL) can be configured.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **ip community-list standard** command, specifying a community number and using the **permit** parameter to create a community ACL that permits traffic.

```
device(config)# ip community-list standard 1 permit 123:2
```

3. Enter the **route-map** command using the **permit** parameter and specifying a route map name to create a route map instance and allow a matching pattern.

```
device(config)# route-map mycommroutemap1 permit 10
```

4. Enter the **match community** command, and specify a community number to match the BGP community access list name in the configured route-map instance.

```
device(config-route-map-mycommroutemap1)# match community 1
```

The following example matches community ACL "1" for route map instance "mycommroutemap1".

```
device# configure terminal
device(config)# ip community-list standard 1 permit 123:2
device(config)# route-map mycommroutemap1 permit 10
device(config-route-map-mycommroutemap1)# match community 1
```

# Matching on a destination network

A route map that matches on a destination network can be configured.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **route-map** command using the **permit** parameter and specifying a route map name to create a route map instance and allow a matching pattern.

```
device(config)# route-map mynetroutemap1 permit 10
```

3. Enter the **match** command with the **ip address** parameter. Specify a prefix list using the **ip prefix-list** *string* parameter to configure the route map to match on the specified prefix.

```
device(config-route-map-mynetroutemap1)# match ip address prefix-list mylist
```

The following example configures a route map instance that matches on a specified destination network.

```
device# configure terminal
device(config)# route-map mynetroutemap1 permit 10
device(config-route-map-mynetroutemap1)# match ip address prefix-list mylist
```

# Matching on a BGP4 static network

A route map that matches on a static network can be configured. In this task, a route-map is configured to match on the BGP4 static network. The device is then configured to filter the outgoing route updates to a specified BGP neighbor according to the set of attributes defined in the configured route map.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **route-map** command using the **permit** parameter and specifying a route map name to create a route map instance and allow a matching pattern.

```
device(config)# route-map mystaticroutemap3 permit 1
```

3. Enter the **match** command with the **protocol bgp static-network** parameter to match on BGP4 static network routes.

```
device(config-routemap-mystaticroutemap3)# match protocol bgp static-network
```

4. Enter the **set local-preference** command and enter a value to set a BGP local-preference path attribute in the route-map instance.

```
device(config-routemap-mystaticroutemap3/)# set local-preference 150
```

5. Enter the **set community** command with the **no export** parameter to set the BGP community attribute for the route-map instance not to export to the next AS.

```
device(config-routemap-mystaticroutemap3)# set community no-export
```

6. Enter the **exit** command to exit route map configuration mode.

```
device(config-routemap-mystaticroutemap3)# exit
```

7. Enter the **router bgp** command to enable BGP routing.

```
device(config)# router bgp
```

8. Enter the **neighbor** *ip-address* **route-map** command and specify the **out** keyword, specifying the route map name, to filter the outgoing route updates to a specified BGP neighbor according to the set of attributes defined in the configured route map.

```
device (config-bgp)# neighbor 10.1.2.3 route-map out mystaticroutemap3
```

The following example configures a route map instance that matches on a BGP4 static network and configures the device to filter the outgoing route updates to a specified BGP neighbor according to the set of attributes defined in the configured route map.

```
device# configure terminal
device(config)# route-map mystaticroutemap3 permit 1
device(config-routemap-mystaticroutemap3)# match protocol bgp static-network
device(config-routemap-mystaticroutemap3)# set local-preference 150
device(config-routemap-mystaticroutemap3)# set community no-export
device(config-routemap-mystaticroutemap3)# exit
device(config)# router bgp
device(config-bgp)# neighbor 10.1.2.3 route-map out mystaticroutemap3
```

# Matching on a next-hop device

A route map that matches on a next-hop device can be configured.

A prefix list must be configured using the **ip prefix-list** command.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **route-map** command using the **permit** parameter and specifying a route map name to create a route map instance and allow a matching pattern.

```
device(config)# route-map myhoproutemap1 permit 10
```

3. Enter the **match** command, using the **next-hop** parameter and specify a prefix-list, to match IP next-hop match conditions for a specified prefix list in a route-map instance.

```
device(config-routemap myaclroutemap1)# match ip next-hop    prefix-list mylist
```

The following example configures a route map and specifies a prefix list to match on a next-hop device.

```
device# configure terminal
device(config)# route-map myhoproutemap1 permit 10
device(config-route-map-myhoproutemap1)# match ip next-hop    prefix-list mylist
```

# Using route-map continue statements

Continuation statements can be configured in a route map. This task configures a route map instance and adds two route-map continue statements to the route map instance.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **route-map** command using the **permit** parameter and specifying a route map name to create a route map instance and allow a matching pattern.

```
device(config)# route-map mcontroutemap1 permit 1
```

3. Enter the **match** command with the metric parameter and specify a value to match a route metric in the route-map instance.

```
device(config-routemap mcontroutemap1)# match metric 10
```

4. Enter the **set weight** command and specify a value to set a BGP weight for the routing table in the route-map instance.

```
device(config-routemap mcontroutemap1)# set weight 10
```

5. Enter the **continue** command and specify a value to configure a route-map instance number that goes in a continue statement in the route-map instance.

```
device(config-routemap mcontroutemap1)# continue 2
```

6. Enter the **exit** command to exit route map configuration mode.

```
device(config-routemap mcontroutemap1)# exit
```

7. Enter the **route-map** command using the **permit** parameter and specifying a route map name to create a route map instance and allow a matching pattern.

```
device(config)# route-map mcontroutemap1 permit 2
```

8. Enter the **match** command with the metric parameter and specify a value to match a route metric in the route-map instance.

```
device(config-routemap mcontroutemap1)# match metric 10
```

9. Enter the **set weight** command and specify a value to set a BGP weight for the routing table in the route-map instance.

```
device(config-routemap mcontroutemap1)# set weight 20
```

10. Enter the **continue** command and specify a value to configure a route-map instance number that goes in a continue statement in the route-map instance.

```
device(config-routemap mcontroutemap2)# continue 3
```

The following example configures a route map instance and adds two route-map continue statements to the route map instance.

```
device# configure terminal
device(config)# route-map mcontroutemap1 permit 1
device(config-routemap-mcontroutemap1)# match metric 10
device(config-routemap-mcontroutemap1)# set weight 10
device(config-routemap-mcontroutemap1)# match metric 10
device(config-routemap-mcontroutemap1)# continue 2
device(config-routemap-mcontroutemap1)# exit
device(config)# route-map mcontroutemap1 permit 2
device(config-routemap-mycontroutemap1)# match tag 10
device(config-routemap-mycontroutemap1)# set weight 20
```

# Route-map continue statement for BGP4 routes

A continue statement in a route-map directs program flow to skip over route-map instances to another, user-specified instance. If a matched instance contains a continue statement, the system looks for the instance that is identified in the statement.

The continue statement in a matching instance initiates another traversal at the instance specified. The system records all of the matched instances and, if no deny statements are encountered, proceeds to execute the set clauses of the matched instances.

If the system scans all route-map instances but finds no matches, or if a deny condition is encountered, then it does not update the routes. Whenever a matched instance contains a deny statement, the current traversal terminates, and none of the updates specified in the set statements of the matched instances in both current and previous traversals are applied to the routes.

This supports a more programmable route-map configuration and route filtering scheme for BGP4 peering. It can also execute additional instances in a route map after an instance is executed by means of successful match statements. You can configure and organize more-modular policy definitions to reduce the number of instances that are repeated within the same route map.

This feature currently applies to BGP4 routes only. For protocols other than BGP4, continue statements are ignored.

# Using a route map to configure dampening

You can set a BGP route-flap dampening penalty in a route-map instance..

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **route-map** command using the **permit** parameter and specifying a route map name to create a route map instance and allow a matching pattern.

```
device(config)# route-map myroutemap permit 1
```

3. Enter the **set dampening** command and specify a value name to set the BGP route-flap dampening penalty for the route-map instance.

```
device(config-route-map-myroutemap)# set dampening 20
```

The following example configures a route map instance and sets a BGP route-flap dampening penalty of 20.

```
device# configure terminal
device(config)# route-map myroutemap permit 1
device(config-route-map-myroutemap)# set dampening 20
```

# Clearing diagnostic buffers

The device stores the following BGP4 diagnostic information in buffers:

- The first 400 bytes of the last packet received that contained an error
- The last NOTIFICATION message either sent or received by the device

This information can be useful if you are working with Extreme Technical Support to resolve a problem. The buffers do not identify the system time when the data was written to the buffer. If you want to ensure that diagnostic data in a buffer is recent, you can clear the buffers. You can clear the buffers for a specific neighbor or for all neighbors.

If you clear the buffer containing the first 400 bytes of the last packet that contained errors, all the bytes are changed to zeros. The Last Connection Reset Reason field of the BGP4 neighbor table also is cleared.

If you clear the buffer containing the last NOTIFICATION message sent or received, the buffer contains no data.

You can clear the buffers for all neighbors, for an individual neighbor, or for all the neighbors within a specific peer group.

Refer to the NetIron Command Reference for more information.

# Displaying BGP4 statistics

Various **show ip bgp** commands verify information about BGP4 configurations.

Use one or more of the following commands to verify BGP4 information. The commands do not have to be entered in this order.

1. Enter the **show ip bgp summary** command.

```
device> show ip bgp summary

  BGP4 Summary
  Router ID: 7.7.7.7   Local AS Number: 100
  Confederation Identifier: not configured
  Confederation Peers:
  Maximum Number of IP ECMP Paths Supported for Load Sharing: 1
  Number of Neighbors Configured: 1, UP: 1
  Number of Routes Installed: 0
  Number of Routes Advertising to All Neighbors: 0 (0 entries)
  Number of Attribute Entries Installed: 0
  '+': Data in InQueue '>': Data in OutQueue '-': Clearing
  '*': Update Policy 'c': Group change 'p': Group change Pending
  'r': Restarting 's': Stale '^': Up before Restart '<': EOR waiting
  Neighbor Address  AS#         State   Time          Rt:Accepted Filtered Sent     ToSend
  10.1.1.8          100         ESTAB   0h 9m16s      0           0        0        0
```

   This example output gives summarized BGP4 information.

2. Enter the **show ip bgp routes** command.

```
device> show ip bgp routes

Total number of BGP Routes: 2000
Status A:AGGREGATE B:BEST b:NOT-INSTALLED-BEST C:CONFED_EBGP D:DAMPED
       E:EBGP H:HISTORY I:IBGP L:LOCAL M:MULTIPATH m:NOT-INSTALLED-MULTIPATH
       S:SUPPRESSED F:FILTERED s:STALE x:BEST-EXTERNAL
       Prefix           Next Hop        MED        LocPrf     Weight Status
1      150.150.150.0/24   103.103.1.1     2          100        0      BEx
         AS_PATH: 201
2      150.150.150.0/24   103.103.2.1     3          100        0      E
         AS_PATH: 202
3      150.150.150.0/24   103.103.3.1     4          100        0      E
         AS_PATH: 203
4      150.150.150.0/24   103.103.4.1     5          100        0      E
         AS_PATH: 204
5      150.150.150.0/24   103.103.5.1     6          100        0      E
…
```

   This example shows general BGP4 route information.

3. Enter the **show ip bgp** command.

```
device> show ip bgp

Total number of BGP Routes: 4
Status codes: s suppressed, d damped, h history, * valid, > best, i internal, S stale, x best-
external
Origin codes: i - IGP, e - EGP, ? - incomplete
    Network          Next Hop        MED        LocPrf     Weight Path
*>i 110.110.110.0/24   50.50.50.10               150        0      i
*x  110.110.110.0/24   20.20.20.10               100        0      200 i
*   110.110.110.0/24   30.30.30.10               100        0      300 i
*   110.110.110.0/24   40.40.40.10               100        0      400 i
```

   This example shows general BGP4 information.

4. Enter the **show ip bgp attribute-entries** command.

```
device> show ip bgp attribute-entries

        Total number of BGP Attribute Entries: 18 (0)
1       Next Hop  :192.168.1.6        MED :1             Origin:INCOMP
        Originator:0.0.0.0            Cluster List:None
        Aggregator:AS Number :0      Router-ID:0.0.0.0     Atomic:None
        Local Pref:100               Communities:Internet
        Extended Community: SOO 300000:3
        AS Path   :90000 80000 (length 11)
        Address: 0x10e4e0c4  Hash:489 (0x03028536), PeerIdx 0
        Links: 0x00000000, 0x00000000, nlri: 0x10f4804a
        Reference Counts: 1:0:1, Magic: 51
2       Next Hop  :192.168.1.5        Metric   :1         Origin:INCOMP
        Originator:0.0.0.0            Cluster List:None
        Aggregator:AS Number :0      Router-ID:0.0.0.0     Atomic:None
        Local Pref:100               Communities:Internet
        Extended Community: RT 200000:2
        AS Path   :90000 75000 (length 11)
        Address: 0x10e4e062  Hash:545 (0x0301e8f6), PeerIdx 0
        Links: 0x00000000, 0x00000000, nlri: 0x10f47ff0
        Reference Counts: 1:0:1, Magic: 49
```

This example shows information about one route-attribute entry that is stored in device memory.

5. Enter the **show ip bgp peer-group** command.

```
device# show ip bgp peer-group STR

1   BGP peer-group is STR
      Address family : IPV4 Unicast
        activate
      Address family : IPV4 Multicast
        no activate
      Address family : IPV6 Unicast
        no activate
      Address family : IPV6 Multicast
        no activate
      Address family : VPNV4 Unicast
        no activate
      Address family : L2VPN VPLS
        no activate
    Members:
       IP Address: 10.1.1.1, AS: 5
```

This example shows output for one BGP peer group, called “STR”.

6. Enter the **show ip bgp routes** command using the **summary** keyword.

```
device> show ip bgp routes summary

  Total number of BGP routes (NLRIs) Installed     : 20
  Distinct BGP destination networks                : 20
  Filtered BGP routes for soft reconfig            : 100178
  Routes originated by this router                 : 2
  Routes selected as BEST routes                   : 19
  BEST routes not installed in IP forwarding table : 1
  Unreachable routes (no IGP route for NEXTHOP)    : 1
  IBGP routes selected as best routes              : 0
  EBGP routes selected as best routes              : 17
```

This example shows summarized BGP4 route information.

# Displaying BGP4 neighbor statistics

Various **show ip bgp neighbor** commands verify information about BGP4 neighbor configurations.

Use one or more of the following commands to verify BGP4 neighbor information. The commands do not have to be entered in this order.

1. Enter the **show ip bgp neighbors** command.

```
device> show ip bgp neighbors

    '+': Data in InQueue '>': Data in OutQueue '-': Clearing
    '*': Update Policy 'c': Group change 'p': Group change Pending
    'r': Restarting 's': Stale '^': Up before Restart '<': EOR waiting

1   IP Address: 60.60.60.20, AS: 200 (IBGP), RouterID: 60.60.60.20, VRF: default-vrf
    State: ESTABLISHED, Time: 4h3m28s, KeepAliveTime: 60, HoldTime: 180
       KeepAliveTimer Expire in 0 seconds, HoldTimer Expire in 159 seconds
    Minimal Route Advertisement Interval: 0 seconds
       RefreshCapability: Received
    Address Family : IPV4 Unicast
       Configured with Add-Path(send receive)capability
       Received Add-Path (send receive)capability in open msg
       Negotiated Add-Path(send receive)capability
    Messages:    Open        Update      KeepAlive   Notification   Refresh-Req
       Sent    : 1           1           275         0              0
       Received: 1           1           275         0              0
    Last Update Time: NLRI       Withdraw          NLRI       Withdraw
                  Tx: 4h3m28s    ---           Rx: 4h3m28s    ---
```

   This example output gives general information about BGP4 neighbors.

2. Enter the **show ip bgp neighbors advertised-routes** command.

```
device> show ip bgp neighbors 192.168.4.211 advertised-routes

       There are 2 routes advertised to neighbor 192.168.4.211
Status A:AGGREGATE B:BEST b:NOT-INSTALLED-BEST I:IBGP L:LOCAL
       Network        Next Hop        Metric  LocPrf   Weight   Status
1      10.102.0.0/24  192.168.2.102   12               32768    BL
2      10.200.1.0/24  192.168.2.102   0                32768    BL
```

   This example shows information about all the routes the BGP4 networking device advertised to the neighbor.

3. Enter the **show ip bgp neighbors** command and specify an IP address.

```
device> show ip bgp neighbors 10.4.0.2

    Total number of BGP neighbors:
1   IP Address: 10.4.0.2, AS: 5 (EBGP), RouterID: 10.0.0.1
       Description: neighbor 10.4.0.2
    Local AS: 101
    State: ESTABLISHED, Time: 0h1m0s, KeepAliveTime: 0, HoldTime: 0
       PeerGroup: pg1
       Multihop-EBGP: yes, ttl: 1
       RouteReflectorClient: yes
       SendCommunity: yes
       NextHopSelf: yes
       DefaultOriginate: yes (default sent)
       MaximumPrefixLimit: 90000
       RemovePrivateAs: : yes
       RefreshCapability: Received
    Route Filter Policies:
       Distribute-list: (out) 20
       Filter-list: (in) 30
       Prefix-list: (in) pf1
       Route-map: (in) setnp1  (out) setnp2
    Messages:    Open    Update  KeepAlive Notification Refresh-Req
       Sent    : 1       1       1         0            0
       Received: 1       8       1         0            0
    Last Update Time: NLRI       Withdraw          NLRI       Withdraw
                  Tx: 0h0m59s    ---           Rx: 0h0m59s    ---
    Last Connection Reset Reason:Unknown
      Notification Sent:     Unspecified
      Notification Received: Unspecified
    TCP Connection state: ESTABLISHED
       Local host:  10.4.0.1, Local  Port: 179
       Remote host: 10.4.0.2, Remote Port: 8053
       ISentSeq:   52837276  SendNext:   52837392  TotUnAck:          0
       TotSent:         116  ReTrans:           0  UnAckSeq:   52837392
       IRcvSeq:  2155052043  RcvNext:  2155052536  SendWnd:       16384
       TotalRcv:        493  DupliRcv:          0  RcvWnd:        16384
       SendQue:           0  RcvQue:            0  CngstWnd:       1460
```

This example shows information about a specifies BGP4 neighbor.

4. Enter the **show ip bgp neighbors received-routes** command.

```
device> show ip bgp neighbor 10.168.4.106 received-routes

       There are 97345 received routes from  neighbor 10.168.4.106
Searching for matching routes, use ^C to quit...
tatus A:AGGREGATE B:BEST b:NOT-INSTALLED-BEST C:CONFED_EBGP D:DAMPED
       E:EBGP H:HISTORY I:IBGP L:LOCAL M:MULTIPATH S:SUPPRESSED F:FILTEREDtatus A:AGGREGATE B:BEST
b:NOT-INSTALLED-BEST C:CONFED_EBGP D:DAMPED
       E:EBGP H:HISTORY I:IBGP L:LOCAL M:MULTIPATH S:SUPPRESSED F:FILTERED
       Prefix             Next Hop        MED     LocPrf    Weight Status
1      10.3.0.0/8           10.168.4.106           100       0      BE
         AS_PATH: 65001 4355 701 8
2      10.4.0.0/8           10.168.4.106           100       0      BE
         AS_PATH: 65001 4355 1
3      10.60.212.0/22       10.168.4.106           100       0      BE
         AS_PATH: 65001 4355 701 1 189
4      10.6.0.0/8           10.168.4.106           100       0      BE
```

This example lists all route information received in route updates from BGP4 neighbors of the device since the soft-reconfiguration feature was enabled.

5. Enter the **show ip bgp neighbors rib-out-routes** command.

```
device> show ip bgp neighbor 192.168.4.211 rib-out-routes 192.168.1.0/24

Status A:AGGREGATE B:BEST b:NOT-INSTALLED-BEST I:IBGP L:LOCAL
       Prefix             Next Hop        Metric     LocPrf     Weight Status
1      10.200.1.0/24       0.0.0.0         0          101        32768  BL
```

This example shows information about BGP4 outbound RIB routes.

# BGP4+



# BGP4+ overview

The implementation of IPv6 supports multiprotocol BGP (MBGP) extensions that allow Border Gateway Protocol version 4 plus (BGP4+) to distribute routing information. BGP4+ supports all of the same features and functionality as IPv4 BGP (BGP4).

IPv6 MBGP enhancements include:

- An IPv6 unicast address family and network layer reachability information (NLRI)
- An IPv6 multicast address family
- Next hop attributes that use IPv6 addresses

The implementation of BGP4+ supports the advertising of routes among different address families. BGP4+ multicast and unicast routes are supported. IPv6 unicast-routing is enabled by default.

# BGP global mode

Configurations that are not specific to address family configuration are available in the BGP global configuration mode.

```
device(config-bgp)# ?

Possible completions:

  address-family                Enter Address Family command mode
  aggregate-address             Configure BGP aggregate entries
  always-compare-med            Allow comparing MED from different neighbors
  always-propagate              Allow readvertisement of best BGP routes not in
                                IP forwarding table
  as-path-ignore                Ignore AS_PATH length info for best route
                                selection
  auto-shutdown-new-neighbors   Automatically set new BGP neighbors to shutdown
                                state
  bfd                           Set BFD global parameters for BGP4
  bfd-enable                    Enable BFD for BGP4
  bgp-redistribute-internal     Allow redistribution of iBGP routes into IGPs
  capability                    Set capability
  clear                         Clear table/statistics/keys
  client-to-client-reflection   Configure client to client route reflection
  cls                           Clear screen
  cluster-id                    Configure Route-Reflector Cluster-ID
  compare-med-empty-aspath      Allow comparing MED from different neighbors
                                even with empty as-path attribute
  compare-routerid              Compare router-id for identical BGP paths
  confederation                 Configure AS confederation parameters
  dampening                     Enable route-flap dampening
  default-information-origina   Distribute a default route
  default-local-preference      Configure default local preference value
  default-metric                Set metric of redistributed routes
  distance                      Define an administrative distance
  enforce-first-as              Enforce the first AS for EBGP routes
  fast-external-fallover        Reset session if link to EBGP peer goes down
  graceful-restart              Enables the BGP graceful restart capability
  install-igp-cost              Install igp cost to nexthop instead of MED
                                value as BGP route cost
  local-as                      Configure local AS number
  log-dampening-debug           Log dampening debug messages
  maxas-limit                   Impose limit on number of ASes in AS-PATH
                                attribute
  maximum-paths                 Forward packets over multiple paths
  med-missing-as-worst          Consider routes missing MED attribute as least
                                desirable
  multipath                     Enable multipath for ibgp or ebgp neighbors only
  neighbor                      Specify a neighbor router
  network                       Specify a network to announce via BGP
  next-hop-enable-default       Enable default route for BGP next-hop lookup
  next-hop-mpls                 Include MPLS routes for BGP next-hop lookup
  next-hop-recursion            Perform next-hop recursive lookup for BGP route
  redistribute                  Redistribute information from another routing
                                protocol
  rib-route-limit               Limit BGP rib count in routing table
  static-network                Special network that do not depends on IGP and
                                always treat as best route in BGP
  table-map                     Map external entry attributes into routing table
  timers                        Adjust routing timers
  update-time                   Configure igp route update interval
```

The following neighbor configuration options are allowed under BGP global configuration mode:

```
device(config-bgp)# neighbor 2001:2018:8192::125 ?

Possible completions:
   activate                 Allow exchange of route in the current family mode
  advertisement-interval   Minimum interval between sending BGP routing updates
  bfd                      Set BFD parameters for BGP4 neighbor
```

```
  capability               Advertise capability to the peer
  description              Neighbor by description
  ebgp-btsh                Enable EBGP TTL Security Hack Protection
  ebgp-multihop            Allow EBGP neighbors not on directly connected
                           networks
  enforce-first-as         Enforce the first AS for EBGP routes
  fail-over                Set Failover source for BGP4 for this peer
  local-as                 Assign local-as number to neighbor
  maxas-limit              Impose limit on number of ASes in AS-PATH attribute
  next-hop-self            Disable the next hop calculation for this neighbor
  password                 Enable TCP-MD5 password protection
  peer-group               Assign peer-group to neighbor
  remote-as                Specify a BGP neighbor
  remove-private-as        Remove private AS number from outbound updates
  shutdown                 Administratively shut down this neighbor
  soft-reconfiguration     Per neighbor soft reconfiguration
  static-network-edge      Neighbor as special service edge, static-network
                           shall not be advertised if installed as DROP
  timers                   BGP per neighbor timers
  update-source            Source of routing updates
```

# IPv6 unicast address family

The IPv6 unicast address family configuration level provides access to commands that allow you to configure BGP4+ unicast routes. The commands that you enter at this level apply only to the IPv6 unicast address family.

BGP4+ supports the IPv6 address family configuration level.

You can generate a configuration for BGP4+ unicast routes that is separate and distinct from configurations for IPv4 unicast routes and BGP multicast routes.

The commands that you can access while at the IPv6 unicast address family configuration level are also available at the IPv4 unicast address family configuration levels. Each address family configuration level allows you to access commands that apply to that particular address family only.

Where relevant, this chapter discusses and provides IPv6-unicast-specific examples. You must first configure IPv6 unicast routing for any IPv6 routing protocol to be active.

The following configuration options are allowed under BGP IPv6 address family unicast mode:

```
device(config-bgp-ipv6u)# ?

Possible completions:

  aggregate-address             Configure BGP aggregate entries
  always-propagate              Allow readvertisement of best BGP routes not in
                                IP forwarding table
  bfd                           Set BFD global parameters for BGP4
  bfd-enable                    Enable BFD for BGP4
  bgp-redistribute-internal     Allow redistribution of iBGP routes into IGPs
  client-to-client-reflection   Configure client to client route reflection
  dampening                     Enable route-flap dampening
  default-information-origina   Distribute a default route
  default-metric                Set metric of redistributed routes
  graceful-restart              Enables the BGP graceful restart capability
  log-dampening-debug           Log dampening debug messages
  maximum-paths                 Forward packets over multiple paths
  multipath                     Enable multipath for ibgp or ebgp neighbors only
  neighbor                      Specify a neighbor router
  network                       Specify a network to announce via BGP
  next-hop-enable-default       Enable default route for BGP next-hop lookup
  next-hop-recursion            Perform next-hop recursive lookup for BGP route
  redistribute                  Redistribute information from another routing
                                protocol
  rib-route-limit               Limit BGP rib count in routing table
```

```
  table-map                     Map external entry attributes into routing table
  update-time                   Configure igp route update interval
```

The following neighbor configuration options are allowed under BGP IPv6 address family unicast mode:

```
device(config-bgp-ipv6u)# neighbor 2001:2018:8192::125 ?

Possible completions:
  activate                  Allow exchange of route in the current family mode
  additional-paths          Specify bgp additional paths
  advertisement-interval    Minimum interval between sending BGP routing updates
  allowas-in                Accept as-path with my AS present in it
  as-override               Override matching AS-number while sending update
  bfd                       Set BFD parameters for BGP4 neighbor
  capability                Advertise capability to the peer
  default-originate         Originate default route to peer
  description               Neighbor by description
  ebgp-btsh                 Enable EBGP TTL Security Hack Protection
  ebgp-multihop             Allow EBGP neighbors not on directly connected
                            networks
  fail-over                 Set Failover source for BGP4 for this peer
  filter-list               Establish BGP filters
  local-as                  Assign local-as number to neighbor
  maximum-prefix            Maximum number of prefix accept from this peer
  next-hop-self             Disable the next hop calculation for this neighbor
  password                  Enable TCP-MD5 password protection
  peer-group                Assign peer-group to neighbor
  prefix-list               Prefix List for filtering routes
  remote-as                 Specify a BGP neighbor
  remove-private-as         Remove private AS number from outbound updates
  route-map                 Apply route map to neighbor
  route-reflector-client    Configure a neighbor as Route Reflector client
  send-community            Send community attribute to this neighbor
  shutdown                  Administratively shut down this neighbor
  soft-reconfiguration      Per neighbor soft reconfiguration
  timers                    BGP per neighbor timers
  unsuppress-map            Route-map to selectively unsuppress suppressed routes
  update-source             Source of routing updates
  weight                    Set default weight for routes from this neighbor
```

# IPv6 multicast address family

The BGP4+ multicast address family configuration level provides access to commands that allow you to configure BGP4+ multicast routes. The commands that you enter at this level apply only to the IPv6 multicast address family.

BGP4+ supports the IPv6 address family configuration level.

You can generate a configuration for BGP4+ multicast routes that is separate and distinct from configurations for BGP unicast routes and IPv4 BGP multicast routes.

The commands that you can access while at the IPv6 multicast address family configuration level are also available at the IPv4 multicast address family configuration levels. Each address family configuration level allows you to access commands that apply to that particular address family only.

The following configurations are allowed under BGP IPv6 address-family multicast mode:

```
device(config-bgp-ipv6m)# ?

Possible completions:

  aggregate-address             Configure BGP aggregate entries
  always-propagate              Allow readvertisement of best BGP routes not in
                                IP forwarding table
  bfd                           Set BFD global parameters for BGP4
  bfd-enable                    Enable BFD for BGP4
  client-to-client-reflection   Configure client to client route reflection
```

```
dampening                       Enable route-flap dampening
default-information-origina     Distribute a default route
default-metric                  Set metric of redistributed routes
graceful-restart                Enables the BGP graceful restart capability
log-dampening-debug             Log dampening debug messages
maximum-paths                   Forward packets over multiple paths
multipath                       Enable multipath for ibgp or ebgp neighbors only
neighbor                        Specify a neighbor router
network                         Specify a network to announce via BGP
next-hop-enable-default         Enable default route for BGP next-hop lookup
redistribute                    Redistribute information from another routing
                                protocol
rib-route-limit                 Limit BGP rib count in routing table
table-map                       Map external entry attributes into routing table
update-time                     Configure igp route update interval
```

# BGP4+ neighbors

BGP4+ neighbors can be configured using link-local addresses or global addresses.

BGP4+ neighbors can be created using link-local addresses for peers in the same link. For link-local peers, the neighbor interface over which the neighbor and local device exchange prefixes is specified through the **neighbor update-source** command, and a route map is configured to set up a global next hop for packets destined for the neighbor.

To configure BGP4+ neighbors that use link-local addresses, you must do the following:

- Add the IPv6 address of a neighbor in a remote autonomous system (AS) to the BGP4+ neighbor table of the local device.
- Identify the neighbor interface over which the neighbor and local device will exchange prefixes using the **neighbor update-source** command.
- Configure a route map to set up a global next hop for packets destined for the neighbor.

The neighbor should be activated in the IPv6 address family configuration mode using the **neighbor activate** command.

BGP4+ neighbors can also be configured using a global address. The global IPv6 address of a neighbor in a remote AS must be added, and the neighbor should be activated in the IPv6 address family configuration mode using the **neighbor activate** command.

## Configuring BGP4+ neighbors using global IPv6 addresses

BGP4+ neighbors can be configured using global IPv6 addresses.

1. Enter the **configure terminal** command to access global configuration mode.

   ```
   device# configure terminal
   ```

2. Enter the **router bgp** command to enable BGP routing.

   ```
   device(config)# router bgp
   ```

3. Enter the **local-as** command to configure the autonomous system number (ASN) in which your device resides.

   ```
   device(config-bgp)# local-as 1000
   ```

4. Enter the **neighbor remote-as** command, specifying an IPv6 address, to specify the ASN in which the remote neighbor resides.

   ```
   device(config-bgp)# neighbor 2001:db8:93e8:cc00::1 remote-as 1001
   ```

5. Enter the **address family ipv6 unicast** command to enter IPv6 address family configuration mode.

```
device(config-bgp)# address-family ipv6 unicast
```

6. Enter the **neighbor activate** command to enable the exchange of information with the neighbor.

```
device(config-bgp-ipv6u)# neighbor 2001:db8:93e8:cc00::1 activate
```

The following example configures a neighbor using a global IPv6 address.

```
device# configure terminal
device(config)# router bgp
device(config-bgp)# local-as 1000
device(config-bgp)# neighbor 2001:db8:93e8:cc00::1 remote-as 1001
device(config-bgp)# address-family ipv6 unicast
device(config-bgp-ipv6u)# neighbor 2001:db8:93e8:cc00::1 activate
```

# Configuring BGP4+ neighbors using global IPv6 addresses in the IPv6 multicast address family

BGP4+ neighbors can be configured in the address-family IPv6 multicast configuration mode using global IPv6 addresses.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **router bgp** command to enable BGP routing.

```
device(config)# router bgp
```

3. Enter the **local-as** command to configure the autonomous system number (ASN) in which your device resides.

```
device(config-bgp)# local-as 1001
```

4. Enter the **neighbor remote-as** command, specifying an IPv6 address, to specify the autonomous system ASN in which the remote neighbor resides.

```
device(config-bgp)# neighbor 2001:db8:93e8:cc00::2 remote-as 1002
```

5. Enter the **address-family ipv6 muliticast** command to enter IPv6 address family multicast configuration mode.

```
device(config-bgp)# address-family ipv6 multicast
```

6. Enter the **neighbor activate** command to enable the exchange of information with the neighbor.

```
device(config-bgp-ipv6m)# neighbor 2001:db8:93e8:cc00::2 activate
```

The following example configures a neighbor using a global IPv6 address in address family IPv6 multicast configuration mode.

```
device# configure terminal
device(config)# router bgp
device(config-bgp)# local-as 1001
device(config-bgp)# neighbor 2001:db8:93e8:cc00::2 remote-as 1002
device(config-bgp)# addresms-family ipv6 multicast
device(config-bgp-ipv6u)# neighbor 2001:db8:93e8:cc00::2 activate
```

# Configuring BGP4+ neighbors using link-local addresses

BGP4+ neighbors can be configured using link-local addresses.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **router bgp** command to enable BGP routing.

```
device(config)# router bgp
```

3. Enter the **local-as** command to configure the autonomous system number (ASN) in which your device resides.

```
device(config-bgp)# local-as 1000
```

4. Enter the **neighbor remote-as** command, specifying an IPv6 address, to specify the ASN in which the remote neighbor resides.

```
device(config-bgp)# neighbor fe80:4398:ab30:45de::1 remote-as 1001
```

5. Enter the **neighbor update-source** command to specify an interface.

```
device(config-bgp)# neighbor fe80:4398:ab30:45de::1 update-source ethernet 3/1
```

6. Enter the **address-family ipv6 unicast** command to enter IPv6 address family configuration mode.

```
device(config-bgp)# address-family ipv6 unicast
```

7. Enter the **neighbor activate** command to enable the exchange of information with the neighbor.

```
device(config-bgp-ipv6u)# neighbor fe80:4398:ab30:45de::1 activate
```

8. Enter the **neighbor route-map** command and specify the **out** keyword to apply a route map to outgoing routes.

```
device(config-bgp-ipv6u)# neighbor fe80:4398:ab30:45de::1 route-map out myroutemap
```

9. Enter the **exit** command until you return to global configuration mode.

```
device(config-bgp-ipv6u)# exit
```

10. Enter the **route-map** *name* **permit** command to define the route map and enter route map configuration mode.

```
device(config)# route-map myroutemap permit 10
```

11. Enter the **set ipv6 next-hop** command and specify an IPv6 address to set the IPv6 address of the next hop.

```
device(config-routemap-myroutemap)# set ipv6 next-hop 2001::10
```

The following example configures a neighbor using a link-local address and configures a route map to set up a global next hop for packets destined for the neighbor.

```
device# configure terminal
device(config)# router bgp
device(config-bgp)# local-as 1000
device(config-bgp)# neighbor fe80:4398:ab30:45de::1 remote-as 1001
device(config-bgp)# neighbor fe80:4398:ab30:45de::1 update-source ethernet 3/1
device(config-bgp)# address-family ipv6 unicast
device(config-bgp-ipv6u)# neighbor fe80:4398:ab30:45de::1 activate
device(config-bgp-ipv6u)# neighbor fe80:4398:ab30:45de::1 route-map out myroutemap
device(config-bgp-ipv6u)# exit
device(config)# route-map myroutemap permit 10
device(config-routemap-myroutemap)# set ipv6 next-hop 2001::10
```

# Configuring BGP4+ neighbors using link-local addresses in the IPv6 multicast address family

BGP4+ neighbors can be configured using link-local addresses in the IPv6 multicast address family.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **router bgp** command to enable BGP routing.

```
device(config)# router bgp
```

3. Enter the **local-as** command to configure the autonomous system number (ASN) in which your device resides.

```
device(config-bgp)# local-as 1001
```

4. Enter the **neighbor remote-as** command, specifying an IPv6 address, to specify the ASN in which the remote neighbor resides.

```
device(config-bgp)# neighbor fe80:4398:ab30:45de::2 remote-as 1002
```

5. Enter the **neighbor update-source** command to specify an interface.

```
device(config-bgp)# neighbor fe80:4398:ab30:45de::2 update-source ethernet 3/1
```

6. Enter the **address-family ipv6 multicast** command to enter IPv6 address family mulitcast configuration mode.

```
device(config-bgp)# address-family ipv6 multicast
```

7. Enter the **neighbor activate** command to enable the exchange of information with the neighbor.

```
device(config-bgp-ipv6m)# neighbor fe80:4398:ab30:45de::2 activate
```

8. Enter the **neighbor route-map** command and specify the **out** keyword to apply a route map to outgoing routes.

```
device(config-bgp-ipv6m)# neighbor fe80:4398:ab30:45de::2 route-map out myroutemap1
```

9. Enter the **exit** command until you return to global configuration mode.

```
device(config-bgp-ipv6m)# exit
```

10. Enter the **route-map** *name* **permit** command to define the route map and enter route-map configuration mode.

```
device(config)# route-map myroutemap1 permit 11
```

11. Enter the **set ipv6 next-hop** command and specify an IPv6 address to set the IPv6 address of the next hop.

```
device(config-routemap myroutemap1)# set ipv6 next-hop 2001::10
```

The following example configures a neighbor using a link-local address and configures a route map to set up a global next hop for packets destined for the neighbor.

```
device# configure terminal
device(config)# router bgp
device(config-bgp)# local-as 1001
device(config-bgp)# neighbor fe80:4398:ab30:45de::2 remote-as 1002
device(config-bgp)# neighbor fe80:4398:ab30:45de::2 update-source ethernet 3/1
device(config-bgp)# address-family ipv6 multicast
device(config-bgp-ipv6m)# neighbor fe80:4398:ab30:45de::2 activate
device(config-bgp-ipv6m)# neighbor fe80:4398:ab30:45de::2 route-map out myroutemap1
device(config-bgp-ipv6m)# exit
device(config)# route-map myroutemap1 permit 11
device(config-routemap myroutemap1)# set ipv6 next-hop 2001::10
```

# BGP4+ peer groups

Neighbors having the same attributes and parameters can be grouped together by means of the **neighbor peer-group** command.

You must first create a peer group, after which you can associate neighbor IPv6 addresses with the peer group. All of the attributes that are allowed on a neighbor are allowed on a peer group as well.

BGP4+ peers and peer groups are activated in the IPv6 address family configuration mode to establish the BGP4+ peering sessions.

An attribute value configured explicitly for a neighbor takes precedence over the attribute value configured on the peer group. In the case where neither the peer group nor the individual neighbor has the attribute configured, the default value for the attribute is used.

> NOTE
> BGP4 neighbors are established and the prefixes are advertised using the **neighbor** *IP address* **remote-as** command in router BGP mode. However, when establishing BGP4+ peer sessions and exchanging IPv6 prefixes, neighbors must also be activated using the **neighbor** *IPv6 address* **activate** command in IPv6 address family configuration mode.

## Configuring BGP4+ peer groups

A peer group can be created and neighbor IPv6 addresses can be associated with the peer group. The peer group is then activated in the IPv6 address family configuration mode.

1. Enter the **configure terminal** command to access global configuration mode.

   ```
   device# configure terminal
   ```

2. Enter the **router bgp** command to enable BGP routing.

   ```
   device(config)# router bgp
   ```

3. Enter the **local-as** command to configure the autonomous system number (ASN) in which your device resides.

   ```
   device(config-bgp)# local-as 1000
   ```

4. Enter the **neighbor peer-group** command to create a peer group.

   ```
   device(config-bgp)# neighbor mypeergroup1 peer-group
   ```

5. Enter the **neighbor remote-as** command, specifying a peer group, to specify the ASN of the peer group.

   ```
   device(config-bgp)# neighbor mypeergroup1 remote-as 11
   ```

6. Enter the **neighbor peer-group** command to associate a neighbor with the peer group.

```
device(config-bgp)# neighbor 2001:2018:8192::125 peer-group mypeergroup1
```

7. Enter the **neighbor peer-group** command to associate another neighbor with the peer group.

```
device(config-bgp)# neighbor 2001:2018:8192::124 peer-group mypeergroup1
```

8. Enter the **address-family ipv6 unicast** command to enter IPv6 address family configuration mode.

```
device(config-bgp)# address-family ipv6 unicast
```

9. Enter the **neighbor activate** command to establish an IPv6 BGP session with the peer group.

```
device(config-bgp-ipv6u)# neighbor mypeergroup1 activate
```

The following example creates a peer group, specifying two neighbors to belong to the peer group, and activates the peer group.

```
device# configure terminal
device(config)# router bgp
device(config-bgp)# local-as 1000
device(config-bgp)# neighbor mypeergroup1 peer-group
device(config-bgp)# neighbor mypeergroup1 remote-as 11
device(config-bgp)# neighbor 2001:2018:8192::125 peer-group mypeergroup1
device(config-bgp)# neighbor 2001:2018:8192::124 peer-group mypeergroup1
device(config-bgp)# address-family ipv6 unicast
device(config-bgp-ipv6u)# neighbor mypeergroup1 activate
```

## Configuring a peer group with IPv4 and IPv6 peers

A peer group that contains both IPv4 and IPv6 peers can be configured.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **router bgp** command to enable BGP routing.

```
device(config)# router bgp
```

3. Enter the **local-as** command to configure the autonomous system number (ASN) in which your device resides.

```
device(config-bgp)# local-as 1000
```

4. Enter the **neighbor peer-group** command, specifying a peer group, to create a peer group.

```
device(config-bgp)# neighbor p1 peer-group
```

5. Enter the **neighbor remote-as** command to specify the ASN of the peer group.

```
device(config-bgp)# neighbor p1 remote-as 11
```

6. Enter the **neighbor peer-group** command, specifying an IPv6 address, to associate a neighbor with the peer group.

```
device(config-bgp)# neighbor 2001:2018:8192::124 peer-group p1
```

7. Enter the **neighbor peer-group** command, specifying a different IPv6 address, to associate another neighbor with the peer group.

```
device(config-bgp)# neighbor 10.0.0.1 peer-group p1
```

8. Enter the **address-family ipv6 unicast** command to enter IPv6 address family configuration mode.

```
device(config-bgp)# address-family ipv6 unicast
```

9. Enter the **neighbor activate** command to establish an IPv6 BGP session with the peer group.

```
device(config-bgp-ipv6u)# neighbor p1 activate
```

The following example creates a peer group with both IPv6 and IPv4 peers and activates the peer group in the IPv6 address family.

```
device# configure terminal
device(config)# router bgp
device(config-bgp)# local-as 1000
device(config-bgp)# neighbor p1 peer-group
device(config-bgp)# neighbor p1 remote-as 11
device(config-bgp)# neighbor 2001:2018:8192::124 peer-group p1
device(config-bgp)# neighbor 10.0.0.1 peer-group p1
device(config-bgp)# address-family ipv6 unicast
device(config-bgp-ipv6u)# neighbor p1 activate
```

# Importing routes into BGP4+

Routes can be explicitly specified for advertisement by BGP.

The routes imported into BGP4+ must first exist in the IPv6 multicast or unicast route table.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **router bgp** command to enable BGP routing.

```
device(config)# router bgp
```

3. Enter the **neighbor remote-as** command, specifying an IPv6 address, to specify the ASN in which the remote neighbor resides.

```
device(config-bgp)# neighbor fe80:4398:ab30:45de::1 remote-as 1001
```

4. Enter the **address-family ipv6 unicast** command to enter IPv6 address family configuration mode.

```
device(config-bgp)# address-family ipv6 unicast
```

5. Enter the **network** command and specify a *network/mask* to import the specified prefix into the BGP4+ database.

```
device(config-bgp-ipv6u)# network 2001:db8::/32
```

The following example imports the 2001:db8::/32 prefix in to the BGP4+ database for advertising.

```
device# configure terminal
device(config)# router bgp
device(config-bgp)# neighbor fe80:4398:ab30:45de::1 remote-as 1001
device(config-bgp)# address-family ipv6 unicast
device(config-bgp-ipv6u)# network 2001:db8::/32
```

# Advertising the default BGP4+ route

A BGP device can be configured to advertise the default IPv6 route to all BGP4+ neighbors and to install that route in the local BGP4+ route table.

The default route must be present in the local IPv6 route table.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **router bgp** command to enable BGP routing.

```
device(config)# router bgp
```

3. Enter the **address-family ipv6 unicast** command to enter IPv6 address family unicast configuration mode.

```
device(config-bgp)# address-family ipv6 unicast
```

4. Enter the **default-information-originate** command to advertise the default IPv6 route to all BGP4+ neighbors and to install that route in the local BGP4+ route table.

```
device(config-bgp-ipv6u)# default-information-originate
```

The following example enables a BGP4+ device to advertise the default IPv6 route to all BGP4+ neighbors and to install that route in the local BGP4+ route table.

```
device# configure terminal
device(config)# router bgp
device(config-bgp)# address-family ipv6 unicast
device(config-bgp-ipv6u)# default-information-originate
```

# Advertising the default BGP4+ route to a specific neighbor

A BGP device can be configured to advertise the default IPv6 route to a specific neighbor.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **router bgp** command to enable BGP routing.

```
device(config)# router bgp
```

3. Enter the **local-as** command to configure the autonomous system number (ASN) in which your device resides.

```
device(config-bgp)# local-as 1000
```

4. Enter the **address-family ipv6 unicast** command to enter IPv6 address family configuration mode.

```
device(config-bgp)# address-family ipv6 unicast
```

5. Enter the **neighbor default-originate** command and specify an IPv6 address to enable the BGP4+ device to advertise the default IPv6 route to a specific neighbor.

```
device(config-bgp-ipv6u)# neighbor 2001:db8:93e8:cc00::1 default-originate
```

The following example enables a BGP4+ device to advertise the default IPv6 route to a specific neighbor.

```
device# configure terminal
device(config)# router bgp
device(config-bgp)# local-as 1000
device(config-bgp)# address-family ipv6 unicast
device(config-bgp-ipv6u)# neighbor 2001:db8:93e8:cc00::1 default-originate
```

# Using the IPv6 default route as a valid next hop for a BGP4+ route

In certain cases, such as when a device is acting as an edge device, it can be configured to use the default route as a valid next hop.

By default, a device does not use a default route to resolve a BGP4+ next-hop route. If the IPv6 route lookup for the BGP4+ next-hop does not result in a valid IGP route (including static or direct routes), the BGP4+ next-hop is considered to be unreachable and the BGP4+ route is not used. You can configure the device to use the default route as a valid next hop.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **router bgp** command to enable BGP routing.

```
device(config)# router bgp
```

3. Enter the **address-family ipv6 unicast** command to enter IPv6 address family configuration mode.

```
device(config-bgp)# address-family ipv6 unicast
```

4. Enter the **next-hop-enable-default** command to configure the device to use the default route as a valid next hop.

```
device(config-bgp-ipv6u)# next-hop-enable-default
```

The following example configures a BGP4+ device to use the default route as a valid next hop.

```
device# configure terminal
device(config)# router bgp
device(config-bgp)# address-family ipv6 unicast
device(config-bgp-ipv6u)# next-hop-enable-default
```

# BGP4+ next hop recursion

A device can find the IGP route to the next-hop gateway for a BGP4+ route.

For each BGP4+ route learned, the device performs a route lookup to obtain the IPv6 address of the next hop for the route. A BGP4+ route is eligible for addition in the IPv6 route table only if the following conditions are true:

- The lookup succeeds in obtaining a valid next-hop IPv6 address for the route.
- The path to the next-hop IPv6 address is an IGP path or a static route path.

By default, the software performs only one lookup for the next-hop IPv6 address for the BGP4+ route. If the next hop lookup does not result in a valid next hop IPv6 address, or the path to the next hop IPv6 address is a BGP4+ path, the BGP4+ route destination is considered unreachable. The route is not eligible to be added to the IPv6 route table.

The BGP4+ route table can contain a route with a next hop IPv6 address that is not reachable through an IGP route, even though the device can reach a hop farther away through an IGP route. This can occur when the IGPs do not learn a complete set of IGP routes, so the device learns about an internal route through IBGP instead of through an IGP. In this case, the IPv6 route table will not contain a route that can be used to reach the BGP4+ route destination.

To enable the device to find the IGP route to the next-hop gateway for a BGP4+ route, enable recursive next-hop lookups. With this feature enabled, if the first lookup for a BGP4+ route results in an IBGP path that originated within the same AS, rather than an IGP path or static route path, the device performs a lookup on the next hop IPv6 address for the next hop gateway. If this second lookup results in an IGP path, the software considers the BGP4+ route to be valid and adds it to the IPv6 route table. Otherwise, the device performs another lookup on the next hop IPv6 address of the next hop for the next hop gateway, and so on, until one of the lookups results in an IGP route.

You must configure a static route or use an IGP to learn the route to the EBGP multihop peer.

# Enabling next-hop recursion

Next hop recursion can be enabled so that a device can find the IGP route to the next hop gateway for a BGP4+ route.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **router bgp** command to enable BGP routing.

```
device(config)# router bgp
```

3. Enter the **address-family ipv6 unicast** command to enter IPv6 address family configuration mode.

```
device(config-bgp)# address-family ipv6 unicast
```

4. Enter the **next-hop-recursion** command to enable recursive next hop lookups.

```
device(config-bgp-ipv6u)# next-hop-recursion
```

The following example enables recursive next hop lookups.

```
device# configure terminal
device(config)# router bgp
device(config-bgp)# address-family ipv6 unicast
device(config-bgp-ipv6u)# next-hop-recursion
```

# Enabling next-hop recursion in the IPv6 multicast address family

Next hop recursion can be enabled so that a device can find the IGP route to the next hop gateway for a BGP4+ route.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **router bgp** command to enable BGP routing.

```
device(config)# router bgp
```

3. Enter the **address-family ipv6 multicast** command to enter IPv6 address family multicast configuration mode.

```
device(config-bgp)# address-family ipv6 multicast
```

4. Enter the **next-hop-recursion** command to enable recursive next hop lookups.

```
device(config-bgp-ipv6m)# next-hop-recursion
```

The following example enables recursive next hop lookups.

```
device# configure terminal
device(config)# router bgp
device(config-bgp)# address-family ipv6 multicast
device(config-bgp-ipv6m)# next-hop-recursion
```

# BGP4+ NLRIs and next hop attributes

BGP4+ introduces new attributes to handle multiprotocol extensions for BGP.

Multiprotocol BGP (MBGP) is an extension to BGP that enables BGP to carry routing information for multiple address families.

BGP4+ introduces new attributes to handle multiprotocol extensions for BGP:

- Multiprotocol reachable Network Layer Reachability Information (MP_REACH_NLRI): Used to carry the set of reachable destinations, together with the next hop information, to be used for forwarding to these destinations.
- Multiprotocol unreachable NLRI (MP_UNREACH_NLRI): Used to carry the set of unreachable destinations.

MP_REACH_NLRI and MP_UNREACH_NLRI are optional and non-transitive, so that a BGP4+ speaker that does not support the multiprotocol capabilities ignores the information carried in these attributes, and does not pass it to other BGP4+ speakers. A BGP speaker that uses multiprotocol extensions for IPv6 uses the capability advertisement procedures to determine whether the speaker can use multiprotocol extensions with a particular peer.

The next hop information carried in the MP_REACH_NLRI path attribute defines the network layer address of the border router that will be used as the next hop to the destinations listed in the MP_NLRI attribute in the UPDATE message.

MP_REACH_NLRI and MP_UNREACH_NLRI carry IPv6 prefixes.

# BGP4+ route reflection

A BGP device can act as a route-reflector client or as a route reflector. You can configure a BGP peer as a route-reflector client from the device that is going to reflect the routes and act as the route reflector using the **neighbor route-reflector-client** command.

When there is more than one route reflector, they should all belong to the same cluster. By default, the value for **cluster-id** is used as the device ID. The device ID can be changed using the **cluster-id** command.

The route-reflector server reflects the routes as follows:

- Routes from the client are reflected to the client as well as to nonclient peers.
- Routes from nonclient peers are reflected only to client peers.

If route-reflector clients are connected in a full IBGP mesh, you can disable client-to-client reflection on the route reflector using the **no client-to-client-reflection** command.

A BGP device advertises only those routes that are preferred ones and are installed into the Routing Table Manager (RTM). When a route cannot be installed into the RTM because the routing table is full, the route reflector may not reflect that route. In cases where the route

reflector is not placed directly in the forwarding path, you can configure the route reflector to reflect routes even though those routes are not in the RTM using the **always-propagate** command.

# Configuring a cluster ID for a route reflector

The cluster ID can be changed if there is more than one route reflector, so that all route reflectors belong to the same cluster.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **router bgp** command to enable BGP routing.

```
device(config)# router bgp
```

3. Enter the **local-as** command to configure the autonomous system number (ASN) in which your device resides.

```
device(config-bgp)# local-as 1000
```

4. Enter the **cluster-id** command and specify a value to change the cluster ID of a device from the default device ID.

```
device(config-bgp)# cluster-id 321
```

The following example changes the cluster ID of a device from the default device ID to 321.

```
device# configure terminal
device(config)# router bgp
device(config-bgp)# cluster-id 321
```

# Configuring a route reflector client

A BGP peer can be configured as a route reflector client.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **router bgp** command to enable BGP routing.

```
device(config)# router bgp
```

3. Enter the **local-as** command to configure the autonomous system number (ASN) in which your device resides.

```
device(config-bgp)# local-as 1000
```

4. Enter the **address-family ipv6 unicast** command to enter IPv6 address family configuration mode.

```
device(config-bgp)# address-family ipv6 unicast
```

5. Enter the **neighbor route-reflector-client** command, specifying an IPv6 address, to configure a specified neighbor to be a route reflector client.

```
device(config-bgp-ipv6u)# neighbor 2001:db8:e0ff:783a::4 route-reflector-client
```

The following example configures a neighbor with the IPv6 address 2001:db8:e0ff:783a::4 to be a route reflector client.

```
device# configure terminal
device(config)# router bgp
device(config-bgp)# local-as 1000
device(config-bgp)# address-family ipv6 unicast
device(config-bgp-ipv6u)# neighbor 2001:db8:e0ff:783a::4 route-reflector-client
```

# BGP4+ route aggregation

A device can be configured to aggregate routes in a range of networks into a single IPv6 prefix.

By default, a device advertises individual BGP4+ routes for all the networks. The aggregation feature allows you to configure a device to aggregate routes in a range of networks into a single IPv6 prefix. For example, without aggregation, a device will individually advertise routes for networks 2001:db8:0001:0000::/64, 2001:db8:0002:0000::/64, 2001:db8:0003:0000::/64, and so on. You can configure the device to send a single, aggregate route for the networks instead so that the aggregate route would be advertised as 2001:db8::/32 to BGP4 neighbors.

## Aggregating routes advertised to BGP neighbors

A device can be configured to aggregate routes in a range of networks into a single IPv6 prefix.

The route-map should already be defined.

You can aggregate BGP4+ routes, for example 2001:db8:0001:0000::/64, 2001:db8:0002:0000::/64, 2001:db8:0003:0000::/64 into a single network prefix: 2001:db8::/24.

1. Enter the **configure terminal** command to access global configuration mode.

   ```
   device# configure terminal
   ```

2. Enter the **router bgp** command to enable BGP routing.

   ```
   device(config)# router bgp
   ```

3. Enter the **address-family ipv6 unicast** command to enter IPv6 address family configuration mode.

   ```
   device(config-bgp)# address-family ipv6 unicast
   ```

4. Enter the **aggregate-address** command to aggregate the routes from a range of networks into a single network prefix.

   ```
   device(config-bgp-ipv6u)# aggregate-address 2001:db8::/32
   ```

The following example enables a BGP4+ device to advertise the default route and send the default route to a specified neighbor.

```
device# configure terminal
device(config)# router bgp
device(config-bgp)# address-family ipv6 unicast
device(config-bgp-ipv6u)# aggregate-address 2001:db8::/32
```

# BGP4+ multipath

The BGP4+ multipath feature can be used to enable load-balancing across different paths.

BGP4+ selects only one best path for each IPv6 prefix it receives before installing it in the IP routing table. If you need load-balancing across different paths, you must enable BGP4+ multipath using the **maximum-paths** command under IPv6 address family configuration mode.

IBGP paths and EBGP paths can be exclusively selected, or a combination of IBGP and EBGP paths can be selected.

The following attributes of parallel paths must match for them to be considered for multipathing:

- Weight
- Local Preference
- Origin
- AS-Path Length
- MED
- Neighbor AS (EBGP multipath)
- AS-PATH match (for IBGP multipath)
- IGP metric to BGP next hop

## Enabling load-balancing across different paths

The BGP4+ multipath feature can be configured, enabling load-balancing across different paths.

1. Enter the **configure terminal** command to access global configuration mode.

   ```
   device# configure terminal
   ```

2. Enter the **router bgp** command to enable BGP routing.

   ```
   device(config)# router bgp
   ```

3. Enter the **address-family ipv6 unicast** command to enter IPv6 address family configuration mode.

   ```
   device(config-bgp)# address-family ipv6 unicast
   ```

4. Do one of the following:
   - Enter the **maximum-paths** command and specify a value to set the maximum number of BGP4+ shared paths.
   - Enter the **maximum-paths** command using the **use-load-sharing** keyword to set the maximum number of BGP4+ shared paths to that of the value already configured by means of the **ip load-sharing** command.

   ```
   device(config-bgp-ipv6u)# maximum-paths 8
   ```

   or

   ```
   device(config-bgp-ipv6u)# maximum-paths use-load-sharing
   ```

The following example sets the maximum number of BGP4+ shared paths to 8.

```
device# configure terminal
device(config)# router bgp
device(config-bgp)# address-family ipv6 unicast
device(config-bgp-ipv6u)# maximum-paths 8
```

The following example sets the maximum number of BGP4+ shared paths to that of the value already configured using the **ip load-sharing** command.

```
device# configure terminal
device(config)# router bgp
device(config-bgp)# address-family ipv6 unicast
device(config-bgp-ipv6u)# maximum-paths use-load-sharing
```

# Route maps

Route maps must be applied to IPv6 unicast address prefixes in IPv6 address family configuration mode.

By default, route maps that are applied under IPv4 address family configuration mode using the **neighbor route-map** command are applied to only IPv4 unicast address prefixes. To apply route maps to IPv6 unicast address prefixes, the **neighbor route-map** command must be used in IPv6 address family configuration mode. The route maps are applied as the inbound or outbound routing policy for neighbors under the specified address family. Configuring separate route maps under each address family type simplifies managing complicated or different policies for each address family.

## Configuring a route map for BGP4+ prefixes

Route maps can be applied to IPv6 unicast address prefixes either as the inbound or outbound routing policy for neighbors under the specified address family.

1. Enter the **configure terminal** command to access global configuration mode.

   ```
   device# configure terminal
   ```

2. Enter the **ipv6 prefix-list** command and enter a name to configure an IPv6 prefix list.

   ```
   device(config)# ipv6 prefix-list myprefixlist seq 10 permit 2001:db8::/32
   ```

   The prefix list name, sequence number, and permits packets are specified.

3. Enter the **route-map** command with the **permit** keyword, and specify a route map name, to define the route map and enter route map configuration mode.

   ```
   device(config)# route-map myroutemap permit 10
   ```

4. Enter the **match ipv6 address** command and specify the name of a prefix list.

   ```
   device(config-route-map-myroutemap)# match ipv6 address prefix-list myprefixlist
   ```

5. Enter the **exit** command to return to global configuration mode.

   ```
   device(config-route-map-myroutemap)# exit
   ```

6. Enter the **router bgp** command to enable BGP routing.

   ```
   device(config)# router bgp
   ```

7. Enter the **local-as** command to configure the autonomous system number (ASN) in which your device resides.

   ```
   device(config-bgp)# local-as 1000
   ```

8. Enter the **neighbor remote-as** command, specifying an IPv6 address, to specify the ASN in which the remote neighbor resides.

   ```
   device(config-bgp)# neighbor fe80:4398:ab30:45de::1 remote-as 1001
   ```

9. Enter the **address-family ipv6 unicast** command to enter IPv6 address family configuration mode.

```
device(config-bgp)# address-family ipv6 unicast
```

10. Enter the **neighbor activate** command to enable the exchange of information with the neighbor.

```
device(config-bgp-ipv6u)# neighbor fe80:4398:ab30:45de::1 activate
```

11. Enter the **neighbor route-map** command and specify the **out** keyword to apply a route map to outgoing routes.

```
device(config-bgp-ipv6u)# neighbor fe80:4398:ab30:45de::1 route-map out myroutemap
```

The following example applies a route map, "myroutemap", as the outbound routing policy for a neighbor.

```
device# configure terminal
device(config)# ipv6 prefix-list myprefixlist seq 10 permit 2001:db8::/32
device(config)# route-map myroutemap permit 10
device(config-route-map-myroutemap)# match ipv6 address prefix-list myprefixlist
device(config-route-map-myroutemap)# exit
device(config)# router bgp
device(config-bgp)# local-as 1000
device(config-bgp)# neighbor fe80:4398:ab30:45de::1 remote-as 1001
device(config-bgp)# address-family ipv6 unicast
device(config-bgp-ipv6u)# neighbor fe80:4398:ab30:45de::1 activate
device(config-bgp-ipv6u)# neighbor fe80:4398:ab30:45de::1 route-map out myroutemap
```

# Redistributing prefixes into BGP4+

Various routes can be redistributed into BGP.

Static, connected, OSPF, ISIS, and RIPng routes can be redistributed into BGP. This task redistributes RIPng routes into BGP4+.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **router bgp** command to enable BGP routing.

```
device(config)# router bgp
```

3. Enter the **address-family ipv6 unicast** command to enter IPv6 address family configuration mode.

```
device(config-bgp)# address-family ipv6 unicast
```

4. Enter the **redistribute** command using the **rip** keyword to redistribute IPv6 RIP routes.

```
device(config-bgp-ipv6u)# redistribute rip
```

The following example redistributes RIPng prefixes into BGP4+.

```
device# configure terminal
device(config)# router bgp
device(config-bgp)# address-family ipv6 unicast
device(config-bgp-ipv6u)# redistribute rip
```

# Redistributing routes into BGP4+

Various routes can be redistributed into BGP. This task redistributes connected routes into BGP.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **router bgp** command to enable BGP routing.

```
device(config)# router bgp
```

3. Enter the **address-family ipv6 unicast** command to enter IPv6 address family unicast configuration mode.

```
device(config-bgp)# address-family ipv6 unicast
```

4. Enter the **redistribute** command using the **connected** keyword to redistribute connected routes into BGP4+.

```
device(config-bgp-ipv6u)# redistribute connected
```

The following example redistributes connected routes into BGP4+.

```
device# configure terminal
device(config)# router bgp
device(config-bgp)# address-family ipv6 unicast
device(config-bgp-ipv6u)# redistribute connected
```

# Specifying the weight added to BGP4+ received routes

The weight that the device adds to received routes can be specified. The following task changes the weight from the default for routes that are received from a specified BGP4+ neighbor.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **router bgp** command to enable BGP routing.

```
device(config)# router bgp
```

3. Enter the **local-as** command to configure the autonomous system number (ASN) in which your device resides.

```
device(config-bgp)# local-as 65520
```

4. Enter the **address-family ipv6 unicast** command to enter address family IPv6 unicast configuration mode.

```
device(config-bgp)# address-family ipv6 unicast
```

5. Enter the **neighbor weight** command and specify an *ipv6 address* and a weight value to specify a weight that the device adds to routes that are received from the specified BGP4+ neighbor.

```
device(config-bgp-ipv6u)# neighbor 2001:db8:e0ff:783a::4 weight 200
```

The following example specifies a weight of 200 that the device adds to routes that are received from the specified BGP4+ neighbor.

```
device# configure terminal
device(config)# router bgp
device(config-bgp)# local-as 65520
device(config-bgp)# address-family ipv6 unicast
device(config-bgp-ipv6u)# neighbor 2001:db8:e0ff:783a::4 weight 200
```

# Enabling BGP4+ in a non-default VRF

When BGP4+ is enabled in a non-default VRF instance, the device enters BGP address-family IPv6 unicast VRF configuration mode. Several commands can then be accessed that allow the configuration of BGP4+ for the non-default VRF instance.

A non-default VRF instance has been configured.

1. Enter the **configure terminal** command to access global configuration mode.

   ```
   device# configure terminal
   ```

2. Enter the **router bgp** command to enable BGP routing globally.

   ```
   device(config)# router bgp
   ```

3. Enter the **address-family ipv6 unicast** command with the **vrf** parameter, and specify a VRF name, to enter BGP address-family IPv6 unicast VRF configuration mode.

   ```
   device(config-bgp)# address-family ipv6 unicast vrf green
   ```

The following example enables BGP address-family IPv6 unicast VRF configuration mode where several commands can be accessed that allow the configuration of BGP4+ for the non-default VRF instance..

```
device# configure terminal
device(config)# router bgp
device(config-bgp)# address-family ipv6 unicast vrf green
device(config-bgp-ipv6u-vrf)#
```

# BGP4+ outbound route filtering

The BGP4+ Outbound Route Filtering Capability (ORF) feature is used to minimize the number of BGP updates sent between BGP peers.

When the ORF feature is enabled, unwanted routing updates are filtered out, reducing the amount of system resources required for generating and processing routing updates. The ORF feature is enabled through the advertisement of ORF capabilities to peer routers. The locally configured BGP4+ inbound prefix filters are sent to the remote peer so that the remote peer applies the filter as an outbound filter for the neighbor.

The ORF feature can be configured with send and receive ORF capabilities. The local peer advertises the ORF capability in send mode, indicating that it will accept a prefix list from a neighbor and apply the prefix list to locally configured ORFs. The local peer exchanges the ORF capability in send mode with a remote peer for a prefix list that is configured as an inbound filter for that peer locally. The remote peer only sends the first update once it receives a ROUTEREFRESH request or BGP ORF with IMMEDIATE from the peer. The local and remote peers exchange updates to maintain the ORF on each router.

# Configuring BGP4+ outbound route filtering

The BGP4+ Outbound Route Filtering (ORF) prefix list capability can be configured in receive mode, send mode, or both send and receive modes, minimizing the number of BGP updates exchanged between BGP peers.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **router bgp** command to enable BGP routing.

```
device(config)# router bgp
```

3. Enter the **address-family ipv6 unicast** command to enter IPv6 address family configuration mode.

```
device(config-bgp)# address-family ipv6 unicast
```

4. Enter the **neighbor activate** command, specifying an IPv6 address, to add a neighbor.

```
device(config-bgp-ipv6u)# neighbor 2001:db8:e0ff:783a::4 activate
```

5. Enter the **neighbor prefix-list** command, specify an IPv6 address and the **in** keyword to filter the incoming route updates from a specified BGP neighbor.

```
device(config-bgp-ipv6u)# neighbor 2001:db8:e0ff:783a::4 prefix-list myprefixlist in
```

6. Do one of the following:
   - Enter the **neighbor capability orf prefixlist** command and specify the **send** keyword to advertise ORF send capabilities.

```
device(config-bgp-ipv6u)# neighbor 2001:db8:e0ff:783a::4 capability orf prefixlist send
```

   - Enter the **neighbor capability orf prefixlist** command and specify the **receive** keyword to advertise ORF receive capabilities.

```
device(config-bgp-ipv6u)# neighbor 2001:db8:e0ff:783a::4 capability orf prefixlist receive
```

   - Enter the **neighbor capability orf prefixlist** command to configure ORF capability in both send and receive modes.

```
device(config-bgp-ipv6u)# neighbor 2001:db8:e0ff:783a::4 capability orf prefixlist
```

The following example configures ORF in receive mode.

```
device# configure terminal
device(config)# router bgp
device(config-bgp)# address-family ipv6 unicast
device(config-bgp-ipv6u)# neighbor 2001:db8:e0ff:783a::4 activate
device(config-bgp-ipv6u)# neighbor 2001:db8:e0ff:783a::4 capability orf prefixlist receive
```

The following example configures ORF in send mode.

```
device# configure terminal
device(config)# router bgp
device(config-bgp)# address-family ipv6 unicast
device(config-bgp-ipv6u)# neighbor 2001:db8:e0ff:783a::4 activate
device(config-bgp-ipv6u)# neighbor 2001:db8:e0ff:783a::4 prefix-list myprefixlist in
device(config-bgp-ipv6u)# neighbor 2001:db8:e0ff:783a::4 capability orf prefixlist send
```

The following example configures ORF in both send and receive modes.

```
device# configure terminal
device(config)# router bgp
device(config-bgp)# address-family ipv6 unicast
device(config-bgp-ipv6u)# neighbor 2001:db8:e0ff:783a::4 activate
device(config-bgp-ipv6u)# neighbor 2001:db8:e0ff:783a::4 prefix-list myprefixlist in
device(config-bgp-ipv6u)# neighbor 2001:db8:e0ff:783a::4 capability orf prefixlist
```

# BGP4+ confederations

A large autonomous system (AS) can be divided into multiple subautonomous systems and grouped into a single BGP4+ confederation.

Each subautonomous system must be uniquely identified within the confederation AS by a subautonomous system number. Within each subautonomous system, all the rules of internal BGP (IBGP) apply. For example, all BGP routers inside the subautonomous system must be fully meshed. Although EBGP is used between subautonomous systems, the subautonomous systems within the confederation exchange routing information like IBGP peers. Next hop, Multi Exit Discriminator (MED), and local preference information is preserved when crossing subautonomous system boundaries. To the outside world, a confederation looks like a single AS.

The AS path list is a loop-avoidance mechanism used to detect routing updates leaving one subautonomous system and attempting to re-enter the same subautonomous system. A routing update attempting to re-enter a subautonomous system it originated from is detected because the subautonomous system sees its own subautonomous system number listed in the update's AS path.

## Configuring BGP confederations

BGP confederations, composed of multiple subautonomous systems, can be created.

1. Enter the **configure terminal** command to access global configuration mode.

   ```
   device# configure terminal
   ```

2. Enter the **router bgp** command to enable BGP routing.

   ```
   device(config)# router bgp
   ```

3. Enter the **local-as** command to configure the autonomous system number (ASN) in which your device resides.

   ```
   device(config-bgp)# local-as 65520
   ```

4. Enter the **confederation identifier** command and specify an ASN to configure a BGP confederation identifier.

   ```
   device(config-bgp)# confederation identifier 100
   ```

5. Enter the **confederation peers** command and specify as many ASNs as needed to list all BGP peers that will belong to the confederation.

   ```
   device(config-bgp)# confederation peers 65520 65521 65522
   ```

The following example creates a confederation with the confederation ID "100" and adds three subautonomous systems to the confederation.

```
device# configure terminal
device(config)# router bgp
device(config-bgp)# local-as 65520
device(config-bgp)# confederation identifier 100
device(config-bgp)# confederation peers 65520 65521 65522
```

# BGP4+ extended community

The BGP4+ extended community feature filters routes based on a regular expression specified when a route has multiple community values in it.

A BGP community is a group of destinations that share a common property. Community information identifying community members is included as a path attribute in BGP UPDATE messages. You can perform actions on a group using community and extended community attributes to trigger routing decisions. All communities of a particular type can be filtered out, or certain values can be specified for a particular type of community. You can also specify whether a particular community is transitive or non-transitive across an autonomous system (AS) boundary.

An extended community is an 8-octet value and provides a larger range for grouping or categorizing communities. BGP extended community attributes are specified in RFC 4360.

You define the extended community list using the **ip extcommunity-list** command. The extended community can then be matched or applied to the neighbor through the route map. The route map must be applied on the neighbor to which routes need to carry the extended community attributes. The "send-community" should be enabled for the neighbor configuration to start including the attributes while sending updates to the neighbor.

## Defining BGP extended communities

In order to apply a BGP extended community filter, a BGP extended community filter must be defined.

1. Enter the **configure terminal** command to access global configuration mode.

   ```
   device# configure terminal
   ```

2. Enter the **ip community-list extended** command and specify a number to set a BGP extended community filter.

   ```
   device(config)# ip community-list extended 1 permit soo 123:2
   ```

3. Enter the **route-map** *name* command to create and define a route map and enter route-map configuration mode.

   ```
   device(config)# route-map extComRmap permit 10
   ```

   Permits a matching pattern.

4. Enter the **match community** command and specify a community list number.

   ```
   device(config-route-map-extComRmap)# match community 1
   ```

5. Enter the **exit** command to return to global configuration mode.

   ```
   device(config-route-map-extComRmap)# exit
   ```

6. Enter the **route-map** command with the **permit** keyword, specifying a route map name, to define a route map and enter route-map configuration mode.

   ```
   device(config)# route-map sendExtComRmap permit 10
   ```

   Permits a matching pattern.

7. Enter the **set community** command and specify the *community value* parameter to specify the community attribute.

   ```
   device(config-route-map-sendExtComRmap)# set community 3:3
   ```

The following example configures an extended community ACL called “extended”, defines a route map, and permits and sets a matching pattern.

```
device# configure terminal
device(config)# ip community-list extended 1 permit soo 123:2
device(config)# route-map extComRmap permit 10
device(config-route-map-extComRmap)# match community 1
device(config-route-map-extComRmap)# exit
device(config)# route-map sendExtComRmap permit 10
device(config-route-map-sendExtComRmap)# set community 3:3
```

# Applying a BGP4+ extended community filter

A BGP4+ extended community filter can be applied .

BGP4+ communities must already be defined.

1. Enter the **configure terminal** command to access global configuration mode.

   ```
   device# configure terminal
   ```

2. Enter the **ip extcommunity-list** command and specify a number to set a BGP extended community filter.

   ```
   device(config)# ip extcommunity-list 1 permit rt 123:2
   ```

3. Enter the **router bgp** command to enable BGP routing.

   ```
   device(config)# router bgp
   ```

4. Enter the **local-as** command to configure the autonomous system number (ASN) in which your device resides.

   ```
   device(config-bgp)# local-as 1000
   ```

5. Enter the **neighbor remote-as** command, specifying an IPv6 address, to specify the ASN in which the remote neighbor resides.

   ```
   device(config-bgp)# neighbor fe80:4398:ab30:45de::1 remote-as 1001
   ```

6. Enter the **address-family ipv6 unicast** command to enter IPv6 address family configuration mode.

   ```
   device(config-bgp)# address-family ipv6 unicast
   ```

7. Enter the **neighbor activate** command to enable the exchange of information with the neighbor.

   ```
   device(config-bgp-ipv6u)# neighbor fe80:4398:ab30:45de::1 activate
   ```

8. Enter the **neighbor route-map** command and specify the **in** keyword to apply a route map to incoming routes.

   ```
   device(config-bgp-ipv6u)# neighbor fe80:4398:ab30:45de::1 route-map in extComRmapt
   ```

9. Enter the **neighbor route-map** command and specify the **out** keyword to apply a route map to outgoing routes.

   ```
   device(config-bgp-ipv6u)# neighbor fe80:4398:ab30:45de::1 route-map out sendExtComRmap
   ```

10. Enter the **neighbor send-community** command and specify the **both** keyword to enable the sending of standard and extended attributes in updates to the specified BGP neighbor.

    ```
    device(config-bgp-ipv6u)# neighbor fe80:4398:ab30:45de::1 send-community both
    ```

The following example applies a BGP4+ extended community filter.

```
device# configure terminal
device(config)# ip extcommunity-list 1 permit rt 123:2
device(config)# router bgp
device(config-bgp)# local-as 1000
device(config-bgp)# neighbor fe80:4398:ab30:45de::1 remote-as 1001
device(config-bgp)# address-family ipv6 unicast
device(config-bgp-ipv6u)# neighbor fe80:4398:ab30:45de::1 activate
device(config-bgp-ipv6u)# neighbor fe80:4398:ab30:45de::1 route-map in extComRmap
device(config-bgp-ipv6u)# neighbor fe80:4398:ab30:45de::1 route-map out sendExtComRmap
device(config-bgp-ipv6u)# neighbor fe80:4398:ab30:45de::1 send-community both
```

# BGP4+ graceful restart

BGP4+ graceful restart (GR) allows for restarts where BGP neighboring devices participate in the restart, helping to ensure that no route and topology changes occur in the network for the duration of the restart.

The GR feature provides a routing device with the capability to inform its neighbors when it is performing a restart.

When a BGP session is established, GR capability for BGP is negotiated by neighbors through the BGP OPEN message. If the neighbor also advertises support for GR, GR is activated for that neighbor session. If both peers do not exchange the GR capability, the session is not GR-capable. If the BGP session is lost, the BGP peer router, known as a GR helper, marks all routes associated with the device as “stale” but continues to forward packets to these routes for a set period of time. The restarting device also continues to forward packets for the duration of the graceful restart. When the graceful restart is complete, routes are obtained from the helper so that the device is able to quickly resume full operation.

When the GR feature is configured on a device, both helper router and restarting router functionalities are supported. It is not possible to disable helper functionality explicitly.

GR is disabled by default and can be enabled in both IPv4 and IPv6 address families. When the GR timer expires, the BGP RASlog message is triggered.

NOTE
GR is not supported in BGP IPv6 address-family multicast configuration mode.

NOTE
BGP4+ GR can be configured for a global routing instance or for a specified VRF instance.

NOTE
BGP4+ GR is fully supported by MLX Series and XMR Series devices. The CER 2000 Series and CES 2000 Series devices only support GR helper mode.

# Configuring BGP4+ graceful restart

The graceful restart (GR) feature can be configured on a routing device, providing it with the capability to inform its neighbors when it is performing a restart.

NOTE
High availability (HA) requires GR to be enabled.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **router bgp** command to enable BGP routing.

```
device(config)# router bgp
```

3. Enter the **local-as** command to configure the autonomous system number (ASN) in which your device resides.

```
device(config-bgp)# local-as 1000
```

4. Enter the **neighbor remote-as** command, specifying an IPv6 address, to specify the autonomous system ASN in which the remote neighbor resides.

```
device(config-bgp)# neighbor 1000::1 remote-as 2
```

5. Enter the **address-family ipv6 unicast** command to enter IPv6 address family configuration mode.

```
device(config-bgp)# address-family ipv6 unicast
```

6. Enter the **neighbor activate** command to add a neighbor.

```
device(config-bgp-ipv6u)# neighbor 1000::1 activate
```

7. Enter the **graceful-restart** command to enable the graceful restart feature.

```
device(config-bgp-ipv6u)# graceful-restart
```

8. Do any of the following:
   - Enter the **graceful-restart** command using the **purge-time** keyword to overwrite the default purge-time value.

```
device(config-bgp-ipv6u)# graceful-restart purge-time 300
```

   - Enter the **graceful-restart** command using the **restart-time** keyword to overwrite the default restart-time advertised to graceful restart-capable neighbors.

```
device(config-bgp-ipv6u)# graceful-restart restart-time 180
```

   - Enter the **graceful-restart** command using the **stale-routes-time** keyword to overwrite the default amount of time that a helper device will wait for an EOR message from a peer.

```
device(config-bgp-ipv6u)# graceful-restart stale-routes-time 100
```

The following example enables the graceful restart feature.

```
device# configure terminal
device(config)# router bgp
device(config-bgp)# local-as 1
device(config-bgp)# neighbor 1000::1 remote-as 2
device(config-bgp)# address-family ipv6 unicast
device(config-bgp-ipv6u)# neighbor 1000::1 activate
device(config-bgp-ipv6u)# graceful-restart
```

The following example enables the graceful restart feature and sets the purge time to 300 seconds, overwriting the default value.

```
device# configure terminal
device(config)# router bgp
device(config-bgp)# local-as 1
device(config-bgp)# neighbor 1000::1 remote-as 2
device(config-bgp)# address-family ipv6 unicast
device(config-bgp-ipv6u)# neighbor 1000::1 activate
device(config-bgp-ipv6u)# graceful-restart purge-time 300
```

The following example enables the graceful restart feature and sets the restart time to 180 seconds, overwriting the default value.

```
device# configure terminal
device(config)# router bgp
device(config-bgp)# local-as 1
device(config-bgp)# neighbor 1000::1 remote-as 2
device(config-bgp)# address-family ipv6 unicast
device(config-bgp-ipv6u)# neighbor 1000::1 activate
device(config-bgp-ipv6u)# graceful-restart restart-time 180
```

The following example enables the graceful restart feature and sets the stale-routes time to 100 seconds, overwriting the default value.

```
device# configure terminal
device(config)# router bgp
device(config-bgp)# local-as 1
device(config-bgp)# neighbor 1000::1 remote-as 2
device(config-bgp)# address-family ipv6 unicast
device(config-bgp-ipv6u)# neighbor 1000::1 activate
device(config-bgp-ipv6u)# graceful-restart stale-routes-time 100
```

Use the **clear ipv6 bgp neighbor** command with the **all** parameter for the changes to the graceful restart parameters to take effect immediately.

# Configuring BGP4+ graceful restart in a nondefault VRF

The BGP4+ Graceful Restart (GR) feature can be enabled in a nondefault VRF.

1. Enter the **configure terminal** command to access global configuration mode.

   ```
   device# configure terminal
   ```

2. Enter the **router bgp** command to enable BGP routing.

   ```
   device(config)# router bgp
   ```

3. Enter the **local-as** command to configure the autonomous system number (ASN) in which your device resides.

   ```
   device(config-bgp)# local-as 1000
   ```

4. Enter the **neighbor remote-as** command, specifying an IPv6 address, to specify the autonomous system ASN in which the remote neighbor resides.

   ```
   device(config-bgp)# neighbor 1000::1 remote-as 2
   ```

5. Enter the **address-family ipv6 unicast** command using the **vrf** keyword, and specify a VRF name to create a BGP IPv6 unicast instance for VRF red and enter BGP address family IPv6 unicast VRF configuration mode.

```
device(config-bgp)# address-family ipv6 unicast vrf red
```

6. Enter the **graceful-restart** command to enable GR.

```
device(config-bgp-ipv6u-vrf)# graceful-restart
```

7. Do any of the following:
   - Enter the **graceful-restart** command using the **purge-time** keyword to overwrite the default purge-time value.

```
device(config-bgp-ipv6u-vrf)# graceful-restart purge-time 145
```

   - Enter the **graceful-restart** command using the **restart-time** keyword to overwrite the default restart-time advertised to graceful restart-capable neighbors.

```
device(config-bgp-ipv6u-vrf)# graceful-restart restart-time 245
```

   - Enter the **graceful-restart** command using the **stale-routes-time** keyword to overwrite the default amount of time that a helper device will wait for an EOR message from a peer.

```
device(config-bgp-ipv6u-vrf)# graceful-restart stale-routes-time 465
```

The following example enables the GR feature in a nondefault VRF instance.

```
device# configure terminal
device(config)# router bgp
device(config-bgp)# local-as 1000
device(config-bgp)# neighbor 1000::1 remote-as 2
device(config-bgp)# address-family ipv6 unicast vrf red
device(config-bgp-ipv6u-vrf)# graceful-restart
```

The following example enables the GR feature in a nondefault VRF instance and sets the purge time to 145 seconds, over-writing the default value.

```
device# configure terminal
device(config)# router bgp
device(config-bgp)# local-as 1000
device(config-bgp)# neighbor 1000::1 remote-as 2
device(config-bgp)# address-family ipv6 unicast vrf red
device(config-bgp-ipv6u-vrf)# graceful-restart purge-time 145
```

The following example enables the GR feature in a nondefault VRF instance, and sets the restart time to 245 seconds, over-writing the default value.

```
device# configure terminal
device(config)# router bgp
device(config-bgp)# local-as 1000
device(config-bgp)# neighbor 1000::1 remote-as 2
device(config-bgp)# address-family ipv6 unicast vrf red
device(config-bgp-ipv6u-vrf)# graceful-restart restart-time 245
```

The following example enables the GR feature in a nondefault VRF instance, and sets the stale-routes time to 100 seconds, over-writing the default value.

```
device# configure terminal
device(config)# router bgp
device(config-bgp)# local-as 1000
device(config-bgp)# neighbor 1000::1 remote-as 2
device(config-bgp)# address-family ipv6 unicast vrf red
device(config-bgp-ipv6u-vrf)# graceful-restart stale-routes-time 465
```

Use the **clear ipv6 bgp neighbor** command with the **all** parameter for the changes to the GR parameters to take effect immediately.

# BGP additional-paths overview

BGP additional-paths provides the ability for multiple paths for the same prefix to be advertised without the new paths implicitly replacing the previous paths. Path diversity is achieved rather than path hiding.

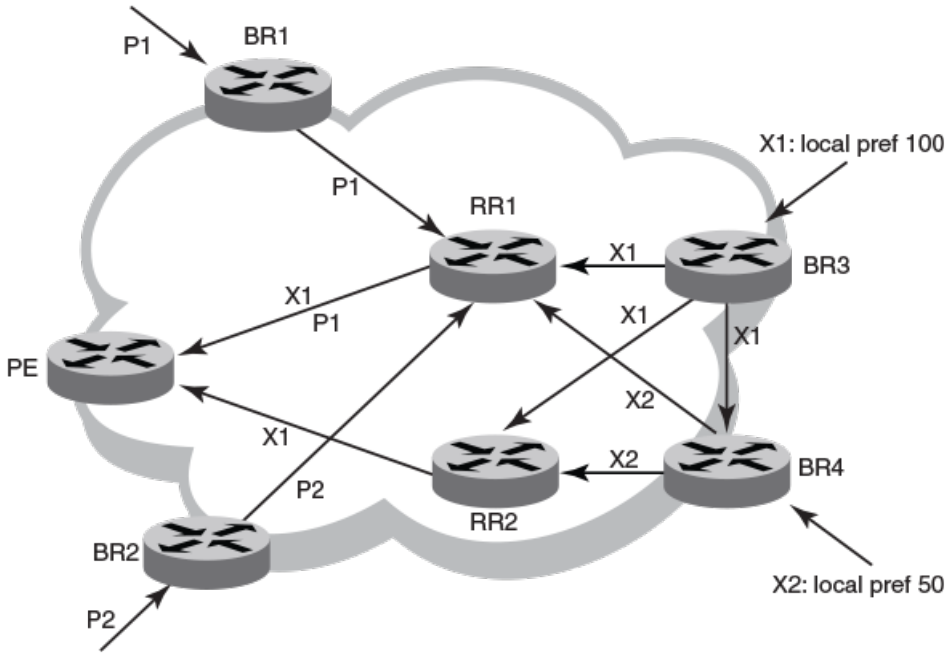
BGP devices generally advertise only their best path to neighboring devices, even when multiple paths exist. The advertisement of the same prefix from the same neighbor replaces the previous announcement of that prefix. This is known as an implicit withdraw, behavior that achieves better scaling but at the cost of path diversity.

Path hiding can affect the efficient use of BGP multipath and path diversity, and prevent hitless planned maintenance. Upon next hop failures, path hiding also inhibits fast and local recovery because the network must wait for BGP control plane convergence to restore traffic. BGP additional-paths enables BGP to advertise even the secondary best routes so that multiple paths for the same prefix can be advertised without the new paths implicitly replacing previous paths. BGP additional-paths provides a generic way of offering path diversity.

In the following figure, path hiding occurs in two ways:

- Prefix P has paths P1 and P2 advertised from BR1 and BR2 to RR1. RR1 selects P1 as the best path and advertises only P1 to PE.
- Prefix X has path X1 advertised from BR3 to BR4 with local preference 100. BR4 also has path X2. However, only the best path, X1, is selected. BR3 advertises X1 to the RRs and X2 is suppressed.

**FIGURE 34** BGP path hiding



## Advantages of BGP additional-paths

- Fast convergence and fault tolerance: When BGP additional-paths is enabled, more than one path to a destination is advertised. If one of the paths goes down, connectivity is easily restored due to the availability of backup paths. If the next hop for the prefix becomes unreachable, the device can switch to the backup route immediately without having to wait for BGP control plane messages.
- Enhanced load balancing capabilities: Traditionally with RRs in an iBGP domain, only the best path is given to the clients, even if ECMP paths exist. This affects load balancing. With additional paths advertised by RRs, the clients have more effective load balancing.

## Considerations and limitations for BGP additional-paths RIB-in

- When BGP additional-paths is not configured, only one NLRI per prefix per peer is supported. Any additional NLRI update for the same prefix from the same peer replaces the existing one.
- When BGP additional-paths is configured, a device can receive multiple NLRI advertisements for the same prefix from the same peer that are uniquely identified by NLRI path identifiers.
- For MLX Series and XMR Series devices, the maximum number of additional paths per peer per prefix is 128.
- For CER 2000 Series and CES 2000 Series devices, the maximum number of additional paths per peer per prefix is 64.
- Changes in the capability of sending or receiving additional paths are reflected only after the BGP session is restarted.

## Considerations and limitations for BGP additional-paths RIB-out

- Changes in the capability of sending or receiving additional paths are reflected only after the BGP session is restarted.
- The maximum number of paths that can be advertised per prefix is 16. If there are more than 16 paths for a prefix in the RIB-in, only 16 can be advertised.
- You should maintain the number of RIB-in paths for any prefix in the range of 16 for smooth RIB-out processing. Otherwise the RIB-out processing time increases exponentially in the scaled scenarios.

## Upgrade and downgrade considerations

If BGP additional-paths is enabled and the configuration saved, an error message occurs if the software is downgraded to an earlier version. BGP additional-paths should be unconfigured before a downgrade take place.

## BGP additional-paths functionality

BGP additional-paths is implemented by including an additional four-octet value known as a path identifier (ID) for each path in the NLRI. Path IDs are unique to a peering session and are generated for each network. A generated Path ID is unique per peer per prefix. A BGP device can receive the same path ID for the same prefix from two different peers, or it can receive the same path ID from the same peer for two different prefixes.

A path ID can apply to the IPv4 or IPv6 unicast or multicast address family.

Therefore, when the same prefix is received with the same path ID from the same peer, it is considered as a replacement route or a duplicate route. When the same prefix is received with a different path ID from the same peer, it is considered as an additional path to the prefix.

To send or receive additional paths, the additional-paths capability must be negotiated. If it is not negotiated, only the best path can be sent. BGP updates carry the path ID once the additional-paths capability is negotiated. In order to carry the path ID in an update message, the existing NLRI encodings are extended by prepending the path ID field, which consists of four octets.

The assignment of the path ID for a path by a BGP device occurs in such a way that the BGP device is able to use the prefix and path ID to uniquely identify a path advertised to a neighbor so as to continue to send further updates for that path. The receiving BGP neighbor that re-advertises a route regenerates its own path ID to be associated with the re-advertised route.

The set of additional paths advertised to each neighbor can be different, and advertisement filters are provided on a per-neighbor basis.

NOTE
BGP additional-paths is supported for the BGP IPv4 and IPv6 unicast address families and the BGP IPv4 and IPv6 multicast address families.

NOTE
BGP additional-paths is not supported for the BGP L2VPN VPLS, BGP VPNv4 unicast, and BGP VPNv6 address families.

There are three basic steps involved in configuring BGP additional-path:

- **Capability Negotiation**: Specify whether the device can send, receive, or send and receive additional paths. This is done at the address family level or peer-group level or the neighbor level. Refer to the sections and the NetIron Command Reference for more information.
- **Select Candidate paths**: Select a set or sets of candidate paths for advertisement by specifying selection criteria. This is done at the address family level.
- **Advertise additional paths from the candidate set**: Advertise to a neighbor a set or sets of additional paths from the candidate paths marked. This is done at the neighbor level or peer-group level.

# Configuring BGP4+ additional-paths and additional-path selection for the default VRF

You can enable BGP4+ additional-paths send and receive capability under the configured IPv6 address family. You can also select a set or sets of candidate paths for advertisement by specifying the selection criteria. This task specifies that the best eight BGP4+ paths are eligible to be selected as additional paths under the IPv6 multicast address family for the default VRF and enables BGP additional-paths.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **router bgp** command to enable BGP routing.

```
device(config)# router bgp
```

3. Enter the **local-as** command to configure the autonomous system number (ASN) in which your device resides.

```
device(config-bgp)# local-as 65520
```

4. Enter the **address-family ipv6 multicast** command to enter BGP address-family IPv6 multicast configuration mode.

```
device(config-bgp)# address-family ipv6 multicast
```

5. Enter the **additional-paths** command, using the **receive** parameter, to enable additional-paths receive capability under the IPv6 multicast address family.

```
device(config-bgp-ipv6m)# additional-paths receive
```

6. Enter the **additional-paths** command, using the **send** parameter, to enable additional-paths send capability under the IPv6 multicast address family.

```
device(config-bgp-ipv6m)# additional-paths send
```

7. Enter the **additional-paths select** command, using the **best** *number* parameter and entering a value of 8, to specify that the eight best BGP paths are eligible to be selected as additional paths under the IPv6 multicast address family.

```
device(config-bgp-ipv6m)# additional-paths select best 8
```

8. Enter the **neighbor additional-paths advertise** command, specifying an IPv6 address and using the **best** *value* parameter, to configure BGP to advertise the specified number of BGP best additional paths to a neighbor.

```
device(config-bgp-ipv6m)# neighbor 2001:2018:8192::124 additional-paths advertise best 8
```

The following example enables BGP4+ additional-paths send and receive capability and specifies that the best eight BGP paths are eligible to be selected as additional paths under the IPv6 multicast address family. The additional-paths feature is enabled and the best eight additional paths can be advertised to a BGP neighbor.

```
device# configure terminal
device(config)# router bgp
device(config-bgp)# local-as 65520
device(config-bgp)# address-family ipv6 unicast
device(config-bgp-ipv6m)# additional-paths receive
device(config-bgp-ipv6m)# additional-paths send
device(config-bgp-ipv6m)# additional-paths select best 8
device(config-bgp-ipv6m)# neighbor 2001:2018:8192::124 additional-paths advertise best 8
```

## Configuring BGP4+ additional-paths and additional-path selection for a non-default VRF instance

You can enable the advertisement of additional paths for all BGP neighbors under the configured IPv6 address family for a non-default VRF instance. You can also select a set or sets of candidate paths for advertisement by specifying the selection criteria. This task specifies that all BGP paths are eligible to be selected as additional paths under the configured IPv6 address family for VRF green.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **router bgp** command to enable BGP routing.

```
device(config)# router bgp
```

3. Enter the **local-as** command to configure the autonomous system number (ASN) in which your device resides.

```
device(config-bgp)# local-as 65520
```

4. Enter the **address-family ipv6 unicast** command, using the **vrf** parameter and specifying a VRF, to enter BGP address-family IPv6 unicast VRFconfiguration mode.

```
device(config-bgp)# address-family ipv6 unicast vrf green
```

5. Enter the **additional-paths** command, using the **receive** parameter, to enable additional-paths receive capability under the configured IPv6 address family for VRF instance green.

```
device(config-bgp-ipv6u-vrf)# additional-paths receive
```

6. Enter the **additional-paths** command, using the **send** parameter, to enable additional-paths send capability under the configured IPv6 address family for VRF instance green.

```
device(config-bgp-ipv6u-vrf)# additional-paths send
```

7. Enter the **additional-paths select** command, using the **all** parameter, to specify that all BGP paths are eligible to be selected as additional paths under the IPv6 unicast address family for VRF instance green.

```
device(config-bgp-ipv6u-vrf)# additional-paths select all
```

8. Enter the **neighbor additional-paths advertise** command, specifying an IPv6 address and using the **all** parameter, to configure BGP to advertise all BGP additional paths to a neighbor for VRF instance green.

```
device(config-bgp-ipv6u-vrf)# neighbor 2001:2018:8192::126 additional-paths advertise all
```

The following example enables BGP4+ additional-paths send and receive capability and specifies that all BGP paths are eligible to be selected as additional paths under the IPv6 unicast address family for VRF instance green. The add paths feature is enabled and all paths can be advertised to a BGP neighbor.

```
device# configure terminal
device(config)# router bgp
device(config-bgp)# local-as 65520
device(config-bgp)# address-family ipv6 unicast vrf green
device(config-bgp-ipv6u-vrf)# additional-paths receive
device(config-bgp-ipv6u-vrf)# additional-paths send
device(config-bgp-ipv6u-vrf)# additional-paths select all
device(config-bgp-ipv6u-vrf)# neighbor 2001:2018:8192::126 additional-paths advertise all
```

# Configuring BGP4+ additional-paths for a specified neighbor

You can apply filters for the advertisement of additional paths for specified BGP neighbors. .

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **router bgp** command to enable BGP routing.

```
device(config)# router bgp
```

3. Enter the **local-as** command to configure the autonomous system number (ASN) in which your device resides.

```
device(config-bgp)# local-as 65520
```

4. Enter the **address-family ipv6 unicast** command to enter BGP address-family IPv6 unicast configuration mode.

```
device(config-bgp)# address-family ipv6 unicast
```

5. Enter the **neighbor additional-paths** command, specifying an IPv6 address and using the **receive** parameter, to enable additional-paths receive capability from a specified BGP neighbor.

```
device(config-bgp-ipv6u)# neighbor 2001:2018:8192::125 additional-paths receive
```

6. Enter the **neighbor additional-paths** command, specifying an IPv6 address and using the **send** parameter, to enable additional-paths send capability to a specified BGP neighbor.

```
device(config-bgp-ipv6u)# neighbor 2001:2018:8192::125 additional-paths send
```

7. Enter the **additional-paths select** command, using the **best** parameter and entering a value, to specify the number of best BGP paths that are eligible to be selected as additional paths under the configured IPv6 address family .

```
device(config-bgp-ipv6u)# additional-paths select best 7
```

8. Enter the **neighbor additional-paths advertise** command, specifying an IPv6 address and using the **best** parameter. Then enter a value of seven to specify that the seven best BGP paths are eligible to be selected as additional paths.

```
device(config-bgp-ipv6u)# neighbor 2001:2018:8192::125 additional-paths advertise best 7
```

The following example enables the capability to send and receive additional paths for BGP4+ and specifies that the seven best BGP paths are eligible to be selected as additional paths. The BGP additional-paths feature is enabled and the seven best paths can be advertised to the neighbor.

```
device# configure terminal
device(config)# router bgp
device(config-bgp)# local-as 65520
device(config-bgp)# address-family ipv6 unicast
device(config-bgp-ipv6u)# neighbor 2001:2018:8192::125 additional-paths receive
device(config-bgp-ipv6u)# neighbor 2001:2018:8192::125 additional-paths send
device(config-bgp-ipv6u)# additional-paths select best 7
device(config-bgp-ipv6u)# neighbor 2001:2018:8192::125 additional-paths advertise best 7
```

# Configuring BGP additional-paths for a specified BGP4+ neighbor for a non-default VRF instance

You can enable the advertisement of additional paths for specified BGP4+ neighbors and apply filters for the advertisement of additional paths for BGP neighbors for a non-default VRF instance.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **router bgp** command to enable BGP routing.

```
device(config)# router bgp
```

3. Enter the **local-as** command to configure the autonomous system number (ASN) in which your device resides.

```
device(config-bgp)# local-as 65520
```

4. Enter the **address-family ipv6 unicast** command, using the **vrf** parameter and specifying a VRF, to enter BGP address-family IPv6 unicast VRF configuration mode.

```
device(config-bgp)# address-family ipv6 unicast vrf green
```

5. Enter the **neighbor additional-paths** command, specifying an IPv6 address and using the **receive** parameter, to enable additional-paths receive capability from a specified BGP neighbor for VRF green.

```
device(config-bgp-ipv6u-vrf)# neighbor 2001:2018:8192::121 additional-paths receive
```

6. Enter the **neighbor additional-paths** command, specifying an IPv6 address and using the **send** parameter, to enable additional-paths send capability to a specified BGP neighbor for VRF green.

```
device(config-bgp-ipv6u-vrf)# neighbor 2001:2018:8192::121 additional-paths send
```

7. Enter the **additional-paths select** command, using the **all** parameter, to specify that all BGP paths are eligible to be selected as additional paths under the IPv6 address unicast family for VRF green.

```
device(config-bgp-ipv6u-vrf)# additional-paths select all
```

8. Enter the **neighbor additional-paths advertise** command, specifying an IPv6 address and using the **all** parameter, to configure BGP to advertise all BGP additional paths for VRF green.

```
device(config-bgp-ipv6u-vrf)# neighbor 2001:2018:8192::121 additional-paths advertise all
```

The following example enables BGP4+ additional-paths send and receive capability and specifies that all BGP paths are eligible to be selected as additional paths under the IPv6 unicast address family for VRF instance green. The additional-paths feature is enabled and all paths can be advertised to a BGP neighbor.

```
device# configure terminal
device(config)# router bgp
device(config-bgp)# local-as 65520
device(config-bgp)# address-family ipv6 unicast vrf green
device(config-bgp-ipv6u-vrf)# neighbor 2001:2018:8192::121 additional-paths receive
device(config-bgp-ipv6u-vrf)# neighbor 2001:2018:8192::121 additional-paths send
device(config-bgp-ipv6u-vrf)# additional-paths select all
device(config-bgp-ipv6u-vrf)# neighbor 2001:2018:8192::121 additional-paths advertise all
```

# Disabling BGP additional-paths for a specified BGP4+ neighbor

By default the BGP additional-paths capability is disabled for BGP neighbors. You can disable BGP additional-paths capability for a specified BGP4+ neighbor if BGP additional-paths is enabled at the peer-group or address family level.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **router bgp** command to enable BGP routing.

```
device(config)# router bgp
```

3. Enter the **local-as** command to configure the autonomous system number (ASN) in which your device resides.

```
device(config-bgp)# local-as 65520
```

4. Enter the **address-family ipv6 unicast** command to enter BGP address-family IPv6 unicast configuration mode.

```
device(config-bgp)# address-family ipv6 unicast
```

5. Enter the **neighbor additional-paths disable** command, specifying an IPv6 address, to disable the sending of additional paths by BGP4+ to the specified BGP neighbor.

```
device(config-bgp-ipv6u)# neighbor 2001:2018:8192::125 additional-paths disable
```

The following example disables the sending of additional paths by BGP4+ to the specified neighbor in address-family IPv6 unicast configuration mode.

```
device# configure terminal
device(config)# router bgp
device(config-bgp)# local-as 65520
device(config-bgp)# address-family ipv6 unicast
device(config-bgp-ipv6u)# neighbor 2001:2018:8192::125 additional-paths disable
```

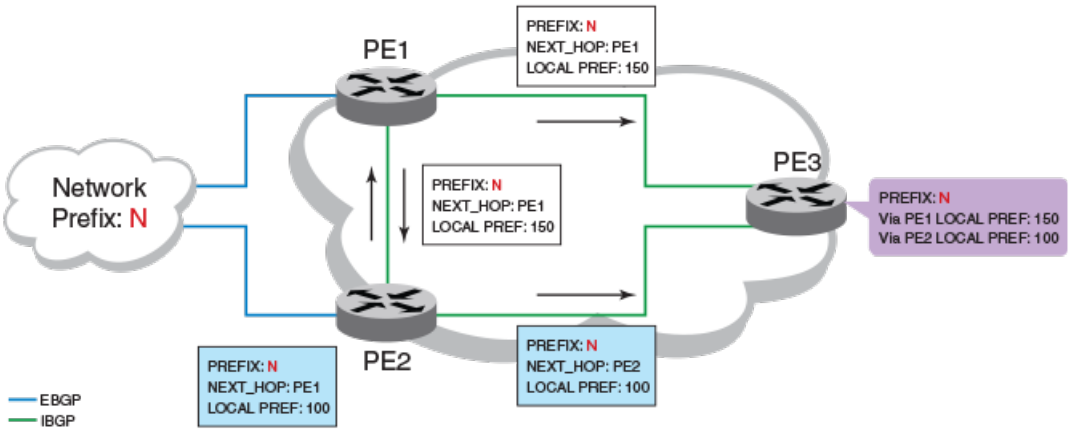
# BGP best external overview

BGP best external enables a device to advertise the most preferred route among those received from external neighbors as a backup route.

In active-backup topologies, service providers use routing policies that cause a border router to choose a path received over an Interior Border Gateway Protocol (iBGP) session as the best path for a prefix. This path is chosen even if an Exterior Border Gateway Protocol (eBGP) learned path exists. BGP best external is beneficial in such a topology. In such a topology, one exit or egress point for the prefix in the autonomous system is defined, and the other points are used as backups if the primary link or eBGP peering is unavailable. The border router does not advertise any path for such prefixes, and the paths learned over its eBGP sessions from the autonomous system (AS) are hidden. To cope with this situation, a device can advertise the best external path.

In the following figure, PE1 is the primary path to network N, and PE2 is the backup path. If BGP best external is not configured, PE2 does not advertise prefix N to its iBGP peers because PE2 prefers the iBGP route from PE1 as the best route compared to its best eBGP route. However, if BGP best external is configured, PE2 propagates its best external path to its iBGP peers so that PE3 has two paths for prefix N.

**FIGURE 35** BGP best external




NOTE
BGP best external is supported for the BGP IPv4 and IPv6 unicast address families and the BGP IPv4 and IPv6 multicast address families.

NOTE
BGP best external is not supported for the BGP L2VPN VPLS, BGP VPNv4 unicast, and BGP VPNv6 address families.

## Limitations of BGP best external

- BGP best external advertises the best external path to iBGP peers only.
- When BGP best external is configured on a route reflector (RR), the best external path is not advertised.

## Upgrade and downgrade considerations

If BGP best external is enabled and the configuration saved, an error message occurs if the software is downgraded to an earlier version. BGP best external should be unconfigured before a downgrade takes place.

## Configuring BGP4+ best external

You can enable BGP4+ to calculate the best external path and to advertise this path to its neighbors.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **router bgp** command to enable BGP routing.

```
device(config)# router bgp
```

3. Enter the **local-as** command to configure the autonomous system number (ASN) in which your device resides.

```
device(config-bgp)# local-as 65520
```

4. Enter the **address-family ipv6 multicast** command to enter BGP address-family IPv6 multicast configuration mode.

```
device(config-bgp)# address-family ipv6 multicast
```

5. Enter the **advertise-best-external** command to configure BGP4+ to calculate the best external path and to advertise this path to its neighbors.

```
device(config-bgp-ipv6m)# advertise-best-external
```

The following example configures BGP4+ to calculate the best external path and to advertise this path to its neighbors under the IPv6 multicast address family .

```
device# configure terminal
device(config)# router bgp
device(config-bgp)# local-as 65520
device(config-bgp)# address-family ipv6 multicast
device(config-bgp-ipv6m)# advertise-best-external
```

# Auto shutdown of BGP neighbors on initial configuration

The auto shutdown of BGP neighbors on initial configuration feature prevents a BGP peer from attempting to establish connections with remote peers when the BGP peer is first configured. Sessions are only initiated when the entire configuration for the BGP peer is complete.

When the auto shutdown of BGP neighbors on initial configuration feature is enabled, the value of the shutdown parameter for any existing configured neighbor is not changed. Any new BGP neighbor configured after the feature is enabled has the shutdown state set to the configured value. When the feature is enabled and a new BGP neighbor is configured, the shutdown parameter of the BGP neighbor is automatically set to enabled. When all the BGP neighbor parameters are configured and it is ready to start the establishment of BGP session with the remote peer, the shutdown parameter of the BGP neighbor is then disabled.

Any new neighbor configured and added to a peer group has the shutdown flag enabled by default. Additionally, any new neighbor configured with the autonomous system (AS) in which that remote neighbor resides specified using the **neighbor remote-as** command has the shutdown flag enabled by default.

# Configuring auto shutdown of BGP neighbors on initial configuration

Follow this procedure to enable auto shutdown of BGP neighbors on initial configuration.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **router bgp** command to enable BGP routing.

```
device(config)# router bgp
```

3. Enter the **local-as** command to configure the autonomous system number (ASN) in which your device resides.

```
device(config-bgp)# local-as 65520
```

4. Enter the **auto-shutdown-new-neighbors** command to prevent the BGP peer from attempting to establish connections with remote peers until all BGP neighbor parameters are configured.

```
device(config-bgp)# auto-shutdown-new-neighbors
```

The following example enables auto shutdown of BGP neighbors on initial configuration.

```
device# configure terminal
device(config)# router bgp
device(config-bgp)# local-as 65520
device(config-bgp)# auto-shutdown-new-neighbors
```

# Disabling the BGP4+ peer shutdown state

When the auto shutdown of BGP4+ neighbors on initial configuration feature is enabled, a BGP4+ peer is prevented from attempting to establish connections with remote peers when the BGP4+ peer is first configured. Once all of the configuration parameters for the peer are complete, you can start the BGP4+ session establishment process and disable the peer shutdown state. Follow this procedure to disable the peer shutdown state.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **router bgp** command to enable BGP routing.

```
device(config)# router bgp
```

3. Enter the **local-as** command to configure the autonomous system number (ASN) in which your device resides.

```
device(config-bgp)# local-as 65520
```

4. Enter the **no neighbor shutdown** command, specifying an IPv6 address, to disable the peer shutdown state and begin the BGP4+ session establishment process.

```
device(config-bgp)# no neighbor 2001:2018:8192::125 shutdown
```

The following example disables the peer shutdown state and begins the BGP4+ session establishment process.

```
device# configure terminal
device(config)# router bgp
device(config-bgp)# local-as 65520
device(config-bgp)# no neighbor 2001:2018:8192::125 shutdown
```

# Generalized TTL Security Mechanism support

Generalized TTL Security Mechanism (GTSM) is a lightweight security mechanism that protects external Border Gateway Protocol (eBGP) peering sessions from CPU utilization-based attacks using forged IP packets. GTSM prevents attempts to hijack the eBGP peering session by a host on a network segment that is not part of either BGP network, or by a host on a network segment that is not between the eBGP peers.

GTSM is enabled by configuring a minimum Time To Live (TTL) value for incoming IP packets received from a specific eBGP peer. BGP establishes and maintains the session only if the TTL value in the IP packet header is equal to or greater than the TTL value configured for the peering session. If the value is less than the configured value, the packet is silently discarded and no Internet Control Message Protocol (ICMP) message is generated.

When GTSM protection is enabled, BGP control packets sent by the device to a neighbor have a Time To Live (TTL) value of 255. In addition, the device expects the BGP control packets received from the neighbor to have a TTL value of either 254 or 255. For multihop peers, the device expects the TTL for BGP control packets received from the neighbor to be greater than or equal to 255, minus the configured number of hops to the neighbor. If the BGP control packets received from the neighbor do not have the anticipated value, the device drops them.

For more information on GTSM protection, refer to RFC 3682.

## Assumptions and limitations

- GTSM is supported for both directly connected peering sessions and multihop eBGP peering sessions.
- GTSM is supported for eBGP only.
- GTSM does not protect the integrity of data sent between eBGP peers and does not validate eBGP peers through any authentication method.
- GTSM validates only the locally configured TTL count against the TTL field in the IP packet header.
- GTSM should be configured on each participating device to maximize the effectiveness of this feature.
- When GTSM is enabled, the eBGP session is secured in the incoming direction only and has no effect on outgoing IP packets or the remote device.

## Configuring GTSM for BGP4+

Generalized TTL Security Mechanism (GTSM) can be configured to protect external Border Gateway Protocol (eBGP) peering sessions .

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **router bgp** command to enable BGP routing.

```
device(config)# router bgp
```

3. Enter the **local-as** command to configure the autonomous system number (ASN) in which your device resides.

```
device(config-bgp)# local-as 65520
```

4. Enter the **address-family ipv6 unicast** command to enter IPv6 address family unicast configuration mode.

```
device(config-bgp)# address-family ipv6 unicast
```

5. Enter the **neighbor remote-as** command, specifying an IPv6 address, to add a neighbor.

```
device(config-bgp-ipv6u)# neighbor 2001:2018:8192::125 remote-as 2
```

6. Enter the **neighbor ebgp-btsh** command, specifying an IPv6 address, to enable GTSM.

```
device(config-bgp-ipv6u)# neighbor 2001:2018:8192::125 ebgp-btsh
```

The following example enables GTSM between a device and a neighbor with the IPv6 address 2001:2018:8192::125.

```
device# configure terminal
device(config)# router bgp
device(config-bgp)# local-as 65520
device(config-bgp)# address-family ipv6 unicast
device(config-bgp-ipv6u)# neighbor 2001:2018:8192::125 remote-as 2
device(config-bgp-ipv6u)# neighbor 2001:2018:8192::125 ebgp-btsh
```

# Disabling the BGP AS_PATH check function

A device can be configured so that the AS_PATH check function for routes learned from a specific location is disabled, and routes that contain the recipient BGP speaker's AS number are not rejected.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **router bgp** command to enable BGP routing.

```
device(config)# router bgp
```

3. Enter the **address-family ipv6 unicast** command to enter IPv6 address family unicast configuration mode.

```
device(config-bgp)# address-family ipv6 unicast
```

4. Enter the **neighbor allowas-in** command with an IPv6 address and specify a **number** to disable the BGP AS_PATH check function, and specify the number of times that the AS path of a received route may contain the recipient BGP speaker's AS number and still be accepted.

```
device(config-bgp-ipv6u)# neighbor 2001:db8:e0ff:783a::4 allowas-in 3
```

The following example specifies that the AS path of a received route may contain the recipient BGP speaker's AS number three times and still be accepted.

```
device# configure terminal
device(config)# router bgp
device(config-bgp)# address-family ipv6 unicast
device(config-bgp-ipv6u)# neighbor 2001:db8:e0ff:783a::4 allowas-in 3
```

# Displaying BGP4+ statistics

Various **show ipv6 bgp** commands verify information about BGP4+ configurations.

Use one or more of the following commands to verify BGP4+ information. The commands do not have to be entered in this order.

1. Enter the **show ipv6 bgp summary** command.

```
device> show ipv6 bgp summary

 BGP4 Summary
  Router ID: 10.7.7.7   Local AS Number: 100
  Confederation Identifier: not configured
  Confederation Peers:
  Maximum Number of IP ECMP Paths Supported for Load Sharing: 1
  Number of Neighbors Configured: 1, UP: 0
  Number of Routes Installed: 0
  Number of Routes Advertising to All Neighbors: 0 (0 entries)
  Number of Attribute Entries Installed: 0
  '+': Data in InQueue '>': Data in OutQueue '-': Clearing
  '*': Update Policy 'c': Group change 'p': Group change Pending
  'r': Restarting 's': Stale '^': Up before Restart '<': EOR waiting
  Neighbor Address    AS#      State   Time        Rt:Accepted Filtered Sent  ToSend
  10:2::2             100      CONN    0h 9m 0s    0           0        0     0
```

   This example output gives summarized BGP4+ information.

2. Enter the **show ipv6 bgp attribute-entries** command.

```
device> show ipv6 bgp attribute-entries

Total number of BGP Attribute Entries: 378
1       Next Hop  :::                 MED :1              Origin:INCOMP
        Originator:0.0.0.0            Cluster List:None
        Aggregator:AS Number :0      Router-ID:0.0.0.0          Atomic:None
        Local Pref:100               Communities:Internet
        AS Path   :(65002) 65001 4355 2548 3561 5400 6669 5548
        Address: 0x27a4cdb0  Hash:877 (0x03000000) Reference Counts: 2:0:2
```

   This example shows information about two route-attribute entries that are stored in device memory.

3. Enter the **show ipv6 bgp peer-group** command.

```
device> show ipv6 bgp peer-group peer_group1

1   BGP peer-group is pg1, Remote AS: 65002
    Description: device group 1
       NextHopSelf: yes
      Address family : IPV4 Unicast
      Address family : IPV4 Multicast
      Address family : IPV6 Unicast
    Members:
       IP Address: 10.169.102.2
       IP Address: 10.169.100.2
       IP Address: 10.169.101.2
       IP Address: 10.169.103.2
       IP Address: 10.169.104.2
       IP Address: 10.169.105.2
       IP Address: 10.169.106.2
       IP Address: 10.169.107.2
       IP Address: 10.169.108.2
       IP Address: 10.169.109.2
       IP Address: 10.169.110.2
       IP Address: 10.169.111.2
       IP Address: 10.169.112.2
```

   This example shows output for a peer group called “peer_group1”.

4. Enter the **show ipv6 bgp routes** command.

```
device> show ipv6 bgp routes

Total number of BGP Routes: 4
Status A:AGGREGATE B:BEST b:NOT-INSTALLED-BEST C:CONFED_EBGP D:DAMPED
       E:EBGP H:HISTORY I:IBGP L:LOCAL M:MULTIPATH m:NOT-INSTALLED-MULTIPATH
       S:SUPPRESSED F:FILTERED s:STALE
       Prefix             Next Hop    MED    LocPrf   Weight Status
1          2001:db8:1::/64   2001:db8:1111::2     1      100      32768       BL

        AS_PATH:
2          2001:db8:2::/64   2001:db8::30.30.30.1    1      100      0           BI

        AS_PATH:
3          2001:db8:1111::/64 ::                   0      100      32768       BL

        AS_PATH:
4          2001:db8:2222::/64 2001:db8::30.30.30.1    0      100     0           BI

        AS_PATH:
```

This example shows general BGP4+ route information.

5. Enter the **show ipv6 bgp routes** command, using the **detail** keyword.

```
device> show ipv6 bgp routes detail

Total number of BGP Routes: 4
Status A:AGGREGATE B:BEST b:NOT-INSTALLED-BEST C:CONFED_EBGP D:DAMPED
       E:EBGP H:HISTORY I:IBGP L:LOCAL M:MULTIPATH m:NOT-INSTALLED-MULTIPATH
       S:SUPPRESSED F:FILTERED s:STALE
1 Prefix: 2001:db8:1::/64,  Status: BL,  Age: 0h1m14s
    NEXT_HOP: 2001:db8:1111::2, Learned from Peer: Local Router
    In-Label: 794624
    LOCAL_PREF: 100,  MED: 1,  ORIGIN: incomplete,  Weight: 32768
    AS_PATH:
    Adj_RIB_out count: 1,  Admin distance 1
2 Prefix: 2001:db8:2::/64,  Status: BI,  Age: 0h0m8s
    NEXT_HOP: 2001:db8::ffff:30:1, Metric: 1,  Learned from Peer: 10.30.30.1 (1)
    Out-Label: 794624
    LOCAL_PREF: 100,  MED: 1,  ORIGIN: incomplete,  Weight: 0
    AS_PATH:
3 Prefix: 2001:db8:1111::/64,  Status: BL,  Age: 0h2m26s
    NEXT_HOP: ::, Learned from Peer: Local Router
    In-Label: 794624
    LOCAL_PREF: 100,  MED: 0,  ORIGIN: incomplete,  Weight: 32768
    AS_PATH:
    Adj_RIB_out count: 1,  Admin distance 1
4 Prefix: 2001:db8:2222::/64,  Status: BI,  Age: 0h0m35s
    NEXT_HOP: 2001:db8::ffff:30:1, Metric: 1,  Learned from Peer: 10.30.30.1 (1)
    Out-Label: 794624
    LOCAL_PREF: 100,  MED: 0,  ORIGIN: incomplete,  Weight: 0
    AS_PATH:
```

This example shows detailed BGP4+ route information.

# Displaying BGP4+ neighbor statistics

Various **show ipv6 bgp neighbor** commands verify information about BGP4+ neighbor configurations.

Use one or more of the following commands to verify BGP4+ neighbor information. The commands do not have to be entered in this order.

1. Enter the **show ipv6 bgp neighbors** command.

```
device> show ipv6 bgp neighbors
Total number of BGP Neighbors: 1
    '+': Data in InQueue '>': Data in OutQueue '-': Clearing
    '*': Update Policy 'c': Group change 'p': Group change Pending
    'r': Restarting 's': Stale '^': Up before Restart '<': EOR waiting

1   IP Address: 78:2::2, AS: 100 (IBGP), RouterID: 0.0.0.0, VRF: default-vrf
    State: CONNECT, Time: 0h9m7s, KeepAliveTime: 60, HoldTime: 180
    Minimal Route Advertisement Interval: 0 seconds
    Messages:    Open    Update  KeepAlive Notification Refresh-Req
       Sent    : 0       0       0         0            0
       Received: 0       0       0         0            0
    Last Connection Reset Reason:Unknown
    Notification Sent:     Unspecified
    Notification Received: Unspecified
    Neighbor NLRI Negotiation:
      Peer configured for IPV6 unicast  Routes
    Neighbor ipv6 MPLS Label Capability Negotiation:
    Neighbor AS4 Capability Negotiation:
    Outbound Policy Group:
       ID: 2, Use Count: 2
    BFD:Disabled
Error: TCP status not available
…
```

   This example output gives summarized information about BGP4+ neighbors.

2. Enter the **show ipv6 bgp neighbors advertised-routes** command.

```
device> show ipv6 bgp neighbors 2001:db8::110 advertised-routes

Status A:AGGREGATE B:BEST b:NOT-INSTALLED-BEST E:EBGP I:IBGP L:LOCAL
       Prefix             Next Hop  MED LocPrf   Weight Status
1      2001:db8:1234::/48 ::        1            32768  BL
         AS_PATH:
2      2001:db8:2002::/48 ::        1            32768  BL
         AS_PATH:
```

   This example shows information about all the routes the BGP4+ networking device advertised to the neighbor.

3. Enter the **show ipv6 bgp neighbors received-routes** command.

```
device> show ipv6 bgp neighbors 2001:db8::10 received-routes

There are 4 received routes from neighbor 2001:db8::10
Searching for matching routes, use ^C to quit...
Status A:AGGREGATE B:BEST b:NOT-INSTALLED-BEST C:CONFED_EBGP D:DAMPED
E:EBGP H:HISTORY I:IBGP L:LOCAL M:MULTIPATH S:SUPPRESSED F:FILTERED
    Prefix    Next Hop    Metric    LocPrf    Weight    Status
1     2001:db8:2002::/64    2001:db8::10     0    100    0    BE
AS_PATH: 400
2    2001:db8:2003::/64    2001:db8::10     1    100    0    BE
AS_PATH: 400
3    2001:db8:2004::/64    2001:db8::10     1    100    0    BE
AS_PATH: 400
4    2001:db8:2005::/64    2001:db8::10     1    100    0    BE
AS_PATH: 400
```

This example lists all route information received in route updates from BGP4+ neighbors of the device since the soft-reconfiguration feature was enabled.

4. Enter the **show ipv6 bgp neighbors rib-out-routes** command.

```
device> show ipv6 bgp neighbors 2001:db8::110 rib-out-routes

       There are 2 RIB_out routes for neighbor 2001:db8::110
Status A:AGGREGATE B:BEST b:NOT-INSTALLED-BEST E:EBGP I:IBGP L:LOCAL
       Prefix             Next Hop     Metric    LocPrf     Weight Status
1      2001:db8:1234::/48  ::           1         100        32768  BL
         AS_PATH:
2      2001:db8:2002::/48  ::           1         100        32768  BL
         AS_PATH:
```

This example shows information about BGP4+ outbound RIB routes.

# DHCPv4



# DHCP snooping

NOTE
DHCP snooping supports only IPv4 traffic.

Dynamic Host Configuration Protocol (DHCP) snooping enables the device to filter untrusted DHCP packets in a subnet. DHCP snooping prevents MiM attacks, such as a malicious user posing as a DHCP server sending false DHCP server reply packets with the intention of misdirecting other users. DHCP snooping can also stop unauthorized DHCP servers and prevent errors due to user mis-configuration of DHCP servers.

NOTE
DHCP snooping does not dynamically build the ARP Inspection table.

## How DHCP snooping works

When enabled on a VLAN, DHCP snooping stands between untrusted ports (those connected to host ports) and trusted ports (those connected to DHCP servers). A VLAN with DHCP snooping enabled forwards DHCP request packets from clients and discards DHCP server reply packets on untrusted ports, and it forwards DHCP server reply packets on trusted ports to DHCP clients, as shown in the following figures.

**FIGURE 36** DHCP snooping at work - on untrusted port

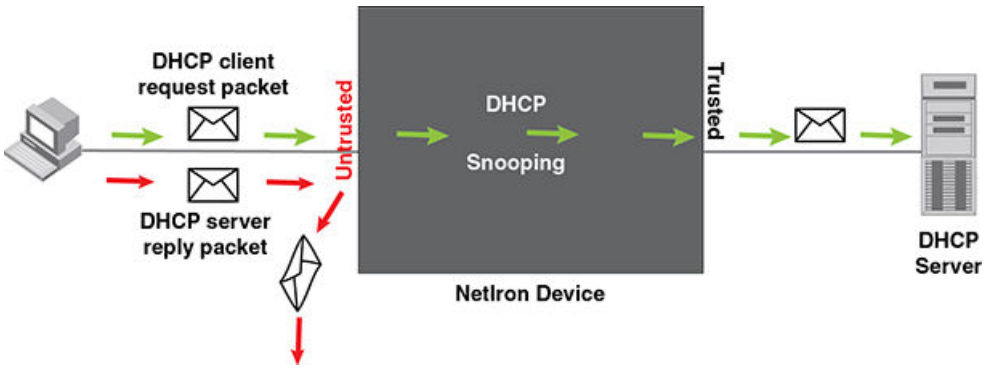
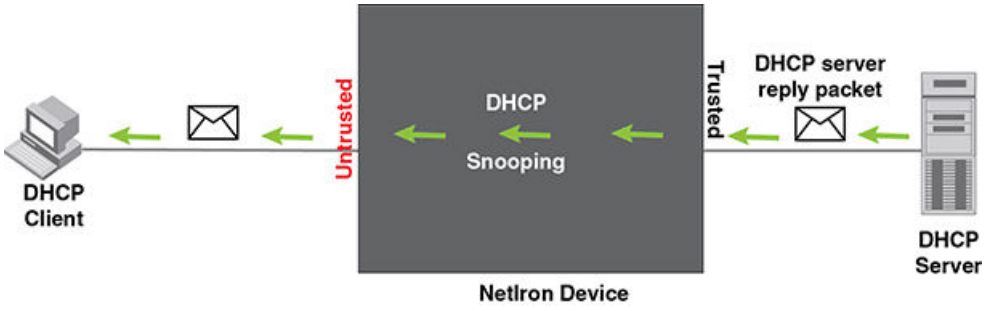


FIGURE 37 DHCP snooping at work - on trusted port




### *DHCP binding database*

On trusted ports, DHCP server reply packets are forwarded to DHCP clients. The DHCP server reply packets collect client IP to MAC address binding information, which is saved in the DHCP binding database. This information includes MAC address, IP address, lease time, VLAN number, and port number.

In the Extreme device configuration, the DHCP binding database is integrated with the enhanced ARP table, which is used by Dynamic ARP Inspection. For more information, refer to ARP entries on page 34.

The lease time will be refreshed when the client renews its IP address with the DHCP server; otherwise the device removes the entry when the lease time expires.

## System reboot and the binding database

To allow DAI and DHCP snooping to work smoothly across a system reboot, the binding database is saved to a file in the system flash memory after the user issues the "reload" command. DHCP learned entries are written to the system flash memory before the device reboots. The flash file is written and read only if DHCP snooping is enabled.

## Configuring DHCP snooping

Follow the steps listed below to configuring DHCP snooping.

1. Enable DHCP snooping on a VLAN.
2. For ports that are connected to a DHCP server, change their trust setting to trusted.

   The following table shows the default settings of DHCP snooping:

| Feature | Default |
|---|---|
| DHCP snooping | Disabled |
| Trust setting for ports | Untrusted |

### *Enabling DHCP snooping on a VLAN*

DHCP packets for a VLAN with DHCP snooping enabled are inspected.

DHCP snooping is disabled by default. This feature must be enabled on the client and the DHCP server VLANs. To enable DHCP snooping, enter the following global command for these VLANs.

```
device(config)#ip dhcp-snooping vlan 2
```

The command enables DHCP snooping for a VLAN or a range of VLANS.

**Syntax: [no] ip dhcp-snooping vlan vlan-number** [ **to vlan_number** ] [ **insert-relay-information** ]

The *vlan-number* variable specifies the ID of a configured client or DHCP server VLAN.

If the [insert-relay-information] option is enabled, then DHCP option 82 is inserted in all the DHCP request packets. Refer to DHCP binding database on page 382 for more information.

# DHCP snooping suboptions

When the DHCP relay agent information option is enabled, the DHCP relay adds the option 82 information to packets it receives from clients, then forwards the packets to the DHCP server. The DHCP server uses the option 82 information to decide which IP address to assign to the client or the DHCP server may use the information in the option 82 field for determining which services to grant to the client. The DHCP server sends its reply back to the DHCP relay, which removes the option 82 information field from the message, and then forwards the packet to the client.

Option 82 information is made up of a series of suboptions. This device supports suboption 1, suboption 2, and suboption 9.

- Agent Circuit ID (suboption 1) --An ASCII string identifying the interface on which a client DHCP packet is received.
- Agent Remote ID (suboption 2) --An ASCII string assigned by the relay agent that securely identifies the client.
- Vendor-Specific (suboption 9) --Contains the Internet Assigned Numbers Authority (IANA) enterprise number (4874) and the layer 2 circuit ID and the user packet class.

Suboption 1 and suboption 2 are usually determined by the client network access device and depend on the network configuration.

Suboption 9 can be used to associate specific data with the DHCP messages relayed between the DHCP relay and the DHCP server. The suboption 9 can include the client's IEEE 802.1p value, which identifies the client's user priority.

## *Enabling trust on a port*

The default trust setting for a port is untrusted. To enable trust on a port connected to a DHCP server, enter commands such as the following.

```
device(config)#interface ethernet 1/1
device(config-if-e10000-1/1)#dhcp-snooping-trust
```

Port 1/1 is connected to a DHCP server. The commands change the CLI to the interface configuration level of port 1/1 and set the trust setting of port 1/1 to trusted.

**Syntax: [no] dhcp-snooping-trust**

# Clearing the DHCP binding database

You can clear the DHCP binding database using the **clear dhcp-binding** command.

You can remove all entries in the database, or remove entries for a specific IP subnet, a VRF instance, or a VLAN id.

To remove all entries from the DHCP binding database, enter the following command.

```
device# clear dhcp-binding
```

For example, to clear entries for a specific IP subnet, enter a command such as the following.

```
device# clear dhcp 10.10.102.4
```

**Syntax: clear dhcp** [ **ip subnet** ] [ **vlan vlan_id** ] [ **vrf vrf_name** ]

The *vlan_id* variable specifies the ID of a configured VLAN.

The *vrf_name* variable specifies the VRF instance.

# DHCP option 82 insertion

DHCP option 82 insertion can be used to assist DHCP servers to implement dynamic address policy. When DHCP option 82 is present in DHCP packets, DHCP servers gets additional information about the clients' identity.

The Extreme device inserts DHCP option 82 when relaying DHCP request packets to DHCP servers. When DHCP server reply packets are forwarded back to DHCP clients, and sub-option 2 matches the local port MAC address, then DHCP option 82 is deleted. The vlan/port information is used to forward the DHCP reply. Refer to the following figures:

**FIGURE 38** DHCP option 82 is added to the packet

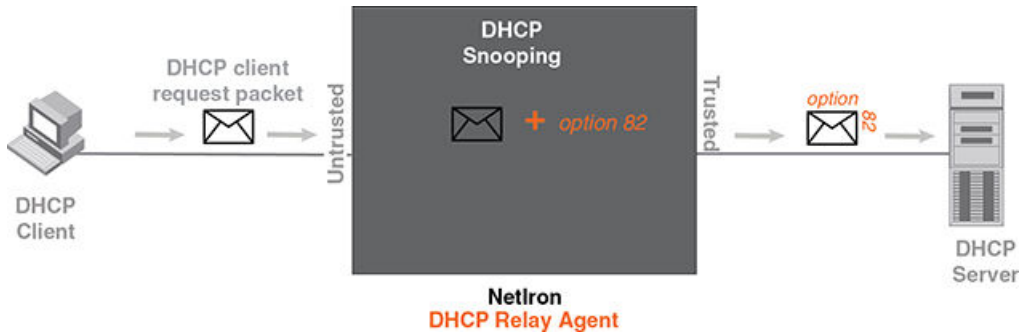



**FIGURE 39** DHCP option 82 is removed from the packet


The option 82 insertion/deletion feature is available only when DHCP snooping is enabled for the client/server ports, and when the device is configured as a DHCP relay agent. By default, DHCP option 82 is off.

DHCP option 82 contains two sub-options; sub-option 1 (circuit ID) and sub-option 2 (remote ID).

Sub-option 1, relay agent circuit ID is in the following format.

VLAN id (2 bytes) / module id (1 byte) / port id (1 byte) (The module and port id will be 1 based).

The circuit ID identifies the location of the port, showing where the DHCP request comes from.

Typical address allocation is based on the gateway address of the relay agent.

Sub-option 2, Remote ID is in the following format.

XMR Series base MAC address (6 bytes)

# Displaying DHCP snooping status and ports

To display the DHCP snooping status for a VLAN and the trusted and untrusted ports in the VLAN, enter the following command.

```
device#show ip dhcp-snooping vlan 172
IP DHCP snooping VLAN 172: Enabled
  Trusted Ports : ethe 5/2 ethe 5/4
  Untrusted Ports : ethe 4/24 ethe 9/4 to 9/5 ethe 9/12 ethe 9/14
```

**Syntax: show ip dhcp-snooping** [ **vlan vlan-id** ]

# Displaying DAI binding entries

To display all ARP inspection binding entries, including dhcp bindings specific to a VRF instance, enter the following command.

```
device(config)#show dai 10.1.1.0/24
Total no. of entries: 51
Idx Type IP Address      MAC Address       Port    Vlan Server IP    LTime
1   D    10.1.1.19       aabb.cc00.0012     10      10.1.1.2       3360
2   D    10.1.1.22       aabb.cc00.0007     10      10.1.1.2       3360
3   D    10.1.1.25       aabb.cc00.0030     10      10.1.1.2       3360
4   D    10.1.1.26       aabb.cc00.0004     10      10.1.1.2       3360
5   D    10.1.1.30       0030.488a.1c25     10      10.1.1.2       40200
6   D    10.1.1.32       aabb.cc00.0001     10      10.1.1.2       3360
7   D    10.1.1.34       aabb.cc00.0019     10      10.1.1.2       1560
8   D    10.1.1.39       aabb.cc00.000d     10      10.1.1.2       3360
9   D    10.1.1.44       aabb.cc00.0020     10      10.1.1.2       3360
10  D    10.1.1.46       aabb.cc00.0022     10      10.1.1.2       3360
```

**Syntax: show dai** [ **vrf vrf_name** ] [ **vlan vlan_id** ] [ **ip-subnet** ]

The *vrf_name* variable specifies the ARP entries that belong to a given VRF instance.

The *vlan_id* variable specifies the ID of a configured VLAN.

The *ip subnet* variable specifies the ARP entries that belong to specific IP-subnet address.

The following table describes the parameters of the **show dai** command:

**TABLE 37** Display of show dai

| This field... | Displays... |
|---|---|
| Index (Idx) | The row number of this entry in the IP route table. |
| Type | The ARP entry type, which can be any one of the following:<br>Dynamic - The Layer 3 Switch learned the entry from an incoming packet.<br>Static - The Layer 3 Switch loaded the entry from the static ARP table when the device for the entry was connected to the Layer 3 Switch.<br>DHCP - The Layer 3 Switch learned the entry from the DHCP binding address table. |
| IP Address | The IP address of the device. |
| MAC Address | The MAC address of the device. |
| Port | The source port for the host vlan. |
| Vlan Server IP | The VLAN Server IP address of the server which assigns the IP/MAC mapping. |
| LTime | The lease time (aging timer for a DHCP entry). |

# Displaying DHCP snooping statistics counters

NOTE

The last five dropped packets are displayed through the CLI. Notifications and traps are not sent.

To display the DHCP snooping statistics counters, enter the following command.

```
device#show ip dhcp-snooping-statistic slot 1
Module 1:
Port     DHCP Packets Captured      DHCP Packets dropped
1/1         0                        0
1/2         0                        0
1/3         0                        0
1/4         0                        0
1/5         0                        0
1/6         0                        0
1/7         0                        0
1/8         0                        0
1/9         0                        0
1/10        0                        0
1/11        0                        0
1/12        0                        0
1/13        0                        0
1/14        0                        0
1/15        0                        0
1/16        0                        0
1/17        0                        0
1/18        0                        0
1/19        9                        0
1/20        8                        6
```

The following table describes the output of the show ip dhcp snooping statistic.

**TABLE 38** Output from the show ip dhcp snooping statistic slot

| This field... | Displays... |
|---|---|
| Module | Module number as positioned in the chassis. |
| Port | Port number in specified module. |
| DHCP Packets Captured | The number of DHCP packets captured on the port. |
| DHCP Packets Dropped | The number of DHCP packets dropped by DHCP snooping. |

To display the DHCP snooping statistics counters for Ethernet ports, enter the following command.

```
device#show ip dhcp-snooping-statistic eth 1/20
DHCP packets captured: 9
DHCP packets dropped by snooping: 7
Last 5 packets dropped by snooping:
Time                 DHCP type Source Mac/     Server IP/      Vlan
                               Source IP       Gateway IP
2008-05-03  00:29:43 OFFER     0030.4843.37ad  10.1.1.125     11
                               10.1.1.125     10.1.1.10
2008-05-03  00:29:59 OFFER     0030.4843.37ad  10.1.1.125     11
                               10.1.1.125     10.1.1.10
2008-05-03  00:31:18 OFFER     0030.4843.37ad  10.1.1.125     11
                               10.1.1.125     10.1.1.10
2008-05-03  00:31:22 OFFER     0030.4843.37ad  10.1.1.125     11
                               10.1.1.125     10.1.1.10
2008-05-03  00:31:30 OFFER     0030.4843.37ad  10.1.1.125     11
                               10.1.1.125     10.1.1.10
```

**Syntax: show ip dhcp-snooping-statistic** [ **slot slot** ] | [ **ethernet slot/port** ]

NOTE
If an Ethernet port is provided, the last five dropped packets are displayed

1. DHCP packets captured
2. The number of DHCP packets captured on port 20.
3. DHCP packets dropped by snooping
4. The number of DHCP packets dropped by DHCP snooping.
5. Last 5 packets dropped by snooping
6. The last 5 DHCP packets dropped per port.
7. Time
8. The time tracking system for collecting statistically information. Date and time are displayed.
9. DHCP Type
10. The DHCP Type displays the following: OFFER - When the server responds with a proposal of parameters. ACK- When the server assign an IP address. NAK- When the server rejects the request from the client.
11. Source MAC or Source IP
12. The Source MAC or Source IP address
13. Server IP or Gateway IP
14. The Serve IP or Gateway IP
15. Vlan
16. The VLAN number that DHCP Snooping was rejected on.

# Clearing DHCP snooping counters

To clear the DHCP snooping statistic counters for a specific slot, enter the following command.

```
device#clear dhcp-snooping-statistics slot 1
```

To clear the DHCP snooping statistic counters for a specific ethernet port, enter the following command.

```
device#clear dhcp-snooping-statistics ethernet 1/20
```

**Syntax: clear dhcp-snooping-statistics** [ **slot slot** ] | [ **ethernet slot/port** ]

# DHCP snooping configuration example

The following example configures VLAN 2 and VLAN 20, and changes the CLI to the global configuration level to enable DHCP snooping on the two VLANs. The commands are as follows.

```
device(config)#vlan 2
device(config-vlan-2)#untagged ethe 1/3 to 1/4
device(config-vlan-2)#router-interface ve 2
device(config-vlan-2)#exit
device(config)# ip dhcp-snooping vlan 2
device(config)#vlan 20
device(config-vlan-20)#untagged ethe 1/1 to 1/2
device(config-vlan-20)#router-interface ve 20
```

```
device(config-vlan-20)#exit
device(config)#ip dhcp-snooping vlan 20
```

On VLAN 2, client ports 1/3 and 1/4 are untrusted by default: all client ports are untrusted. Hence, only DHCP client request packets received on ports 1/3 and 1/4 are forwarded.

On VLAN 20, ports 1/1 and 1/2 are connected to a DHCP server. DHCP server ports are set to trusted.

```
device(config)#interface ethernet 1/1
device(config-if-e1000-1/1)#dhcp-snooping-trust
device(config-if-e1000-1/1)#exit
device(config)#interface ethernet 1/2
device(config-if-e1000-1/2)#dhcp-snooping-trust
device(config-if-e1000-1/2)#exit
```

Hence, DHCP sever reply packets received on ports 1/1 and 1/2 are forwarded, and client IP or MAC binding information is collected.

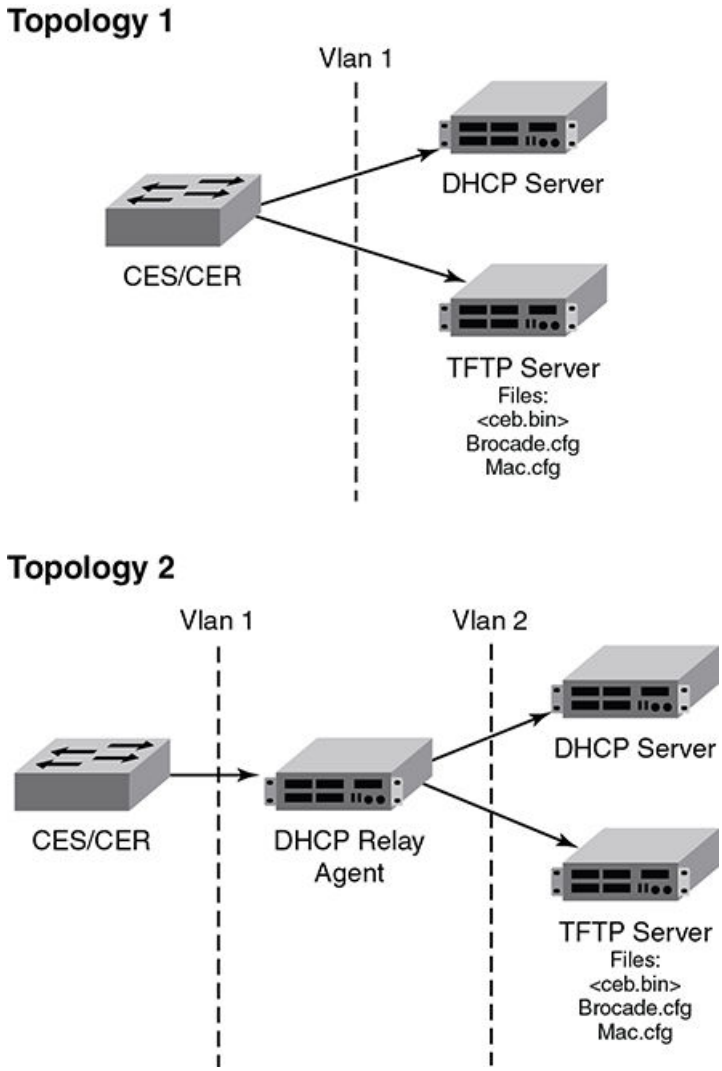
The example also sets the DHCP server address for the local relay agent.

```
device(config)# interface ve 2
device(config-vif-2)#ip address 10.20.20.1/24
device(config-vif-2)#ip helper-address 10.30.30.4
device(config-vif-2)#interface ve 20
device(config-vif-20)#ip address 10.30.30.1/24
```

# Zero Touch Provisioning

Zero Touch Provisioning allows Extreme devices to automatically obtain a dynamically assigned DHCP address, negotiate address lease renewal, and download flash image and configuration files. Figure 40 provides information about this feature.

FIGURE 40 Zero Touch Provisioning




Zero Touch Provisioning consists of the following steps.

1. The IP address validation and lease negotiation enables the DHCP client to automatically obtain and configure an IP address.
   a) As the Extreme device comes online, it checks if the DHCP client is enabled on any of the data ports.
   b) If no data port is enabled, the device tries to obtain an address on the management port.

   NOTE
   The management port is not enabled by default and needs to be enabled manually for the feature to operate.If the management port is configured with a static IP address, the Zero Touch Provisioning feature is automatically disabled.

2. The TFTP flash image is downloaded and updated. The device compares the file names of the requested flash image and the image stored in flash memory. If the names are different, the device downloads the new image from a TFTP server and writes the downloaded image to flash memory.
3. The Zero Touch Provisioning feature supports update of the existing monitor image and reload of the device.

4. The device downloads configuration files from a TFTP server and saves them as the running configuration.

# Zero Touch Provisioning limitations

The following limitations apply to the Zero Touch Provisioning feature.

- By default, Zero Touch Provisioning is always enabled on a management port.
- Zero Touch Provisioning fails on a management port which has a static address configured on it.
- Zero Touch Provisioning does not support trunked ports or Link Aggregation Control Protocol (LACP) ports.
- During the Zero Touch Provisioning update, the existing configuration takes precedence over any configuration downloaded from the TFTP server.
- VE and VLAN numbers that are chosen for Zero Touch Provisioning cannot be used for other configurations.

# Upgrade and downgrade considerations

- During a network upgrade procedure, the downloaded configuration files (as a part of the Zero Touch Provisioning process) may contain commands that cannot be executed using the current software version. In such a scenario, download the configuration files after a system reboot following the image download.
- During a network downgrade procedure, inspect the running configuration, because the system ignores errors due to incompatible commands from the previous configuration.

# Supported options for DHCP

Zero Touch Provisioning supports the following DHCP options:

- DHCP Parameter Request List
  - Subnet Mask
  - Domain Name
  - Router
  - Host Name (optional)
  - TFTP Server Name
- DHCP Client
  - Server ID
  - IP Address Lease
  - Renewal Time Value
  - Rebind Time value
  - Subnet Net mask
  - Domain Name
  - Router
  - Domain Server
  - Host Name
  - TFTP Server Name

# Supported messages for DHCP servers

Zero Touch Provisioning supports the following DHCP messages:

- DHCPACK

- DHCPDECLINE
- DHCPDISCOVER
- DHCPNAK
- DHCPOFFER
- DHCPRELEASE
- DHCPREQUEST

NOTE
Zero Touch Provisioning does not support the DHCPINFORM message.

# Configuring Zero Touch Provisioning

Zero Touch Provisioning allows this device to dynamically update its running configuration. This feature uses the DHCP client for address allocation and the TFTP server to download a specific configuration file.

NOTE
Virtual Ethernet (VE) and virtual LAN (VLAN) have to exist for the DHCP configuration to work.

To enable Zero Touch Provisioning on a port, use the **ip dhcp-client vlan** command. This command enables the autoconfiguration on any in-band port with the DHCP client configured on it.

```
device(config)# ip dhcp-client vlan 227 ve 227 tagged 1/1 auto-update enabled
```

**Syntax: [no] ip dhcp-client vlan** *vlannumber* **ve** [ *venumber* | **tagged** | **untagged** | *slot/port* | **auto-update enabled** | **auto-update disabled** ]

The *vlan number* variable is the desired VLAN number for sending out tagged or untagged DHCP requests.

The *ve number* variable is the router interface number for sending out tagged or untagged DHCP requests.

The **tagged** or **untagged** options add the port as a tagged or untagged member of the VLAN.

The *slot/port* variable is the desired slot and port for the DHCP client.

The **auto-update enabled** option enables autoconfiguration on the port.

The **auto-update disabled** option disables autoconfiguration on the port.

NOTE
VLAN and VE are created when you run the **ip dhcp-client vlan** command.

## *Disabling auto-update on a port*

To disable only the auto-update feature on a port, enter the following command.

```
device(config)# ip dhcp-client default-vlan untagged 1/1 auto-update disabled
```

**Syntax: ip dhcp-client vlan** *vlannumber* **ve** [ *venumber* | **tagged** | **untagged** | *slot/port* | **auto-update disabled** ]

## *Enabling autoconfiguration on a default VLAN*

To enable autoconfiguration on a default VLAN, use the following command when auto-update is enabled on port 1/1.

```
device(config)# ip dhcp-client default-vlan untagged 1/1 auto-update enabled
```

## Enabling autoconfiguration on a tagged VLAN

To enable autoconfiguration on a tagged VLAN, use the following command when the VLAN number is 227, the VE number is 227, and auto-update is enabled on port 1/1.

```
device(config)# ip dhcp-client vlan 227 ve 227 tagged 1/1 auto-update enabled
```

## Enabling autoconfiguration on an untagged VLAN

To enable autoconfiguration on an untagged VLAN, use the following command when the VLAN number is 227, the VE number is 227, and auto-update is enabled on port 1/1.

```
device(config)# ip dhcp-client vlan 227 ve 227 untagged 1/1 auto-update enabled
```

## Enabling autoconfiguration on a management port

To enable autoconfiguration on a management port, use the following command.

```
device(config)# ip dhcp-client vlan
```

Use the **no ip dhcp-client vlan** command to disable the DHCP client and autoconfiguration for the designated port.

## Displaying Zero Touch Provisioning information

Run the **show ip** and the **show ip interface** commands to display information about the successful implementation of Zero Touch Provisioning.

```
device#show ip
Global Settings
IP CAM Mode: static IPVPN CAM Mode: static
  ttl: 64, arp-age: 10, bootp-relay-max-hops: 4, icmp-error-rate: 400
  IP Router-Id: 10.1.1.1
DHCP server address: 10.21.96.1
TFTP server address: 10.21.96.1
Configuration filename: extreme.cfg
enabled: UDP-Broadcast-Forwarding  ICMP-Redirect  Source-Route  Load-Sharing  RARP
disabled: Directed-Broadcast-Forwarding  drop-arp-pending-packets  IRDP  Proxy-ARP  RPF-Check  RPF-Exclude-
Default  RIP  BGP4 IS-IS OSPF  VRRP  VRRP-Extended  VSRP
Configured Static Routes: 2

device#show ip interface
Interface  IP-Address      OK?  Method   Status       Protocol   VRF          Type     Lease Time
eth 1/1    10.1.1.1         YES  NVRAM    admin/down   down       default-vrf  Static   N/A
mgmt 1     10.21.96.160   YES  NVRAM    up           up         default-vrf  Dynamic  672651
```

Table 39 describes the fields from the output of **show ip interface** command.

**TABLE 39** Output display of show ip interface command

| Field | Description |
|---|---|
| Interface | The type and the slot and port number of the interface. |
| IP-Address | The IP address of the interface.<br><br>NOTE<br>If an "s" is listed following the address, this is a secondary address. When the address was configured, the interface already had an IP address in the same subnet, so the software required the "secondary" option before the software could add the interface. |
| OK? | Whether the IP address has been configured on the interface. |

**TABLE 39** Output display of show ip interface command (continued)

| Field | Description |
|---|---|
| Method | Whether the IP address has been saved in NVRAM. If you have set the IP address for the interface in the CLI or Web Management Interface, but have not saved the configuration, the entry for the interface in the Method field is "manual". |
| Status | The link status of the interface. If you have disabled the interface with the **disable** command, the entry in the Status field will be "administratively down". Otherwise, the entry in the Status field will be either "up" or "down". |
| Protocol | Whether the interface can provide two-way communication. If the IP address is configured, and the link status of the interface is up, the entry in the Protocol field will be "up". Otherwise, the entry in the Protocol field will be "down". |
| VRF | The VRF type. |
| Type | The type of lease. |
| Lease Time | The time when this lease will expire. |

# DHCPv6



# DHCP relay agent for IPv6

A client locates a DHCP server using a reserved, link-scoped multicast address. Direct communication between client and the server requires that they be attached by the same link. In some situations where ease-of-management, economy, and scalability are concerns, you can allow a DHCPv6 client to send a message to a DHCP server using a DHCPv6 relay agent. A DHCPv6 relay agent, which may reside on the client link, but is transparent to the client, relays messages between the client and the server.

When the relay agent receives a message, it creates a new relay-forward message, inserts the original DHCPv6 message, and sends the relay-forward message as the DHCP server.

## Configuring DHCP for IPv6 relay agent

You can enable the DHCP for IPv6 relay agent function and specify the relay destination address (i.e. the DHCP server) on an interface by entering this command at the interface level.

```
device(config)# interface ethernet 2/3
device(config-if-e10000-2/3)#ipv6 dhcp-relay destination 2001:DB8::2
```

**Syntax: [no] ipv6 dhcp-relay destination** *ipv6-address*

Specify the *ipv6-address* as a destination address to which client messages are forwarded and which enables DHCP for IPv6 relay service on the interface. A maximum of 16 relay destination addresses may be entered.

### *DHCPv6 relay agent include options*

You can configure the DHCPv6 relay agent to include the client's remote ID, interface ID, or client link layer address as identifiers in the relay forward DHCPv6 messages.

In some network environments, it is useful for the relay agent to add information to the DHCPv6 message before relaying it. The information that the relay agent carries can also be used by the DHCP server to make decisions about the addresses, delegated prefixes, and configuration parameters that the client should receive. The DHCPv6 relay-forward message contains relay agent parameters that identify the client-facing interface on which the reply messages can be forwarded. You can use either one or all of the parameters as client identifiers.

The following options can be included in the relay-forward messages:

- Interface-ID option (18)
- Remote-ID option (37)
- Client link layer (MAC) address option (79)

The relay agent may send the interface-ID option (18) to identify the interface on which a client message was received. If the relay agent cannot use the address in the link-address field to identify the interface through which the response to the client will be relayed, the relay agent must include an interface-ID option in the relay-forward message. If the relay agent receives a relay-reply message with an interface-ID option, the relay agent relays the message to the client through the interface identified by the option. The server must also copy the interface-ID option from the relay-forward message into the relay-reply message the server sends to the relay agent in response to the relay-forward message.

The remote-ID option (37) may be added by the DHCP relay agent that terminates switched or permanent circuits and uses a mechanism to identify the remote host end of the circuit. The remote ID must be unique. A DHCPv6 relay agent can be configured to include a remote-ID option in the relay-forward DHCPv6 messages.

The client link layer (MAC) address option (79) can be used along with other identifiers to associate DHCPv4 and DHCPv6 messages from a dual-stack client, and is useful in environments where operators using an existing DHCPv4 system with the client link layer address as the customer identifier need to correlate DHCPv6 assignments using the same identifier.

> NOTE
> If you enable the client link layer (MAC) option and save the configuration, and then downgrade to a version of the software that does not support this feature, an error message displays. You must remove any configuration related to this option before the downgrade and add the configuration after the upgrade to prevent this error.

### *Specifying the IPv6 DHCP relay include options*

You can specify either one or all of the IPv6 DHCP relay include options in the relay-forward message.

The options include the interface-ID, remote-ID, or client MAC address. Perform the following steps to include the DHCPv6 relay options.

1. Enter global configuration mode by issuing the **configure terminal** command.

   ```
   device# configure terminal
   ```

2. Enter interface configuration mode.

   ```
   device(config)# interface ethernet 2/1
   ```

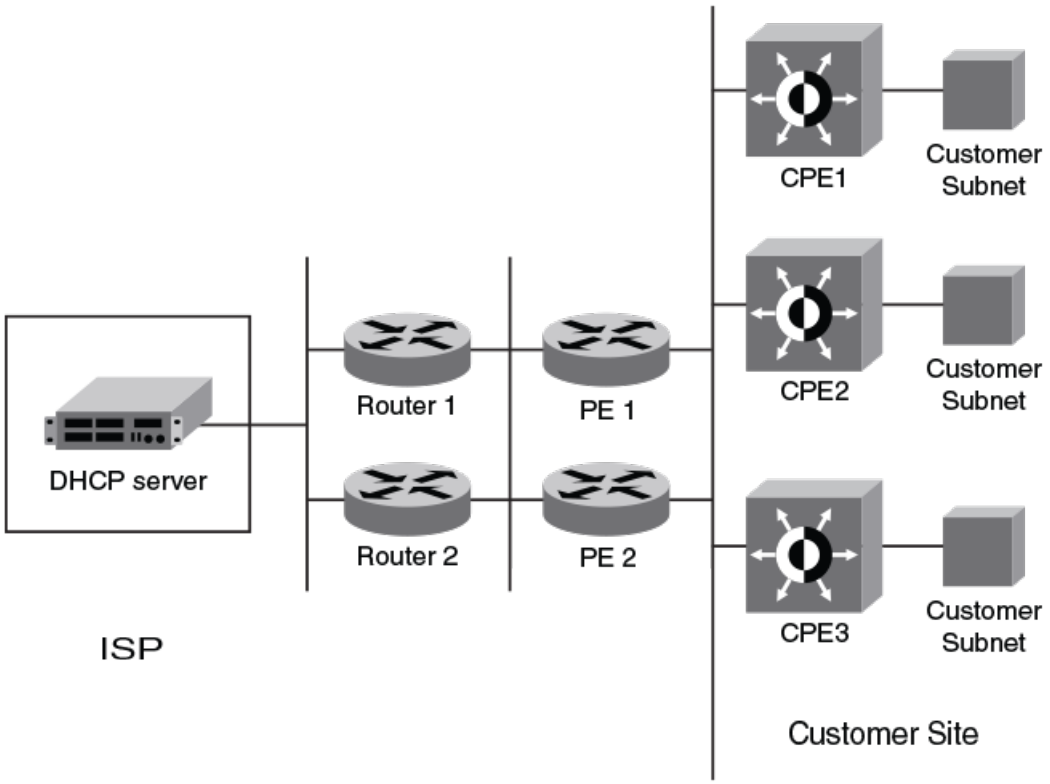
3. Enter the **ipv6 dhcp-relay include-options** command followed by the required options - **interface-ID**, **remote-ID** or **client-MAC-address**.
   The following example shows specifying the client-MAC-address as an option.

   ```
   device(config-if-e1000-2/1)# ipv6 dhcp-relay include-options client-mac-address
   ```

## DHCPv6 Relay Agent Prefix Delegation Notification

DHCPv6 Relay Agent Prefix Delegation Notification allows a DHCPv6 server to dynamically delegate IPv6 prefixes to a DHCPv6 client using the DHCPv6 Prefix Delegation (PD) option. DHCPv6 prefix delegation enables an Internet Service Provider (ISP) to automate the process of assigning prefixes to a customer premises equipment (CPE) network. The CPE then assigns IPv6 subnets from the delegated IPv6 prefix to its downstream customer interfaces.

**FIGURE 41** DHCPv6 Relay Agent Prefix Delegation Notification




A route is added to the IPv6 route table on the provider edge router (PE) for the delegated prefix to be delegated to requesting routers. The DHCP server chooses a prefix for delegation and responds with it to the CPEx. to the external network, and to enable the correct forwarding of the IPv6 packets for the delegated IPv6 prefix. Adding the delegated prefix to the IPv6 route table ensures that the unicast Reverse Path Forwarding (uRPF) works correctly.

Because the PE is also a DHCPv6 relay agent (it relays DHCPv6 messages between the CPE and the DHCP server), it examines all DHCPv6 messages relayed between the CPE and the DHCP server and gathers information about a delegated prefix and then manages the advertisement of this delegated prefix to the external network.

## *DHCPv6 Relay Agent Prefix Delegation Notification limitations*

The following limitations apply to DHCPv6 Relay Agent Prefix Delegation Notification.

- The PD notification fails when the DHCPv6 messages between a DHCPv6 server and a DHCPv6 client containing the PD option are not relayed by way of the DHCPv6 relay agent.
- If the delegated prefix is released or renewed by the client at the time when the DHCPv6 relay agent is down or rebooting, then this release or renewal of the delegated prefix will not be detected by the relay agent. In such a condition, there could be stale static routes in the routing table. You must clear the stale routes.
- If there is no sufficient disk space on a flash disk, then the system may not store all the delegated prefixes in the IPv6 route table.

## Upgrade and downgrade considerations

- When a router is upgraded to the version of software that supports this feature DHCPv6 Relay Agent Prefix Delegation Notification, the saved information about delegated prefixes will be examined and if the delegated prefix lifetime is not expired, then the prefix will be added to the IPv6 static route table.
- When a router is downgraded to the version of software that does not support DHCPv6 Relay Agent Prefix Delegation Notification, the saved information about delegated prefixes is retained and it cannot be used.

## Configuring DHCPv6 Relay Agent Prefix Delegation Notification

To set the number of delegated prefixes that can be learned at the global system level, use the **ipv6 dhcp-relay maximum-delegated-prefixes** command.

By default, the DHCPv6 Relay Agent Prefix Delegation Notification is enabled when the DHCPv6 relay agent feature is enabled on an interface. User can disable the DHCPv6 Relay Agent Prefix Delegation Notification at the system or the interface level by setting **ipv6 dhcp-relay maximum-delegated-prefixes** to 0 at the system or interface level as required.

> NOTE
> There should be a minimum free space of 7 MB in the flash memory to save information about delegated prefixes in flash on both the Active and Standby management processor

```
device(config)# ipv6 dhcp-relay maximum-delegated-prefixes 5000
```

**Syntax: [no] ipv6 dhcp-relay maximum-delegated-prefixes** *value*

The *value* parameter is used to limit the maximum number of prefixes that can be learned at the global level. The range is from 0 to 100000. The default value varies for different platforms.

Use the **no ipv6 dhcp-relay maximum-delegated-prefixes** command to set the parameter to the default value of the specified platform. Refer to Enabling DHCPv6 Relay Agent Prefix Delegation notification on an interface on page 398 for more information.

### Enabling DHCPv6 Relay Agent Prefix Delegation notification on an interface

To set the number of delegated prefixes that can be learned at the interface level, use the **ipv6 dhcp-relay maximum-delegated-prefixes** command. This command limits the maximum number of prefixes that can be learned on the interface.

```
device(config-if-eth2/1)# ipv6 dhcp-relay maximum-delegated-prefixes 4000
```

**Syntax: [no] ipv6 dhcp-relay maximum-delegated-prefixes** *value*

The *value* parameter is used to limit the maximum number of prefixes that can be delegated. The range is from 0 to 20000. The default value is 20000.

Use the **no ipv6 dhcp-relay maximum-delegated-prefixes** command to set the parameter to the default value of the specified platform.

Table 40 lists the default and maximum prefix values for different platforms.

**TABLE 40** Default and maximum values for different platforms

| Platform | Default Maximum System Prefixes supported | Maximum System Prefixes supported | Default Maximum Interface Prefixes supported | Maximum interface Prefixes supported | Default Maximum IPv6 Route | Maximum IPv6 routes that can be supported |
|---|---|---|---|---|---|---|
| CES 2000 Series | 1000 | 8000 | 250 | 2000 | 1024 | 8192 |
| CER 2000 Series | 8000 | 100000 | 2000 | 20000 | 8192 | 131072 |
| MLX Series | 32000 | 100000 | 8000 | 20000 | 32768 | 114688 |
| XMR Series | 60000 | 100000 | 20000 | 20000 | 65536 | 245760 |

### Assigning the administrative distance to DHCPv6 static routes

To assign the administrative distance to DHCPv6 static routes installed in IPv6 route table for the delegated prefixes on the interface, use the **ipv6 dhcp-relay distance** command at the interface level. The administrative distance value has to be set so that it does not replace the same IPv6 static route configured by the user.

```
device(config-if-eth-2/1)# ipv6 dhcp-relay distance 25
```

**Syntax: [no] ipv6 dhcp-relay distance** *value*

The *value* parameter is used to assign the administrative distance to DHCPv6 static routes on the interface. The range is from 1 to 255. The default value is 10. If the value is set to 255, then the delegated prefixes for this interface will not be installed in the IPv6 static route table.

Use the **no ipv6 dhcp-relay distance** command to set the parameter to a default value of 10.

## Displaying the DHCPv6 Relay Agent Prefix Delegation Notification information

Enter the **show ipv6 dhcp-relay delegated-prefixes** command to display information about the delegated prefixes.

```
device#show ipv6 dhcp-relay delegated-prefixes vrf red
IPv6 DHCP Relay Delegated Prefixes Table - 2 entries VRF: red
IPv6 Prefix           Client             Interface     ExpireTime
2001:db8:aaa::/48     2001:db8:103:10:1::8        eth 1/3       3h24m10s
2001:db8:bbb::/48     2001:db8:104:10:1::6        eth 1/4       0m28s
device#
```

**Syntax: show ipv6 dhcp-relay delegated-prefixes vrf** *vrf-name* } { *X:X::X:X/M* | **client-id** *client ipv6 address* | **interface** *interface-id* }

The **vrf** *vrf-name* parameter is used to display the DHCPv6 delegated prefixes for a specific VRF.

The *X:X::X:X/M* parameter is used to display the specified delegated prefix information.

The **client-id***client ipv6 address* parameter is used to display the delegated prefix for the specific client.

The **interface***interface-id* parameter is used to display delegated prefixes for the specified outgoing interface.

Table 41 describes the fields from the output of **show ipv6 dhcp-relay delegated-prefixes** command.

**TABLE 41** Output from the show ipv6 dhcp-relay delegated-prefixes command

| Field | Description |
|---|---|
| IPv6 Prefix | The IPv6 prefix delegated to the client. |
| Client | The IPv6 address of the client. |
| Interface | The interface on which the DHCPv6 messages are relayed to the client. |
| ExpireTime | The remaining lifetime of the delegated prefix. |

### *Displaying the DHCPv6 Relay configured destinations*

Enter the **show ipv6 dhcp-relay destinations** command to display information about the delegated prefixes' configured destinations for a specific interface.

```
device#show ipv6 dhcp-relay destinations
DHCPv6 Relay Destinations:
Interface ve 100:
   Destination                   OutgoingInterface
   2001:db8:1::39                   NA
Interface ve 101:
   Destination                   OutgoingInterface
```

```
   2001:db8:1::39                    NA
Interface ve 102:
   Destination                    OutgoingInterface
   2001:db8:1::39                    NA
```

**Syntax: show ipv6 dhcp-relay destinations**

Table 42 describes the fields from the output of **show ipv6 dhcp-relay destinations** command.

**TABLE 42** Output from the show ipv6 dhcp-relay destinations command

| Field | Description |
|---|---|
| Destination | The configured destination IPv6 address. |
| OutgoingInterface | The interface on which packets will be relayed if the destination relay address is local link or multicast. |

## *Displaying the DHCPv6 relay agent options*

Enter the **show ipv6 dhcp-relay options** command to display information about the relay options available to the prefixed delegates for a specific interface.

```
device# show ipv6 dhcp-relay options
DHCPv6 Relay Options Information:
Interface       Interface-Id     Remote-Id   Client-mac-address
ve 100          No               No                Yes
ve 101          Yes              No
ve 102          No               Yes
```

**Syntax: show ipv6 dhcp-relay options**

The following table describes the fields from the output of the **show ipv6 dhcp-relay options** command.

**TABLE 43** Output from the show ipv6 dhcp-relay options command

| Field | Description |
|---|---|
| Interface | The interface name. |
| Interface-Id | The interface ID option. Yes indicates the option is used; No indicates the option is not used. |
| Remote-Id | The remote ID option. Yes indicates the option is used; No indicates the option is not used. |
| Client-mac-address | The client MAC address option. Yes indicates the option is used; No indicates the option is not used. |

## *Displaying the DHCPv6 Relay prefix delegation information*

Enter the **show ipv6 dhcp-relay prefix-delegation-information** command to display additional information about the DHCPv6 prefix delegation.

```
device# show ipv6 dhcp-relay prefix-delegation-information
DHCPv6 Relay Prefix Delegation Notification Information:
  Interface  Current    Maximum    AdminDistance
  ve 100     20         20000      10
  ve 101     4000       20000      10
  ve 102     0          20000      10
  ve 103     0          20000      10
  ve 104     0          20000      10
  ve 105     0          20000      10
```

**Syntax: show ipv6 dhcp-relay prefix-delegation-information**

Table 44 describes the fields from the output of the **show ipv6 dhcp-relay prefix-delegation-information** command.

**TABLE 44** Output from the show ipv6 dhcp-relay prefix-delegation-information command

| Field | Description |
|---|---|
| Interface | The interface name. |
| Current | The number of delegated prefixes currently learned on the interface. |
| Maximum | The maximum number of delegated prefixes that can be learned on the interface. |
| AdminDistance | The current administrative distance used for prefixes learned on this interface when added to the IPv6 static route table. |

## *Displaying the DHCPv6 relay information for an interface*

Enter the **show ipv6 dhcp-relay interface** command to display DHCPv6 relay information for a specific interface.

```
device# show ipv6 dhcp-relay interface ve 100
DHCPv6 Relay Information for interface ve 100:
Destinations:
 Destination                   OutgoingInterface
 2001:db8:1::39                   NA
Options:
 Interface-Id: No     Remote-Id:No  Client-mac-address: Yes
Prefix Delegation Notification:
 Current:0 Maximum:20000 AdminDistance:10
```

**Syntax: show ipv6 dhcp-relay interface** *interface type*

The *interface type* variable presents the interface type, such as Ethernet, Point of Service (POS), or VE and the specific port number.

The following table describes the fields from the output of the **show ipv6 dhcp-relay interface** command.

**TABLE 45** Output from the show ipv6 dhcp-relay interface command

| Field | Description |
|---|---|
| Destinations | The DHCPv6 relay destination configured on the interface.<br>• Destination: The configured destination IPv6 address.<br>• OutgoingInterface: The interface on which packet will be relayed if the destination relay address is link local or multicast. |
| Options | The current information about DHCPv6 relay options for the interface.<br>• Interface-Id: The interface ID option indicating whether the option is used.<br>• Remote-Id: The remote ID option indicating whether the option is used.<br>• Client-mac-address: The client MAC address indicating whether the option is used. |
| Prefix Delegation Notification | The current information about the DHCPv6 prefix delegation for the interface.<br>• Interface: The name of the interface.<br>• Current: The number of delegated prefixes currently learned on the interface.<br>• Maximum: The maximum number of delegated prefixes that can be learned on the interface.<br>• AdminDistance: The administrative distance used for prefixes learned on the specific interface when added to the IPv6 static route table. |

## *Clearing the DHCPv6 delegated prefixes*

To clear the DHCPv6 delegated prefixes for specific VRFs, use the **clear ipv6 dhcp-relay delegated-prefixes** command at the privilege level.

```
device# clear ipv6 dhcp-relay delegated-prefixes vrf VRF1
```

**Syntax: clear ipv6 dhcp-relay delegated-prefixes** { **vrf** *vrf-name* } { **X:X::X:X/M** | **all** | **interface** *interface-id* }

The **vrf***vrf-name* parameter is used to clear the DHCPv6 delegated prefixes for a specific VRF. If this parameter is not provided, then the information for the default VRF is cleared.

The *X:X::X:X/M* parameter is used to clear the specified delegated prefix and remove the corresponding route permanently from the router.

The **all** parameter is used to clear all the delegated prefixes and remove the corresponding routes permanently from the router for the VRF.

The **interface***interface-id* parameter is used to clear all the delegated prefixes and remove the corresponding routes permanently from the router for the specified outgoing interface.

### *Clearing the DHCPv6 packet counters*

To clear all DHCPv6 packet counters, use the **clear ipv6 dhcp-relay statistics** command at the privilege level.

```
device# clear ipv6 dhcp-relay statistics
```

**Syntax: clear ipv6 dhcp-relay statistics**

## Enabling support for network-based ECMP load sharing for IPv6

If network-based ECMP load sharing is configured, traffic is distributed across equal-cost paths based on the destination network address. Routes to each network are stored in CAM and accessed when a path to a network is required. Because multiple hosts are likely to reside on a network, this method uses fewer CAM entries than load sharing by host. When you configure network-based ECMP load sharing, you can choose either of the following CAM modes:

- **Dynamic mode** - In the dynamic mode, routes are entered into the CAM dynamically using a flow-based scheme, where routes are only added to the CAM as they are required. Once routes are added to the CAM, they can be aged-out when they are not in use. Because this mode conserves CAM, it is useful for situations where CAM resources are stressed or limited.
- **Static mode** - In the static mode, routes are entered into the CAM whenever they are discovered. Routes are not aged once routes are added to the CAM and can be aged-out when they are not in use.

IPv6 VPN CAM supports ECMP load sharing, which is created for IPv6 VPN routes.

### *Configuring the CAM mode to support network-based ECMP load sharing for IPv6*

To configure the CAM mode to support network-based ECMP load sharing for IPv6, enter a command such as the following at the Global Configuration level.

```
device(config)# cam-mode ipv6 dynamic
```

**Syntax: [no] cam-mode ipv6** [ **dynamic** | **static** | **host** ]

The **dynamic** parameter configures the device for network-based ECMP load sharing using the dynamic CAM mode.

The**static** parameter configures the device for network-based ECMP load sharing using the static CAM mode.

The **host** parameter configures the device for host-based ECMP load sharing using the dynamic CAM mode.

You must restart the device for this command to take effect.

# Displaying ECMP load-sharing information for IPv6

To display the status of ECMP load sharing for IPv6, enter the following command.

```
device# show ipv6
Global Settings
  unicast-routing enabled, ipv6 allowed to run, hop-limit 64
  reverse-path-check disabled
  urpf-exclude-default disabled
  session-logging-age 5
  No Inbound Access List Set
  No Outbound Access List Set
  Prefix-based IPv6 Load-sharing is Enabled, Number of load share paths: 4
  source-route disabled, forward-source-route disabled
Configured Static Routes: 66
Configured Static Mroutes: 66
RIP: enabled
OSPF (default VRF): enabled
BGP: enabled, 1 active neighbor(s) configured
```

**Syntax: show ipv6**

You can display the entries in the IPv6 forwarding cache by entering the **show ipv6 cache** command.

```
device# show ipv6 cache
Total number of IPv6 and IPv6 VPN cache entries: 3
       IPv6 Address                      Next Hop                    Interface
1      6000::                            LOCAL                       ve 60
2      6000::2                           LOCAL                       ve 60
3      fe80::768e:f8ff:fe2a:6200         LOCAL                       ve 60
```

**Syntax: show ipv6 cache** [ *index-number* | *ipv6-prefix/prefix-length* | *ipv6-address* | **ethernet** *port* | **ve** *number* | **tunnel** *number* ]

# IS-IS (IPv4)

# IS-IS overview

The Intermediate System to Intermediate System (IS-IS) protocol is a link-state Interior Gateway Protocol (IGP) that is based on the International Standard for Organization/ International Electrotechnical Commission (ISO/IEC) Open Systems Internet Networking model (OSI). In IS-IS, an intermediate system (router) is designated as either a Level 1 or Level 2 device. A Level 1 router routes traffic only within the area in which the router resides. A Level 2 router routes traffic between areas within a routing domain.

The implementation of IS-IS is based on the following specifications and draft specifications:

- ISO/IEC 10589 - "Information Technology - Telecommunication and information exchange between systems - Intermediate system to Intermediate system intra-domain routing information exchange protocol for use in conjunction with the protocol for providing the connection less-mode Network Service (ISO 8473)", 1992
- ISO/IEC 8473 - "Information processing systems - Data Communications - Protocols for providing the connectionless-mode network service", 1988
- ISO/IEC 9542 - "Information Technology - Telecommunication and information exchange between systems - End system to Intermediate system intra-domain routing information exchange protocol for use in conjunction with the protocol for providing the connection less-mode Network Service (ISO 8473)", 1988
- RFC 1195 - "Use of OSI IS-IS for Routing in TCP/IP and Dual Environments", 1990.
- RFC 2763 - "Dynamic Host Name Exchange Mechanism for IS-IS", 2000.
- RFC 2966 - "Domain-wide Prefix Distribution with Two-Level IS-IS", 2000
- RFC 3373 - "Three-Way Handshake for Intermediate System to Intermediate System (IS-IS) Point-to-Point Adjacencies", 2002
- Portions of the Internet Draft "IS-IS extensions for Traffic Engineering" draft-ieff-isis-traffic-02.txt (dated 2000). that describe the Extended IP reachability type-length-value (TLV type 135) and the extended Intermediate System (IS) reachability TLV (TLV type 22). These portions provide support for the wide metric version of IS-IS. No other portion is supported on Extreme's implementation of IS-IS.

  NOTE
  The Extreme device does not support routing of Connectionless-Mode Network Protocol (CLNP) packets. The Extreme device uses IS-IS for TCP/IP only.

# Relationship to the IP route table

The IS-IS routes are calculated and first placed in the IS-IS route table. The routes are then transferred to the IP route table.

The best IS-IS path for a given destination is sent to the IP route table for comparison to the best paths from other protocols to the same destination. The CPU selects the path with the lowest administrative distance and places that path in the IP route table:

- If the path provided by IS-IS has the lowest administrative distance, then the CPU places that IS-IS path in the IP route table.
- If a path to the same destination supplied by another protocol has a lower administrative distance, the CPU installs the other protocol's path in the IP route table instead.

The **administrative distance** is a protocol-independent value from 1 - 255. Each path sent to the CPU, regardless of the source of the path (IS-IS, OSPF, static IP route, and so on) has an administrative distance.

Each route source has a default administrative distance. The default administrative distance for IS-IS is 115.

You can change the administrative distance for IS-IS and other routes sources.

# Intermediate systems and end systems

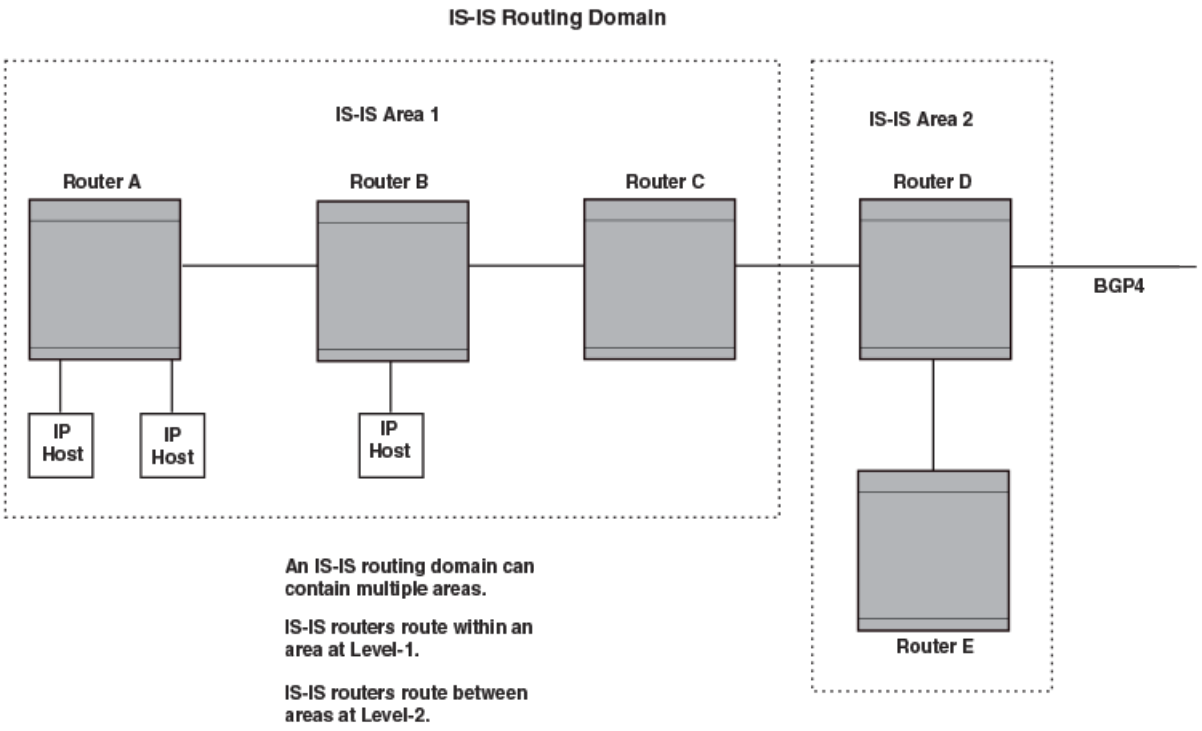
IS-IS uses the following categories to describe devices within an IS-IS routing domain (similar to an OSPF Autonomous System):

- Intermediate System (IS) - A device capable of forwarding packets from one device to another within the domain. In Internet Protocol (IP) terminology, an IS is a router.
- End System (ES) - A device capable of generating or receiving packets within the domain. In IP terminology, an ES is an end node or IP host.

When you configure IS-IS on a device, the device is an IS.

Figure 42 shows an example of an IS-IS network.

**FIGURE 42** An IS-IS network contains Intermediate Systems (ISs) and host systems




NOTE

Since the implementation of IS-IS does not route OSI traffic but instead routes IP traffic, IP hosts are shown instead of ESs.

The other basic IS-IS concepts illustrated in this figure are explained in the following sections.

# Domain and areas

IS-IS is an IGP, and thus applies only to routes within a single routing domain. However, you can configure multiple areas within a domain. A device can be a member of one area for each Network Entity Title (NET) you configure on the device. The NET contains the area ID for the area the NET is in.

In Intermediate systems and end systems on page 407, Routers A, B, and C are in area 1. Routers D and E are in area 2. All the routers are in the same domain.

# Level-1 routing and Level-2 routing

You can configure an IS-IS router to perform one or both of the following levels of IS-IS routing:

NOTE
The ISO/IEC specifications use the spelling "routeing", but this document uses the spelling "routing" to remain consistent with other Extreme documentation.

- Level-1 - A Level-1 router routes traffic only within the area the router is in. To forward traffic to another area, the Level-1 router sends the traffic to its nearest Level-2 router.
- Level-2 - A Level-2 router routes traffic between areas within a domain.

In Intermediate systems and end systems on page 407, Routers A and B are Level-1s only. Routers C and D are Level-1 and Level-2 ISs. Router E is a Level-1 IS only.

# Neighbors and adjacencies

A device configured for IS-IS forms an adjacency with each of the IS-IS devices to which it is directly connected. An adjacency is a two-way direct link (a link without router hops) over which the two devices can exchange IS-IS routes and other protocol-related information. The link is sometimes called a “circuit”. The devices with which the device forms adjacencies are its neighbors , which are other ISs.

In Intermediate systems and end systems on page 407, Router A has an IS-IS adjacency with Router B. Likewise, Router B has an IS-IS adjacency with Router A and Router C.

# Designated IS

A Designated IS is an IS-IS router that is responsible for gathering and distributing link state information to other Level 1 or Level 2 ISs within the same broadcast network (LAN). The Level 1 and Level 2 Designated ISs within a broadcast network are independent, although the same device can be a Level 1 Designated IS and a Level 2 Designated IS at the same time.

The Designated IS is elected based on the priority of each IS in the broadcast network. When an IS becomes operational, it sends a Level 1 or Level 2 Hello PDU to advertise itself to other ISs. If the IS is configured to be both a Level 1 and a Level 2 IS, the IS sends a separate advertisement for each level:

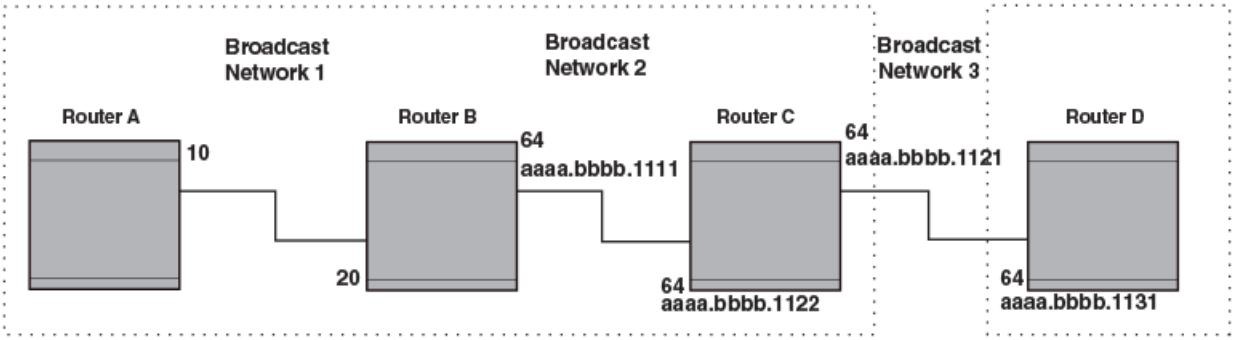
- The Level 1 IS that has the highest priority becomes the Level 1 Designated IS for the broadcast network.
- The Level 2 IS that has the highest priority becomes the Level 2 Designated IS for the broadcast network.

If the Designated IS becomes unavailable, for example in the case of a reboot, the IS with the next highest priority becomes the new IS. If two or more ISs have the highest priority, the IS with the highest MAC address becomes the Designated IS.

The priority is an interface parameter. Each interface that is enabled for IS-IS can have a different priority.

The figure below shows an example of the results of Designated IS elections. For simplicity, this example shows four of the five routers in Intermediate systems and end systems on page 407, with the same domain and areas.

**FIGURE 43** Each broadcast network has a Level-1 Designated IS and a Level-2 Designated IS




Designated IS election has the following results in this network topology:

- Router B is the Level 1 Designated IS for broadcast network 1
- Router C is the Level 1 Designated IS for broadcast network 2
- Router D is the Level 2 Designated IS for broadcast network 3

In this example, the IS-IS priorities for the IS-IS interfaces in broadcast network 1 have been changed by an administrator. The priorities for the interfaces in the other broadcast networks are still set to the default (64). When there is a tie, IS-IS selects the interface with the highest MAC address.

## Broadcast pseudonode

In a broadcast network, the Designated IS maintains and distributes link state information to other ISs by maintaining a pseudonode . A pseudonode is a logical host representing all the Level 1 or Level 2 links among the ISs in a broadcast network. Level 1 and Level 2 have separate pseudonodes, although the same device can be the pseudonode for Level 1 and Level 2.

## Route calculation and selection

The Designated IS uses a Shortest Path First (SPF) algorithm to calculate paths to destination ISs and ESs. The SPF algorithm uses Link State PDUs (LSPDUs) received from other ISs as input, and creates the paths as output.

After calculating the paths, the Designated IS then selects the best paths and places them in the IS-IS route table. The Designated IS uses the following process to select the best paths.

1. Prefer the Level 1 path over the Level 2 path.
2. If there is no Level 1 path, prefer the internal Level 2 path over the external Level 2 path.
3. If there is still more than one path, prefer the path with the lowest metric.
4. If there is more than one path with the lowest metric, load share among the paths.

   After selecting the best path to a destination, the software places the path in the IS-IS route table.

## Three-way handshake for point-to-point adjacencies

Three-Way Handshake for Point-to-Point adjacencies provides three-way handshake mechanisms on point-to-point interfaces for the following benefits:

- Identifies neighbor restarts within the holding time period

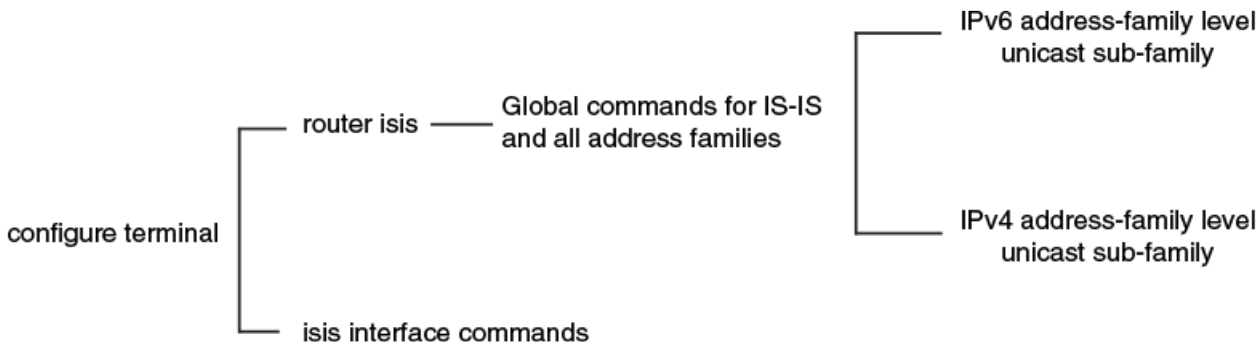
- Identifies uni-directional link failures and stops forming of an adjacency with a peer where such link failures occur.

  NOTE
  This feature is the default operation and cannot be turned off. Extreme devices with this feature are fully backward compatible with Extreme devices running an earlier release.

# IS-IS CLI levels

The CLI includes various levels of commands for IS-IS. The figure below illustrates these levels.

**FIGURE 44** IS-IS CLI levels




The following IS-IS CLI configuration modes are available

- ISIS router configuration mode: A global level for the configuration of the IS-IS protocol. At this level, all IS-IS configurations apply to IPv4 and IPv6.
- ISIS address-family IPv4 unicast configuration mode: Commands entered in ISIS address-family IPv4 unicast configuration mode apply only to IS-IS IPv4 configurations.
- ISIS address-family IPv6 unicast configuration mode: Commands entered in ISIS address-family IPv6 unicast configuration mode apply only to IS-IS IPv6 configurations.
- Interface subtype configuration mode: Various IS-IS options are configured on a specified interface.

Refer to the relevant Command Reference for more information on each of these configuration modes and for information on all of the commands accessed in these configuration modes.

Each address family configuration level allows you to access commands that apply to that particular address family only. To enable a feature in a particular address family, you must specify any associated commands for that feature in that particular address family. You cannot expect the feature, which you may have configured in the IPv4 IS-IS unicast address family, to work in the IPv6 IS-IS unicast address family unless it is explicitly configured in the IPv6 IS-IS unicast address family.

# Enabling IS-IS globally

When IS-IS is enabled on a device, the device enters IS-IS router configuration mode. Several commands can then be accessed that allow the configuration of IS-IS.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **router isis** command to enter IS-IS router configuration mode and enable IS-IS on the device.

```
device(config)# router isis
ISIS: Please configure NET!
```

3. Enter the **net** command and specify a NSAP address to configure a NET for IS-IS.

```
device(config-isis-router)# net 49.2211.0000.00bb.cccc.00
```

The following example enables IS-IS on a device.

```
device# configure terminal
device(config)# router isis
ISIS: Please configure NET!
device(config-isis-router)# net 49.2211.0000.00bb.cccc.00
```

# Configuring the IS-IS IPv4 unicast address family

The IS-IS IPv4 unicast address family allows you to configure IPv4 IS-IS unicast settings that are separate and distinct from IPv6 IS-IS unicast settings. The following task enables ISIS IPv4 address family unicast configuration mode where a variety of IS-IS features can be configured.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **router isis** command to enter IS-IS router configuration mode and enable IS-IS on the device.

```
device(config)# router isis
```

3. Enter the **address-family ipv4 unicast** command to enable ISIS address-family IPv4 unicast configuration mode.

```
device(config-isis-router)# address-family ipv4 unicast
```

The following example enables ISIS address-family IPv4 unicast configuration mode.

```
device# configure terminal
device(config)# router isis
device(config-isis-router)# address-family ipv4 unicast
device(config-isis-router-ipv4u)#
```

# Overload bit

If an IS's resources are overloaded and are preventing the IS from properly performing IS-IS routing, the IS can inform other ISs of this condition by setting the overload bit in LSPDUs sent to other ISs from 0 (off) to 1 (on). Thus, when an IS is overloaded, other ISs will not use the overloaded IS to forward traffic.

An IS can be in the overload state for Level 1 or Level 2 as follows:

- If an IS is in the overload state for Level 1, other Level 1 ISs stop using the overloaded IS to forward Level 1 traffic. However, the IS can still forward Level 2 traffic, if applicable.
- If an IS is in the overload state for Level 2, other Level 2 ISs stop using the overloaded IS to forward Level 2 traffic. However, the IS can still forward Level 1 traffic, if applicable.
- If an IS is in the overload state for both levels, the IS cannot forward traffic at either level.

By default, the device automatically sets the overload bit to 1 (on) in its LSPDUs to other ISs if an overload condition occurs.

The overload bit can also be set to administratively shut down IS-IS without disabling the protocol. Setting the overload bit on is useful when configuration changes are needed but you do not want to remove the device from the network. A device can also be configured to set the overload bit on for a specific number of seconds during startup, to allow IS-IS to become fully active before the device begins IS-IS routing.

## Setting the overload bit

The overload bit can be configured so that when an IS is overloaded, other ISs will not use the overloaded IS to forward traffic. The following task specifies that the overload bit is set upon system startup and remains set until BGP has converged and specifies that the device that 240 seconds is the maximum time that IS-IS will wait for BGP convergence to complete.

1. Enter the **configure terminal** command to access global configuration mode.

   ```
   device# configure terminal
   ```

2. Enter the **router isis** command to enter IS-IS router configuration mode and enable IS-IS on the device.

   ```
   device(config)# router isis
   ```

3. Enter the **set-overload-bit** command with the **on-startup** and **wait-for-bgp** parameters, specifying a value, to configure the device to set the overload bit upon system startup. The overload bit remains set until BGP has converged. 240 seconds is the maximum time that IS-IS waits for BGP convergence to complete.

   ```
   device(config-isis-router)# set-overload-bit on-startup wait-for-bgp 240
   ```

The following example configures the overload bit on system startup until BGP has converged. The maximum time that the device waits for BGP convergence to complete is 86400 seconds.

```
device# configure terminal
device(config)# router isis
device(config-isis-router)# set-overload-bit on-startup wait-for-bgp 86400
```

# Authentication

By default, a device does not authenticate packets sent to or received from an end system (ES) or other intermediate system (IS).

In previous releases, the NetIron software let you configure area, domain, and circuit passwords to direct the device to check for a password in packets sent from the device. The commands for setting the password used in previous versions of the NetIron software are now hidden in the CLI. However, they are backward compatible and will operate in this release.

An authentication password can be configured using the Hashed Message Authentication codes - Message Digest 5 (HMAC-MD5) algorithm, in conformance with RFC 3567 - Intermediate System to Intermediate System (IS-IS) Cryptographic Authentication.

IS-IS authentication checking is enabled by default. When transitioning from one authentication mode to another, changing the authentication mode can cause packets to drop because only some of the routers have been reconfigured. During such a transition, it can be useful to disable IS-IS authentication checking temporarily until all devices are reconfigured and the network is stable.

Authentication can be configured globally or for a specified interface.

To configure IS-IS authentication globally you must perform the following tasks:

- Configure IS-IS Authentication Mode
- Configure IS-IS Authentication Key
- Disable IS-IS Authentication Check (optional)

To configure IS-IS authentication on a specified interface, you must perform the following tasks:

- Configure IS-IS Interface Authentication Mode for the specified interface
- Configure IS-IS Authentication Key for the specified interface
- Disable IS-IS Authentication Check for the specified interface (optional)

## Configuring authentication

A device can be configured to authenticate packets sent to or received from an end system (ES) or other intermediate system (IS). The following task configures IS-IS authentication mode, an IS-IS authentication key and temporarily disables IS-IS authentication checking for Level 1 packets.

1. Enter the **configure terminal** command to access global configuration mode.

   ```
   device# configure terminal
   ```

2. Enter the **router isis** command to enter IS-IS router configuration mode and enable IS-IS on the device.

   ```
   device(config)# router isis
   ```

3. Enter the **auth-mode** command with the **md5** and **level-1** parameters to configure MD5 authentication for Level 1 packets.

   ```
   device(config-isis-router)# auth-mode MD5 level-1
   ```

4. Enter the **no auth-check** command with the **level-1** parameter to temporarily disable authentication checking globally for Level 1 packets.

   ```
   device(config-isis-router)# no auth-check level-1
   ```

5. Enter the **auth-key** command with the **level-1** parameters, specifying an authentication password to configure an authentication key globally for Level 1 packets.

   ```
   device(config-isis-router)# auth-key 0 mysecurekey level-1
   ```

The following example configures IS-IS authentication mode, an IS-IS authentication key and temporarily disables IS-IS authentication checking for Level 1 packets.

```
device# configure terminal
device(config)# router isis
device(config-isis-router)# auth-mode MD5 level-1
device(config-isis-router)# no auth-check level-1
device(config-isis-router)# auth-key 0 mysecurekey level-1
```

# Changing the IS-IS level globally

By default, a device can operate as both a Level 1 and Level 2 router. This task globally changes the level supported from Level 1 and Level 2 to Level 1 only.

1. Enter the **configure terminal** command to access global configuration mode.

   ```
   device# configure terminal
   ```

2. Enter the **router isis** command to enter IS-IS router configuration mode and enable IS-IS on the device.

   ```
   device(config)# router isis
   ```

3. Enter the **is-type** command with the **level-1** parameter to globally change the IS-IS level supported from Level-1 and Level-2 to Level-1 only.

   ```
   device(config-isis-router)# is-type level-1
   ```

The following example globally changes the IS-IS level supported from Level-1 and Level-2 to Level-1 only.

```
device# configure terminal
device(config)# router isis
device(config-isis-router)# is-type level-1
```

# Logging adjacency changes

You can configure a device to log changes in the status of an adjacency with another IS. The following task enables the logging of adjacency changes.

1. Enter the **configure terminal** command to access global configuration mode.

   ```
   device# configure terminal
   ```

2. Enter the **router isis** command to enter IS-IS router configuration mode and enable IS-IS on the device.

   ```
   device(config)# router isis
   ```

3. Enter the **log adjacency** command to enable the logging of adjacency changes.

   ```
   device(config-isis-router)# log adjacency
   ```

The following example enables the logging of adjacency changes.

```
device# configure terminal
device(config)# router isis
device(config-isis-router)# log adjacency
```

# Complete Sequence Numbers PDU interval

A Complete Sequence Numbers PDU (CSNP) is a complete list of the LSPs in the Designated IS' link state database. The CSNP contains a list of all the LSPs in the database, as well as other information that helps IS neighbors determine whether their LSP databases are in sync with one another. The Designated IS sends CSNPs to the broadcast interface. Level 1 and Level 2 each have their own Designated IS.

A Partial Sequence Numbers PDU (PSNP) is a partial list of LSPs. ISs other than the Designated IS (that is, the non-Designated ISs) send PSNPs to the broadcast interface. The CSNP interval specifies how often the Designated IS sends a CSNP to the broadcast interface. Likewise, the PSNP interval specifies how often other ISs (non-Designated ISs) send a PSNP to the broadcast interface.

## Configuring the CSNP interval

By default the Complete Sequence Numbers PDU (CSNP) interval is 10 seconds. The following task configures CSNP interval of 25 seconds for Level 1 and Level 2 packets.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **router isis** command to enter IS-IS router configuration mode and enable IS-IS on the device.

```
device(config)# router isis
```

3. Enter the **csnp-interval** command and specify a value to globally configure the CSNP interval.

```
device(config-isis-router)# csnp-interval 25
```

The following example configures a CSNP interval of 25 seconds.

```
device# configure terminal
device(config)# router isis
device(config-isis-router)# csnp-interval 25
```

# Changing the maximum LSP lifetime

This task changes the maximum number of seconds an unrefreshed LSP remains in a device's LSP database from the default of 1200 seconds (20 minutes) to 2400 seconds (40 minutes).

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **router isis** command to enter IS-IS router configuration mode and enable IS-IS on the device.

```
device(config)# router isis
```

3. Enter the **max-lsp-lifetime** command and specify a value to globally configure the maximum LSP lifetime.

```
device(config-isis-router)# max-lsp-lifetime 2400
```

The following example changes the maximum LSP lifetime to 2400 seconds.

```
device# configure terminal
device(config)# router isis
device(config-isis-router)# max-lsp-lifetime 2400
```

# Changing the LSP refresh interval

This task changes the maximum number of seconds a device waits between sending updated LSPs to its IS-IS neighbors from the default of 900 seconds to 1800 seconds.

1. Enter the **configure terminal** command to access global configuration mode.

   ```
   device# configure terminal
   ```

2. Enter the **router isis** command to enter IS-IS router configuration mode and enable IS-IS on the device.

   ```
   device(config)# router isis
   ```

3. Enter the **lsp-refresh-interval** command and specify a value to change the LSP refresh interval.

   ```
   device(config-isis-router)# lsp-refresh-interval 1800
   ```

The following example changes the LSP refresh interval to 1800 seconds.

```
device# configure terminal
device(config)# router isis
device(config-isis-router)# lsp-refresh-interval 1800
```

# Changing the LSP generation interval

This task changes the minimum number of seconds the device waits between sending updated LSPs to its IS-IS neighbors from the default of 10 seconds to 60 seconds (one minute).

1. Enter the **configure terminal** command to access global configuration mode.

   ```
   device# configure terminal
   ```

2. Enter the **router isis** command to enter IS-IS router configuration mode and enable IS-IS on the device.

   ```
   device(config)# router isis
   ```

3. Enter the **lsp-gen-interval** command and specify a value to change the LSP generation interval.

   ```
   device(config-isis-router)# lsp-gen-interval 60
   ```

The following example changes the LSP generation interval to 60 seconds.

```
device# configure terminal
device(config)# router isis
device(config-isis-router)# lsp-gen-interval 60
```

# Changing the LSP interval and retransmit interval

The LSP interval is the rate of transmission, in milliseconds, of the LSPs. The retransmit interval is the time, in seconds, the device waits before it retransmits LSPs. This task changes the LSP interval to 45 milliseconds and changes the retransmission interval to 10 milliseconds.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **router isis** command to enter IS-IS router configuration mode and enable IS-IS on the device.

```
device(config)# router isis
```

3. Enter the **lsp-interval 45** command and specify a value to change the LSP interval from the default of 33 milliseconds.

```
device(config-isis-router)# lsp-interval 42
```

4. Enter the **retransmit-interval** command and specify a value to change the retransmission interval from the default of 5 seconds.

```
device(config-isis-router)# retransmit-interval 10
```

The following example changes the LSP interval to 45 milliseconds and changes the retransmission interval to 10 milliseconds.

```
device# configure terminal
device(config)# router isis
device(config-isis-router)# lsp-interval 42
device(config-isis-router)# retransmit-interval 10
```

# Disabling IS-IS name mapping capability

The implementation of IS-IS supports RFC 2763, which describes a mechanism for mapping IS-IS system IDs to the hostnames of the devices with those IDs. The following task disables IS-IS name mapping.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **router isis** command to enter IS-IS router configuration mode and enable IS-IS on the device.

```
device(config)# router isis
```

3. Enter the **no hostname** command to disable IS-IS name mapping.

```
device(config-isis-router)# no hostname
```

The following example globally disables IS-IS name mapping.

```
device# configure terminal
device(config)# router isis
device(config-isis-router)# no hostname
```

# Logging invalid LSP packets received

You can configure a device to provide logging of invalid LSP packets. The following task enables the logging of invalid LSP packets.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **router isis** command to enter IS-IS router configuration mode and enable IS-IS on the device.

```
device(config)# router isis
```

3. Enter the **log invalid-lsp-packets** command to enable the logging of invalid LSP packets.

```
device(config-isis-router)# log invalid-lsp-packets
```

The following example enables the logging of invalid LSP packets.

```
device# configure terminal
device(config)# router isis
device(config-isis-router)# log invalid-lsp-packets
```

# Changing the SPF timer

Every IS maintains a Shortest Path First (SPF) tree, which is a representation of the states of each of the IS's links to ESs and other ISs. If the IS is both a Level-1 and Level-2 IS, it maintains separate SPF trees for each level. To ensure that the SPF tree remains current, the IS updates the tree at regular intervals following a change in network topology or the link state database. This task changes the maximum interval in seconds between SPF recalculations, the initial SPF calculation delay, and the hold time between the first and second SPF calculation.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **router isis** command to enter IS-IS router configuration mode and enable IS-IS on the device.

```
device(config)# router isis
```

3. Enter the **spf-interval** command with the **level-1** parameter, specifying values for the *max-wait* , *initial-wait* , and *second-wait* parameters, to set the maximum interval in seconds between SPF recalculations, the initial SPF calculation delay, and the hold time between the first and second SPF calculation.

```
device(config-isis-router)# spf-interval level-1 15 10000 15000
```

The following example specifies that the maximum interval in seconds between SPF recalculations is 15 seconds for Level 1 packets. The initial SPF calculation delay is 10000 milliseconds and the hold time between the first and second SPF calculation is 15000 millseconds.

```
device# configure terminal
device(config)# router isis
device(config-isis-router)# spf-interval level-1 15 10000 15000
```

# Configuring the IS-IS flooding mechanism

You can configure IS-IS to flood Link State PDUs to other devices in the network before running SPF, thus improving database synchronization by allowing LSP changes to be propagated to neighbors before running SPF. The following task configures the IS-IS fast-flood feature.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **router isis** command to enter IS-IS router configuration mode and enable IS-IS on the device.

```
device(config)# router isis
```

3. Enter the **fast-flood** command and specify a value to implement IS-IS fast-flood.

```
device(config-isis-router)# fast-flood 10
```

The following example configures IS-IS to flood 10 LSPs before running SPF.

```
device# configure terminal
device(config)# router isis
device(config-isis-router)# fast-flood 10
```

# Configuring IS-IS PSPF exponential back-off

The following task changes the partial shortest path first (PSPF) interval, changing the maximum interval between SPF recalculations, the initial SPF calculation delay, and the hold time between the first and second SPF calculation.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **router isis** command to enter IS-IS router configuration mode and enable IS-IS on the device.

```
device(config)# router isis
```

3. Enter the **partial-spf-interval** command with the **level-1** parameter, specifying values for the *max-wait* , *initial-wait* , and *second-wait* parameters, to set the maximum interval between SPF recalculationss, the initial SPF calculation delay, and the hold time between the first and second SPF calculation.

```
device(config-isis-router)# partial-spf-interval 15 10000 15000
```

The following example specifies that the maximum interval in seconds between SPF recalculations is 15 seconds. The initial SPF calculation delay is 10000 milliseconds and the hold time between the first and second SPF calculation is 15000 milliseconds.

```
device# configure terminal
device(config)# router isis
device(config-isis-router)# partial-spf-interval 15 10000 15000
```

# Hello padding

By default, a device adds extra data to the end of a hello packet to make the packet the same size as the maximum length of PDU the device supports.

The padding applies to the following types of hello packets:

- ES hello (ESH PDU)
- IS hello (ISH PDU)
- IS to IS hello (IIH PDU)

The padding consists of arbitrarily valued octets. A padded hello PDU indicates the largest PDU that the device can receive. Other ISs that receive a padded hello PDU from the device can, therefore, ensure that the IS-IS PDUs they send the device are lower than the maximum value available. Similarly, if the device receives a padded hello PDU from a neighbor IS, the device knows the maximum size PDU that can be sent to the neighbor.

When padding is enabled, the maximum length of a Hello PDU sent by the device is 1514 bytes.

If you need to disable hello padding, you can do so globally or on individual interfaces. Generally, you do not need to disable padding unless a link is experiencing slow performance. If you enable or disable padding on an interface, the interface setting overrides the global setting.

## Disabling hello padding globally

By default, a device adds extra data to the end of a hello packet to make the packet the same size as the maximum length of PDU the device supports. The following task disables hello padding globally.

1. Enter the **configure terminal** command to access global configuration mode.

   ```
   device# configure terminal
   ```

2. Enter the **router isis** command to enter IS-IS router configuration mode and enable IS-IS on the device.

   ```
   device(config)# router isis
   ```

3. Enter the **no hello padding** command to disable hello padding globally.

   ```
   device(config-isis-router)# no hello padding
   ```

The following example disables hello padding globally.

```
device# configure terminal
device(config)# router isis
device(config-isis-router)# no hello padding
```

# Partial SPF optimizations

IS-IS employs certain partial SPF optimizations to make partial changes to the routing table in network change situations where the topology of the network has not changed but where there may be changes in the IP networks advertised.

These optimizations are termed partial SPF optimizations. IS-IS can be configured to perform a full SPF calculation when any network (non-topology) change occurs so that IS-IS always runs full SPF for all such network changes.

NOTE
If you disable the partial SPF optimizations, IS-IS automatically disables the incremental SPF optimizations and always runs full SPF. However, the reverse is not true: disabling incremental SPF optimizations does not disable partial optimizations.

## Disabling partial SPF optimizations

You can configure IS-IS to perform a full SPF calculation when any network (non-topology) change occurs so that IS-IS always runs full SPF for all such network changes. This task disables partial SPF optimizationss.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **router isis** command to enter IS-IS router configuration mode and enable IS-IS on the device.

```
device(config)# router isis
```

3. Enter the **disable-partial-spf-opt** command to disable partial SPF optimizations.

```
device(config-isis-router)# disable-partial-spf-opt
```

The following example disables partial SPF optimizations.

```
device# configure terminal
device(config)# router isis
device(config-isis-router)# disable-partial-spf-opt
```

# Incremental SPF optimizations

In the event of certain topology changes, for instance non-local adjacency flaps, IS-IS employs incremental SPF optimizations to efficiently update the routing table. An incremental SPF is faster and takes fewer CPU cycles than a full SPF.

IS-IS can be configured to perform a full SPF calculation when any network topology change occurs so that IS-IS always runs full SPF for all such network changes.

NOTE
If you disable the partial SPF optimizations, IS-IS automatically disables the incremental SPF optimizations and always runs full SPF. However, the reverse is not true: disabling incremental SPF optimizations does not disable partial optimizations.

## Disabling incremental shortcut SPF optimizations

You can configure IS-IS to perform a full SPF calculation when any network topology change occurs so that IS-IS always runs full SPF for all such network changes. This task disables incremental SPF optimizations so that IS-IS always runs full SPF for all such network changes.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **router isis** command to enter IS-IS router configuration mode and enable IS-IS on the device.

```
device(config)# router isis
```

3. Enter the **disable-partial-spf-opt** command to disable partial SPF optimizations.

```
device(config-isis-router)# disable-incremental-spf-opt
```

The following example disables incremental SPF optimizations.

```
device# configure terminal
device(config)# router isis
device(config-isis-router)# disable-incremental-spf-opt
```

# IS-IS incremental shortcut LSP SPF optimization

IS-IS can be configured to use an incremental shortcut LSP SPF optimization algorithm. Incremental shortcut LSP SPF optimization is more efficient when updating the routes in cases where the shortcut LSP state change does not influence the topology. Incremental Shortcut LSP SPF Optimizations are enabled by default.

NOTE
If you disable the partial SPF optimizations, IS-IS automatically disables the incremental SPF optimizations and always runs full SPF. However, the reverse is not true: disabling incremental SPF optimizations does not disable partial optimizations.

NOTE
Incremental Shortcut SPF optimizations are not applicable to LSP shortcuts with metrics configured on them.

NOTE
Incremental Shortcut SPF optimizations are not applicable to LSP shortcuts with negative relative metrics configured.

NOTE
Incremental Shortcut SPF optimizations are not applicable to announced LSP shortcuts.

## Disabling incremental SPF optimizations

IS-IS is configured to use an incremental shortcut LSP SPF optimization algorithm by default. This task disables incremental shortcut LSP SPF optimization.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **router isis** command to enter IS-IS router configuration mode and enable IS-IS on the device.

```
device(config)# router isis
```

3. Enter the **disable-inc-stct-spf-opt** command to disable incremental shortcut LSP SPF optimization.

```
device(config-isis-router)# disable-inc-stct-spf-opt
```

The following example disables incremental shortcut LSP SPF optimization.

```
device# configure terminal
device(config)# router isis
device(config-isis-router)# disable-inc-stct-spf-opt
```

# Maximum number of load sharing paths

You can change the number of paths IPv4 IS-IS can calculate and install in the IPv4 forwarding table. If you change the number of paths to one, the device does not load share multiple route paths learned from IPv4 IS-IS.

NOTE
The maximum number of paths supported by the BR-MLX-10Gx24-DM module is 16.

By default, IPv4 IS-IS can calculate and install four equal-cost paths into the IPv4 forwarding table.

## Changing the maximum number of load sharing paths

You can specify the number of paths IS-IS can calculate and install in the IP forwarding table. The following task specifies that the number of paths IS-IS can calculate and install in the IP forwarding table is 7.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **router isis** command to enter IS-IS router configuration mode and enable IS-IS on the device.

```
device(config)# router isis
```

3. Enter the **address-family ipv4 unicast** command to enable ISIS address-family IPv4 unicast configuration mode.

```
device(config-isis-router)# address-family ipv4 unicast
```

4. Enter the **maximum-paths** command and specify a value to specify the number of paths IS-IS can calculate and install in the IP forwarding table.

```
device(config-isis-router-ipv4u)# maximum-paths 7
```

The following example specifies that the number of paths IS-IS can calculate and install in the IP forwarding table is 7.

```
device# configure terminal
device(config)# router isis
device(config-isis-router)# address-family ipv4 unicast
device(config-isis-router-ipv4u)# maximum-paths 7
```

# Default route advertisement

By default, a device does not generate or advertise a default route to its neighboring ISs. A default route is not advertised even if the device's IPv4 route table contains a default route. You can enable the device to advertise a default route to all neighboring ISs.

NOTE
This feature requires the presence of a default route in the IPv4 route table.

By default, this feature originates the default route at Level 2 only. However, you can apply a route map to originate the default route to Level 1 only or at both Level 1 and Level 2. The route map must be configured before you can use the route map to configure the device to advertise the default route.

# Enabling advertisement of a default route

You can enable the device to advertise a default route to all neighboring ISs. The following task enables the advertisement of a default route.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **router isis** command to enter IS-IS router configuration mode and enable IS-IS on the device.

```
device(config)# router isis
```

3. Enter the **address-family ipv4 unicast** command to enable ISIS address-family IPv4 unicast configuration mode.

```
device(config-isis-router)# address-family ipv4 unicast
```

4. Enter the **default-information-originate** command to enable the advertisement of a default route.

```
device(config-isis-router-ipv4u)# default-information-originate
```

The following example enables the advertisement of a default route.

```
device# configure terminal
device(config)# router isis
device(config-isis-router)# address-family ipv4 unicast
device(config-isis-router-ipv4u)# default-information-originate
```

# Using a route map to advertise a default route

You can use a route map to configure the device to advertise a default route to all neighboring ISs. The following task enables the advertisement of a default route if the route map "myroutemap" is satisfied.

The route-map "myroutemap" must already be configured.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **router isis** command to enter IS-IS router configuration mode and enable IS-IS on the device.

```
device(config)# router isis
```

3. Enter the **address-family ipv4 unicast** command to enable ISIS address-family IPv4 unicast configuration mode.

```
device(config-isis-router)# address-family ipv4 unicast
```

4. Enter the **default-information-originate** command with the **route-map** parameter and specify a route-map parameter to use the route map to enable the advertisement of a default route.

```
device(config-isis-router-ipv4u)# default-information-originate route-map myroutemap
```

The following example enables the advertisement of a default route if the route map "myroutemap" is satisfied.

```
device# configure terminal
device(config)# router isis
device(config-isis-router)# address-family ipv4 unicast
device(config-isis-router-ipv4u)# default-information-originate route-map myroutemap
```

# IS-IS administrative distance

When the device has paths from multiple routing protocols to the same destination, it compares the administrative distances of the paths and selects the path with the lowest administrative distance to place in the IPv4 route table.

For example, if the device has a path from iBGP, from OSPF, and IPv4 IS-IS to the same destination, and all the paths are using their protocols' default administrative distances, the device selects the OSPF path, because that path has a lower administrative distance than the iBGP and IPv4 IS-IS paths.

The default IPv4 administrative distances on the device are:

- Directly connected - 0 (this value is not configurable)
- Static - 1 (applies to all static routes, including default routes)
- eBGP - 20
- OSPF - 110
- IPv4 IS-IS - 115
- RIP - 120
- iBGP - 200
- Local BGP - 200
- Unknown - 255 (the device will not use this route)

Lower administrative distances are preferred over higher distances. For example, if the device receives routes for the same network from IPv4 IS-IS and from iBGP, it will prefer the IPv4 IS-IS route by default.

## Changing the administrative distance for IPv4 IS-IS

You can change the administrative distance for IPv4 IS-IS routes. The following task changes the administrative distance for the IPv4 IS-IS unicast address family.

1. Enter the **configure terminal** command to access global configuration mode.

   ```
   device# configure terminal
   ```

2. Enter the **router isis** command to enter IS-IS router configuration mode and enable IS-IS on the device.

   ```
   device(config)# router isis
   ```

3. Enter the **address-family ipv4 unicast** command to enable ISIS address-family IPv4 unicast configuration mode.

   ```
   device(config-isis-router)# address-family ipv4 unicast
   ```

4. Enter the **distance** command and specify a value to change the administrative distance for the IPv4 IS-IS unicast address family.

   ```
   device(config-isis-router-ipv4u)# distance 60
   ```

The following example sets an administrative distance of 60 for the IPv4 unicast address family.

```
device# configure terminal
device(config)# router isis
device(config-isis-router)# address-family ipv4 unicast
device(config-isis-router-ipv4u)# distance 60
```

# Configuring summary addresses

You can configure summary addresses to aggregate IS-IS route information. Summary addresses can enhance performance by reducing the size of the Link State database, reducing the amount of data the device needs to send to its neighbors, and reducing the CPU cycles used for IS-IS. The following task configures a summary address of 10.1.0.0 with a mask of 255.255.0.0 for Level 1 and Level 2 routes.

1. Enter the **configure terminal** command to access global configuration mode.

   ```
   device# configure terminal
   ```

2. Enter the **router isis** command to enter IS-IS router configuration mode and enable IS-IS on the device.

   ```
   device(config)# router isis
   ```

3. Enter the **address-family ipv4 unicast** command to enable ISIS address-family IPv4 unicast configuration mode.

   ```
   device(config-isis-router)# address-family ipv4 unicast
   ```

4. Enter the **summary-address** command with the **level-1-2** parameter, specifying an IPv4 address and network mask to configure a summary address and network mask for Level 1 and Level 2 routes.

   ```
   device(config-isis-router-ipv4u)# summary-address 10.1.0.0 255.255.0.0 level-1-2
   ```

The following example configures a summary address of 10.1.0.0 with a mask of 255.255.0.0 for Level 1 and Level 2 routes.

```
device# configure terminal
device(config)# router isis
device(config-isis-router)# address-family ipv4 unicast
device(config-isis-router-ipv4u)# summary-address 10.1.0.0 255.255.0.0 level-1-2
```

# IPv4 IS-IS route redistribution

Routes can be redistributed into IPv4 IS-IS by performing the following configuration tasks:

- Changing the default redistribution metric (optional).
- Configuring the redistribution of a particular route type into IPv4 IS-IS (mandatory).

The device can redistribute routes from the following route sources into IPv4 IS-IS:

- BGP4+
- RIP
- OSPF
- Static IPv4 routes
- IPv4 routes learned from directly connected networks

The device can also can redistribute Level 1 IPv4 IS-IS routes into Level 2 IPv4 IS-IS routes, and Level 2 IPv4 IS-IS routes into Level 1 IPv4 IS-IS routes.

Route redistribution from other sources into IPv4 IS-IS is disabled by default. When you enable redistribution, the device redistributes routes only into Level 2 by default. You can specify Level 1 only, Level 2 only, or Level 1 and Level 2 when you enable redistribution.

The device automatically redistributes Level 1 routes into Level 2 routes. Thus, you do not need to enable this type of redistribution. You also can enable redistribution of Level 2 routes into Level 1 routes.

The device attempts to use the redistributed route's metric as the route's IPv4 IS-IS metric. For example, if an OSPF route has an OSPF cost of 20, the device uses 20 as the route's IPv4 IS-IS metric. The device uses the redistributed route's metric as the IPv4 IS-IS metric unless the route does not a have a valid metric. In this case, the device assigns the default metric value to the route.

An IS-IS LSP can hold around 32,000 IPv4 routes or 11,000 IPv6 routes. This number can vary depending on the prefix length of the routes.

# Redistributing routes into IPv4 IS-IS

Routes can be redistributed for IPv4-IS-IS, and the routes to be redistributed can be specified.

The redistribution of static routes into IPv4 IS-IS is configured on device1. The redistribution of connected routes into IPv4 IS-IS is configured on device2, and the connected routes to be redistributed are specified.

1. On device1, enter the **configure terminal** command to access global configuration mode.

   ```
   device1# configure terminal
   ```

2. Enter the **router isis** command to enter IS-IS router configuration mode and enable IS-IS on the device.

   ```
   device1(config)# router isis
   ```

3. Enter the **address-family ipv4 unicast** command to enable ISIS address-family IPv4 unicast configuration mode.

   ```
   device1(config-isis-router)# address-family ipv4 unicast
   ```

4. Enter the **redistribute** command with the **static** parameter to redistribute static routes.

   ```
   device(1config-isis-router-ipv4u)# redistribute static
   ```

5. On device2, enter the **configure terminal** command to access global configuration mode.

   ```
   device2# configure terminal
   ```

6. Enter the **router isis** command to enter IS-IS router configuration mode and enable IS-IS on the device.

   ```
   device2(config)# router isis
   ```

7. Enter the **address-family ipv4 unicast** command to enable ISIS address-family IPv4 unicast configuration mode.

   ```
   device2(config-isis-router)# address-family ipv4 unicast
   ```

8. Enter the **redistribute** command with the **connected** and **route-map** parameters to redistribute connected routes and specify a route map.

   ```
   device2(config-isis-router-ipv4u)# redistribute connected route-map rmap1
   ```

The following example redistributes static routes into IPv4 IS-IS on a device.

```
device# configure terminal
device(config)# router isis
device(config-isis-router)# address-family ipv4 unicast
device(config-isis-router-ipv4u)# redistribute static
```

The following example redistributes connected routes into IPv4 IS-IS on a device and specifies a route map.

```
device# configure terminal
device(config)# router isis
device(config-isis-router)# address-family ipv4 unicast
device(config-isis-router-ipv4u)# redistribute connected route-map rmap1
```

# Default redistribution metric

When IPv4 IS-IS redistributes a route from another route source (such as OSPF, BGP4+, or a static IPv4 route) into IPv4 IS-IS, the route's metric value is used as its metric when the metric is not modified by a route map or metric parameter and the default redistribution metric is set to its default value of 0.

The default metric value can be changed.

NOTE
The implementation of IS-IS does not support the optional metric types Delay, Expense, or Error.

## Changing the default redistribution metric

You can change the default metric for the IS-IS routing protocol. The following task changes the default metric value for the IPv4 unicast address family.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **router isis** command to enter IS-IS router configuration mode and enable IS-IS on the device.

```
device(config)# router isis
```

3. Enter the **address-family ipv4 unicast** command to enable ISIS address-family IPv4 unicast configuration mode.

```
device(config-isis-router)# address-family ipv4 unicast
```

4. Enter the **default-metric** command and specify a value to change the default metric value for the IPv4 unicast address family.

```
device(config-isis-router-ipv4u)# default-metric 30
```

The following example sets the default value to 30 for the IPv4 unicast address family.

```
device# configure terminal
device(config)# router isis
device(config-isis-router)# address-family ipv4 unicast
device(config-isis-router-ipv4u)# default-metric 30
```

# Configuring the default link metric value globally

You can configure the metric value globally on all active IS-IS interfaces for the configured address family. The following task sets the IS-IS default link metric value for Level 1 for the IPv4 address family.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **router isis** command to enter IS-IS router configuration mode and enable IS-IS on the device.

```
device(config)# router isis
```

3. Enter the **address-family ipv4 unicast** command to enable ISIS address-family IPv4 unicast configuration mode.

```
device(config-isis-router)# address-family ipv4 unicast
```

4. Enter the **default-link-metric** command with the **level-1** parameter, and specify a value, to set the IS-IS default link metric value for Level 1 for the IPv4 address family.

```
device(config-isis-router-ipv4u)# default-link-metric 30 level-1
```

The following example sets the IS-IS default link metric value for Level 1 for the IPv4 address family.

```
device# configure terminal
device(config)# router isis
device(config-isis-router)# address-family ipv4 unicast
device(config-isis-router-ipv4u)# default-link-metric 30 level-1
```

# IS-IS metric styles

There are two types of metric styles in IS-IS:

- narrow metric
- wide metric

The range of the metric value is different for both of these styles. If there is a change in the metric-style configuration, the default-link-metric changes with it so that the new value of the default-link-metric is equal to the minimum of both the configured value and the maximum value supported for the new metric-style.

If the metric style changes from narrow metric to wide metric, no change in the value of default-link-metric occurs.

If the metric style changes from wide metric to narrow metric, and if the value of default-link-metric is greater than 63, the default-link-metric takes the value 63, as it is the maximum supported in the narrow metric.

## Changing the metric style

You can increase the range of metric values supported by the device by changing the metric style to wide. The following task enables the wide metric type for Level 2 packets for the IS-IS IPv4 unicast address-family.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **router isis** command to enter IS-IS router configuration mode and enable IS-IS on the device.

```
device(config)# router isis
```

3. Enter the **address-family ipv4 unicast** command to enable ISIS address-family IPv4 unicast configuration mode.

```
device(config-isis-router)# address-family ipv4 unicast
```

4. Enter the **metric-style wide** command to enable the wide metric type for Level 2 packets for the IS-IS IPv4 unicast address-family.

```
device(config-isis-router-ipv4u)# metric-style wide level-2
```

The following example enables the wide metric type for Level 2 packets for the IS-IS IPv4 unicast address-family.

```
device# configure terminal
device(config)# router isis
device(config-isis-router)# address-family ipv4 unicast
device(config-isis-router-ipv4u)# metric-style wide level-2
```

# IS-IS non-stop routing

IS-IS can continue operation without interruption during a processor switchover or hitless-reload event when the Non-Stop Routing (NSR) feature is enabled.

In general, a device restart causes its peer to remove the routes originated from the device and reinstalls them. The IS-IS NSR feature enables the device to maintain neighbors and LSA database with its peer in the event of a restart. In IS-IS NSR, the processor switchovers and the hitless-reloads are treated the same as they are during startup and the overload bit is set in the same way as it is after a reboot.

NOTE
IS-IS NSR is not supported for the CES 2000 Series and CER 2000 Series platforms.

NOTE
IS-IS NSR is independent of Graceful Restart (GR) and GR help role mechanisms.

## Enabling non-stop routing

IS-IS non-stop routing (NSR) can be re-enabled if it has been disabled. The following task re-enables NSR for IS-IS.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **router isis** command to enter IS-IS router configuration mode and enable IS-IS on the device.

```
device(config)# router isis
```

3. Enter the **nonstop-routing** command to re-enable NSR.

```
device(config-isis-router)# nonstop-routing
```

The following example re-enables NSR.

```
device# configure terminal
device(config)# router isis
device(config-isis-router)# nonstop-routing
```

# Limitations of IS-IS non-stop routing

The following limitations apply for IS-IS non-stop routing (NSR)

- The IS-IS over GRE tunnel feature does not support IS-IS NSR. The GRE tunnel interface types are not supported.
- The IS-IS shortcuts are not supported because they depend on the MPLS tunnel.
- If the IS-IS hellos are forwarded at Layer 2 and the device executes a hitless-reload, hellos will not be forwarded for a brief time. The IS-IS adjacencies are lost for 12 seconds and there will be data traffic loss.
- The configuration events that occur close to switchover or hitless-reload may get lost due to CLI synchronization issues.
- The neighbor or interface state changes close to switchover or hitless-reload cannot be handled.
- The IS-IS neighbor hold timer is restarted upon IS-IS NSR switchover or hitless-reload.
- It is recommended to use the default IS-IS hello timer value. IS-IS neighbor sessions may flap during a linecard software upgrade if a shorter timer value is used.

- The traffic counters are not synchronized because the neighbor and LSP database counters are recalculated on the standby module during synchronization.
- With IS-IS NSR enabled, after switchover or hitless-reload to standby MP, IS-IS routes, LSP database and neighbor adjacencies are maintained so that there will be no loss of existing traffic to the IS-IS destinations.
- The IS-IS NSR hitless failover event may not be completely invisible to the network because, after switchover, additional flooding of CSNP packets will occur in the directly connected neighbors.

## DIS hello interval

The DIS hello interval value is derived from the hello interval configured under the interface.

The default IS-IS hello interval is 10 sec. The default DIS hello interval is 10/3 = sec. The default value of the DIS hello interval is not changed. However, if you configure a hello interval of 20 for an interface, then the DIS hello interval for the interface becomes 20/3 =6 sec.

The DIS hello multipler is the same as the hello multipler configured under the interface.

# Enabling IS-IS for an Interface

IS-IS can be enabled for a specified interface for a device. Several commands can then be accessed that allow the configuration of IS-IS on the specified interface. This task enables IS-IS routing for an interface Ethernet.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **interface ethernet** command and specify an interface.

```
device(config)# interface ethernet 1/2
```

3. Enter the **ip router isis** command to enable IS-IS for the interface.

```
device(config-if-e1000-1/2)# ip router isis
```

The following example enables IS-IS routing for an interface Ethernet.

```
device# configure terminal
device(config)# interface ethernet 1/2
device(config-if-e1000-1/2)# ip router isis
```

# Configuring authentication on an IS-IS interface

The following task configures IS-IS authentication mode, an IS-IS authentication key, and temporarily disables IS-IS authentication checking for Level 2 packets on an IS-IS Ethernet interface.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **interface ethernet** command and specify an interface.

```
device(config)# interface ethernet 1/2
```

3. Enter the **isis auth-mode** command with the **md5** and **level-2** parameters to configure MD5 authentication for Level 2 packets for the IS-IS interface.

```
device(config-if-e1000-1/2)# isis auth-mode MD5 level-2
```

4. Enter the **isis auth-key** command with the **level-2** parameter, specifying an authentication password, to configure an authentication key for Level 2 packets for the IS-IS interface.

```
device(config-if-e1000-1/2)# isis auth-key 0 mykey level-2
```

5. Enter the **no isis auth-check** command to temporarily disable authentication checking for Level 1 and Level 2 packets for the IS-IS interface.

```
device(config-if-e1000-1/2)# no isis auth-check
```

The following example configures IS-IS authentication mode, an IS-IS authentication key and temporarily disables IS-IS authentication checking for a specified IS-IS interface.

```
device# configure terminal
device(config)# interface ethernet 1/2
device(config-if-e1000-1/2)# isis auth-mode MD5 level-2
device(config-if-e1000-1/2)# isis auth-key 0 mykey level-2
device(config-if-e1000-1/2)# no isis auth-check
```

# Disabling hello padding for an IS-IS interface

By default, a device adds extra data to the end of a hello packet to make the packet the same size as the maximum length of PDU the device supports. The following task disables hello padding for an IS-IS Ethernet interface.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **interface ethernet** command and specify an interface.

```
device(config)# interface ethernet 1/2
```

3. Enter the **no isis hello padding** command disable hello padding for the IS-IS Ethernet interface.

```
device(config-if-e1000-1/2)# no isis hello padding
```

The following example disables hello padding for a specified IS-IS Ethernet interface.

```
device# configure terminal
device(config)# interface ethernet 1/2
device(config-if-e1000-1/2)# no isis hello padding
```

# Changing the IS-IS level on an IS-IS interface

By default, a device can operate as both a Level 1 and Level 2 router. The following task changes the level supported from Level 1 and Level 2 to Level 1 for an IS-IS interface.

If you change the IS-IS type on an individual interface, the type you specify must also be specified globally. For example, if you globally set the type to Level 2 only, you cannot set the type on an individual interface to Level 1. The setting is accepted but does not take effect.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **interface ethernet** command and specify an interface.

```
device(config)# interface ethernet 1/2
```

3. Enter the **isis circut-type** command with the **level-1** parameter to configure a Level 1 adjacency for the interface.

```
device(config-if-e1000-1/2)# isis circut-type level-1
device(conf-if-eth-1/2)# isis circut-type level-1
```

The following example configures a Level 1 adjacency on an IS-IS Ethernet interface.

```
device# configure terminal
device(config)# interface ethernet 1/2
device(config-if-e1000-1/2)# isis circut-type level-1
```

# Changing the hello multiplier for an IS-IS interface

The hello multiplier is the number by which an IS-IS interface multiplies the hello interval to obtain the hold time for Level 1 and Level 2 IS-to-IS hello PDUs. The following task changes the hello multiplier for Level 2 packets for an Ethernet interface.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **interface ethernet** command and specify an interface.

```
device(config)# interface ethernet 1/2
```

3. Enter the **isis hello-mulitplier** command with the **level-2** parameter, and specify a value, to change the hello multiplier for Level 2 packets for the interface.

```
device(config-if-e1000-1/2)# isis hello-mulitplier 20 level-2
```

The following example changes the hello multiplier to 20 for Level 2 packets for an Ethernet interface.

```
device# configure terminal
device(config)# interface ethernet 1/2
device(config-if-e1000-1/2)# isis hello-mulitplier 20 level-2
```

# Changing the hello interval for an IS-IS interface

The hello interval specifies how often an IS-IS interface sends hello messages to its IS-IS neighbors. The following task changes the hello interval for Level 1 packets to 20 for an IS-IS interface.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **interface ethernet** command and specify an interface.

```
device(config)# interface ethernet 1/2
```

3. Enter the **isis hello-interval** command with the **level-1** parameter, and specify a value, to change the hello interval for Level 1 packets for the interface.

```
device(config-if-e1000-1/2)# isis hello-interval 20 level-1
```

The following example changes the hello interval for Level 1 packets to 20 on a Loopback interface.

```
device# configure terminal
device(config)# interface ethernet 1/2
device(config-if-e1000-1/2)# isis hello-interval 20 level-1
```

# IS-IS point-to-point over Ethernet

IS-IS uses its neighbor's MAC address to form an adjacency and stores the neighbors MAC address to recognize the adjacency in the future. This causes no problems with directly adjacent devices but problems can occur when adjacency is required between devices that are more than one hop away. IS-IS Point-to-Point over Ethernet accommodates an IS-IS network with this type of configuration.

When IS-IS Point-to-Point over Ethernet is used, devices that are several hops away or available through an IP GRE tunnel can form an IS-IS adjacency. IS-IS Point-to-Point over Ethernet can also be used when only two IS's are part of the broadcast network. IS-IS Point-to-Point over Ethernet is configured at the interface level of the devices that are forming an adjacency.The figure below shows two devices several hops away from each other that are configured for IS-IS adjacency.

**FIGURE 45** IS-IS Point-to-Point configuration

# Enabling IS-IS point-to-point over Ethernet

IS-IS Point-to-Point over Ethernet can be configured so that devices that are several hops away can form an IS-IS adjacency. The following task configures two devices, Device A and Device B, for IS-IS point-to-point over Ethernet.

1. On Device A, enter the **configure terminal** command to access global configuration mode.

```
deviceA# configure terminal
```

2. Enter the **interface ethernet** command and specify an interface.

```
deviceA(config)# interface ethernet 1/2
```

3. Enter the **ip router isis** command to enable IS-IS for the interface.

```
deviceA(config-if-e1000-1/2)# ip router isis
```

4. Enter the **ip address** command and specify an IP address and mask to configure a primary IP address for the interface.

```
deviceA(config-if-e1000-1/2)# ip address 10.10.1.1/31
```

5. Enter the **isis point-to-point** command to configure IS-IS point-to-point for the Ethernet interface.

```
deviceA(config-if-e1000-1/2)# isis point-to-point
```

6. On Device B, enter the **configure terminal** command to access global configuration mode.

```
deviceB# configure terminal
```

7. Enter the **interface ethernet** command and specify an interface.

```
deviceB(config)# interface ethernet 2/1
```

8. Enter the **ip router isis** command to enable IS-IS for the interface.

```
deviceB(config-if-e1000-2/1)# ip router isis
```

9. Enter the **ip address** command and specify an IP address and mask to configure a primary IP address for the interface.

```
deviceB(config-if-e1000-2/1)# ip address 10.10.1.2/31
```

10. Enter the **isis point-to-point** command to configure IS-IS point-to-point for the Ethernet interface.

```
deviceB(config-if-e1000-2/1)# isis point-to-point
```

The following example configures two devices, Device A and Device B, for IS-IS point-to-point over Ethernet.

```
deviceA# configure terminal
deviceA(config)# interface ethernet 1/2
deviceA(config-if-e1000-1/2)# ip router isis
deviceA(config-if-e1000-1/2)# ip address 10.10.1.1/31
deviceA(config-if-e1000-1/2)# isis point-to-point


deviceB# configure terminal
deviceB(config)# interface ethernet 2/1
deviceB(config-if-e1000-2/1)# ip router isis
deviceB(config-if-e1000-2/1)# ip address 10.10.1.2/31
deviceB(config-if-e1000-2/1)# isis point-to-point
```

# IS-IS over a GRE IP tunnel

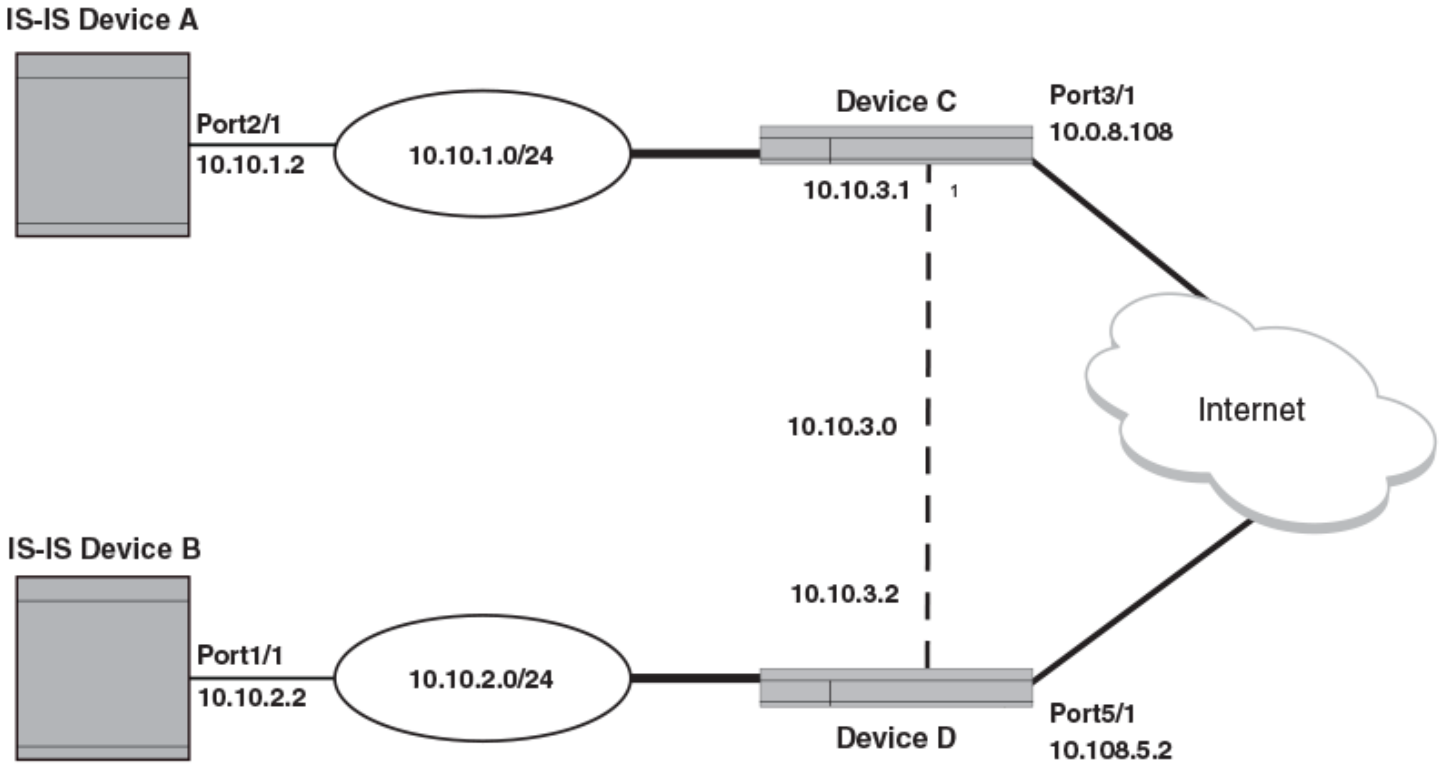
IS-IS adjacency can be established over Ethernet between devices that are more than one hop away using the IS-IS Point-to-Point feature. IS-IS over a GRE IP tunnel extends this capability by allowing you to configure IS-IS adjacency between devices on either end of a GRE IP tunnel.

To configure IS-IS over a GRE IP Tunnel you must configure the following:

- Configure the devices that you want to establish adjacency for IS-IS point-to-point.
- Configure a GRE IP Tunnel.
- Configure the devices used for the GRE IP Tunnel for IS-IS using the**router isis** command.
- Configure the tunnel interfaces on the devices used for the GRE IP Tunnel for IS-IS point-to-point using the **isis point-to-point** command.

The figure below displays a network configured for IS-IS over a GRE IP tunnel. In the example, IS-IS Device A and IS-IS Device B are configured for adjacency. Devices C and D are configured with a GRE IP tunnel.

**FIGURE 46** IS-IS over a GRE IP tunnel




The following examples describe the configurations that support IS-IS over a GRE IP tunnel for each of the routers in Figure 46.

# Configuration considerations for IS-IS over a GRE IP tunnel

Consider the following when configuring IS-IS over a GRE IP tunnel

- When a GRE tunnel is configured, you cannot configure the same routing protocol on the tunnel through which the device learns the route to the tunnel destination. For example, if a device learns the tunnel destination route through the OSPF protocol, you cannot configure the OSPF protocol on the same tunnel and vice-versa. When a tunnel has OSPF configured, the device cannot learn the tunnel destination route through OSPF. This will cause the system to become unstable.
- When you have keepalive configured on both sides of a GRE tunnel, it is recommended that you disable the tunnel before changing any tunnel configurations. You can then re-enable the tunnel to restore it to normal functionality.
- When configuring a GRE IP Tunnel, the device must be configured with one of the following CAM Profiles: ipv4, ipv6, mpls-l3vpn, ipv4-vpn, multi-service-2 or mpls-l3vpn-2.

# Configuring IS-IS over a GRE IP tunnel

IS-IS can be configured over a GRE IP tunnel. The following task configures IS-IS adjacency between two devices, Device C and Device D, on either end of GRE IP tunnel.

The devices should be configured for IS-IS point-to-point as outlined in the "*Enabling IS-IS point-to-point over Ethernet*" section.

1. On Device C, enter the **configure terminal** command to access global configuration mode.

   ```
   deviceC# configure terminal
   ```

2. Enter the **interface tunnel** command and specify a tunnel number.

   ```
   deviceC(config)# interface tunnel 1
   ```

3. Enter the **tunnel source** command and specify an IP address to configure the source IP address for the tunnel interface.

   ```
   deviceC(config-tnif-1)# tunnel source 10.0.8.108
   ```

4. Enter the **tunnel destination** command and specify an IP address to configure the destination source IP address for the tunnel interface.

   ```
   deviceC(config-tnif-1)# tunnel destination 10.108.5.2
   ```

5. Enter the **tunnel mode gre ip** command to enable generic routing encapsulation (GRE) over on a tunnel interface and specifies that the tunneling protocol is IPv4.

   ```
   deviceC(config-tnif-1)# tunnel mode gre ip
   ```

6. Enter the **isis point-to-point** command to configure IS-IS point-to-point for the tunnel interface.

   ```
   deviceC(config-tnif-1)# isis point-to-point
   ```

7. Enter the **ip address** command and specify an IP address to configure a primary IP address for the tunnel interface.

   ```
   deviceC(config-tnif-1)ip address 10.10.3.1/24
   ```

8. Enter the **exit** command.

   ```
   deviceC(config-tnif-1)exit
   ```

9. Enter the **ip route** command and specify a IP address and the IP address of the next hop to to configure a static route with a specified next-hop gateway.

```
deviceC(config) ip route 10.10.2.0/24 10.10.3.1
```

10. On Device D, enter the **configure terminal** command to access global configuration mode.

```
deviceC# configure terminal
```

11. Enter the **interface tunnel** command and specify a tunnel number.

```
deviceD(config)# interface tunnel 1
```

12. Enter the **tunnel source** command with the ethernet parameter and specify an Ethernet interface to configure the source interface for the tunnel interface.

```
deviceD(config-tnif-1)# tunnel source ethernet 5/1
```

13. Enter the **tunnel destination** command and specify an IP address to configure the destination source IP address for the tunnel interface.

```
deviceD(config-tnif-1)# tunnel destination 10.0.8.108
```

14. Enter the **tunnel mode gre ip** command to enable generic routing encapsulation (GRE) over on a tunnel interface and specifies that the tunneling protocol is IPv4.

```
deviceD(config-tnif-1)# tunnel mode gre ip
```

15. Enter the **isis point-to-point** command to configure IS-IS point-to-point for the tunnel interface.

```
deviceD(config-tnif-1)# isis point-to-point
```

16. Enter the **ip address** command and specify an IP address to configure a primary IP address for the tunnel interface.

```
deviceD(config-tnif-1)ip address 10.10.3.2/24
```

17. Enter the **exit** command.

```
deviceD(config-tnif-1)exit
```

18. Enter the **ip route** command and specify a IP address and the IP address of the next hop to to configure a static route with a specified next-hop gateway.

```
deviceD(config) ip route 10.10.1.0/24 10.10.3.1
```

The following example configures IS-IS adjacency between two devices, Device C and Device D, on either end of GRE IP tunnel.

```
deviceC# configure terminal
deviceC(config)# interface tunnel 1
deviceC(config-tnif-1)tunnel source 10.0.8.108
deviceC(config-tnif-1)tunnel destination 10.108.5.2
deviceC(config-tnif-1)tunnel mode gre ip
deviceC(config-tnif-1)isis point-to-point
deviceC(config-tnif-1)ip address 10.10.3.1/24
deviceC(config-tnif-1)exit
deviceC(config) ip route 10.10.2.0/24 10.10.3.1


deviceD# configure terminal
deviceD(config)# interface tunnel 1
deviceD(config-tnif-1) tunnel source ethernet 5/1
deviceD(config-tnif-1) tunnel destination 10.0.8.108
deviceD(config-tnif-1) tunnel mode gre ip
deviceD(config-tnif-1) isis point-to-point
deviceD(config-tnif-1) ip address 10.10.3.2/24
deviceD(config-tnif-1) exit
deviceD(config) ip route 10.10.1.0/24 10.10.3.1
```

# Matching based on IS-IS protocol type

You can configure a route map that allows IS-IS routes to be matched based on level 1 or level 2 or all IS-IS routes.

1. Enter the **configure terminal** command to access global configuration mode.

   ```
   device# configure terminal
   ```

2. Enter the **route-map** command using the **permit** parameter and specifying a route map name to create a route map instance and allow a matching pattern.

   ```
   device(config)# route-map myroutemap permit 10
   ```

3. Enter the **match** command with the **protocol**, **isis** and **level-1** parameters to configure the route map to match on IS-IS level 1 routes.

   ```
   device(config-routemap myroutemap)# match protocol isis level-1
   ```

The following example configures a route map instance that matches on IS-IS level 1 routes.

```
device# configure terminal
device(config)# route-map myroutemap permit 10
device(config-routemap myroutemap)# match protocol isis level-1
```

# Displaying IS-IS statistics

Various **show isis** commands verify information about IPv4 and IPv6 IS-IS configurations.

Use one or more of the following commands to verify IS-IS information. The commands do not have to be entered in this order.

1. Enter the **show isis** command.

```
device# show isis

IS-IS Routing Protocol Operation State: Enabled
IS-Type: Level-1-2
System Id: 768e.f805.5812
Manual area address(es): 11
Level-1-2 Database State: On
Administrative Distance 115
Maximum Paths 8
Default redistribution metric 0
Default link metric for level-1 0 (conf)/ 10 (adv)
Default link metric for level-2 0 (conf)/ 10 (adv)
Protocol Routes Redistributed into IS-IS: None
Number of Routes Redistributed into IS-IS: 0
Level-1 Auth-mode: None
Level-2 Auth-mode: None
Metric Style Supported for Level-1: Narrow
Metric Style Supported for Level-2: Narrow
Graceful-Restart Helper Support: Enabled
ISIS Partial SPF Optimizations: Enabled
Timers:
L1 SPF: Max-wait 5s Init-wait 5000ms Second-wait 5000ms
L2 SPF: Max-wait 5s Init-wait 5000ms Second-wait 5000ms
L1 SPF is not scheduled
L2 SPF is not scheduled
PSPF: Max-wait 5000ms Init-wait 2000ms Second-wait 5000ms
PSPF is not scheduled
LSP: max-lifetime 1200s refresh-interval 900s gen-interval 10s
retransmit-interval 5s, lsp-interval 33ms
SNP: csnp-interval 10s psnp-interval 2s
Global Hello Padding: Enabled
Global Hello Padding For Point to Point Circuits: Enabled
Ptpt Three Way HandShake Mechanism: Enabled
BGP Ipv4 Converged: False BGP Ipv6 Converged: False
IS-IS Traffic Engineering Support: Disabled
No ISIS Shortcuts Configured
BFD: Disabled, BFD HoldoverInterval: 0
NSR: Disabled
LSP-SYNC: Not Globally Enabled
```

   This example output displays general IS-IS information..

2. Enter the **show isis config** command.

```
device# show isis config

router isis
net 11.768e.f805.5812.00
address-family ipv4 unicast
!
address-family ipv6 unicast
!
```

   This example shows the global IS-IS configuration commands that are in effect on the device..

3. Enter the **show isis counts** command.

```
device# show isis counts

Area Mismatch: 0
Max Area Mismatch: 0
System ID Length Mismatch: 0
LSP Sequence Number Skipped: 0
LSP Max Sequence Number Exceeded: 0
Level-1 Database Overload: 1
Level-2 Database Overload: 0
Our LSP Purged: 2
```

This example shows IS-IS error statistics..

4. Enter the **show isis hostname** command.

```
device# show isis hostname

Total number of entries in IS-IS Hostname Table: 1
System ID Hostname * = local IS
--------------------------------------------------
* 768e.f805.5812 R1
```

This example shows the router-name-to-system-ID mapping table entries for the device.

5. Enter the **show isis interface** command.

```
device# show isis interface

Total number of IS-IS Interfaces: 11
Interface: Ve 301
Circuit State: UP Circuit Mode: Level 1-2
Circuit Type: BCAST Passive State: FALSE
Circuit Number: 2, MTU: 1500
Level-1 Auth-mode: NONE
Level-2 Auth-mode: NONE
Level-1 Metric: 10, Level-1 Priority: 64
Level-1 Hello Interval: 10, Level-1 Hello Multiplier: 3
Level-1 Designated IS: R1-02 Level-1 DIS Changes: 2
Level-2 Metric: 10, Level-2 Priority: 64
Level-2 Hello Interval: 10, Level-2 Hello Multiplier: 3
Level-2 Designated IS: R1-02 Level-2 DIS Changes: 2
Next IS-IS LAN Level-2 Hello in 11 seconds
Number of active Level-2 adjacencies: 0
Next IS-IS LAN Level-1 Hello in 1 seconds
Number of active Level-1 adjacencies: 0
Circuit State Changes: 1 Circuit Adjacencies State Changes: 0
Rejected Adjacencies: 0
Circuit Authentication L1 failures: 0
Circuit Authentication L2 failures: 0
Bad LSPs: 0
Control Messages Sent: 7577 Control Messages Received: 0
Hello Padding: Enabled
IP Enabled: TRUE
IP Addresses:
11.2.1.1/30
IPv6 Enabled: FALSE
MPLS TE Enabled: FALSE
BFD Enabled: FALSE
LDP-SYNC: Disabled, State:
…
```

This example shows information about IS-IS interfaces for a device.

6. Enter the **show isis interface** command, using the **brief** keyword.

```
device# show isis interface brief

Total number of IS-IS Interfaces: 11
Interface Type State Mode Passive MTU UpAdj DIS StateChg AdjStateChg
Ve 301 BCAST UP L12 FALSE 1500 0 None 1 0
Ve 302 BCAST UP L12 FALSE 1500 0 None 1 0
Ve 303 BCAST UP L12 FALSE 1500 0 None 1 0
Ve 304 BCAST UP L12 FALSE 1500 0 None 1 0
Ve 305 BCAST UP L12 FALSE 1500 0 None 1 0
Ve 306 BCAST UP L12 FALSE 1500 0 None 1 0
Ve 307 BCAST UP L12 FALSE 1500 0 None 1 0
Ve 308 BCAST UP L12 FALSE 1500 0 None 1 0
Ve 309 BCAST UP L12 FALSE 1500 0 None 1 0
Ve 310 BCAST UP L12 FALSE 1500 0 None 1 0
Lo 1 BCAST UP L12 TRUE 0 0 None 1 0
```

This example shows summarized information about IS-IS interfaces for a device.

7. Enter the **show isis neighbor** command.

```
device# show isis neighbor
```

This example shows IS-IS neighbor information.

8. Enter the **show isis spf-log** command.

```
device# show isis spf-log

ISIS Level-1 SPF Log
When Duration Nodes Count Last-Trigger-LSP Trigger
10h34m13s 0ms 1 10 R1.00-00 Interface State Change
10h34m38s 0ms 1 2 R1.00-00 Interface Config Change
10h34m43s 0ms 1 18 R1.00-00 Interface Config Change
10h34m48s 0ms 1 5 R1.00-00 Interface State Change
ISIS Level-2 SPF Log
When Duration Nodes Count Last-Trigger-LSP Trigger
10h34m13s 0ms 1 10 R1.00-00 Interface State Change
10h34m38s 0ms 1 2 R1.00-00 Interface Config Change
10h34m43s 0ms 1 18 R1.00-00 Interface Config Change
10h34m48s 0ms 1 5 R1.00-00 Interface State Change
```

This example shows IS-IS link-state packet (LSP) logging information.

9. Enter the **show isis traffic** command.

```
show isis traffic

                            Message Received     Message Sent
 Level-1 Hellos             0                    44912
 Level-2 Hellos             0                    44927
 PTP Hellos                 0                    0
 Level-1 LSP                0                    0
 Level-2 LSP                0                    0
 Level-1 CSNP               0                    0
 Level-2 CSNP               0                    0
 Level-1 PSNP               0                    0
 Level-2 PSNP               0                    0
```

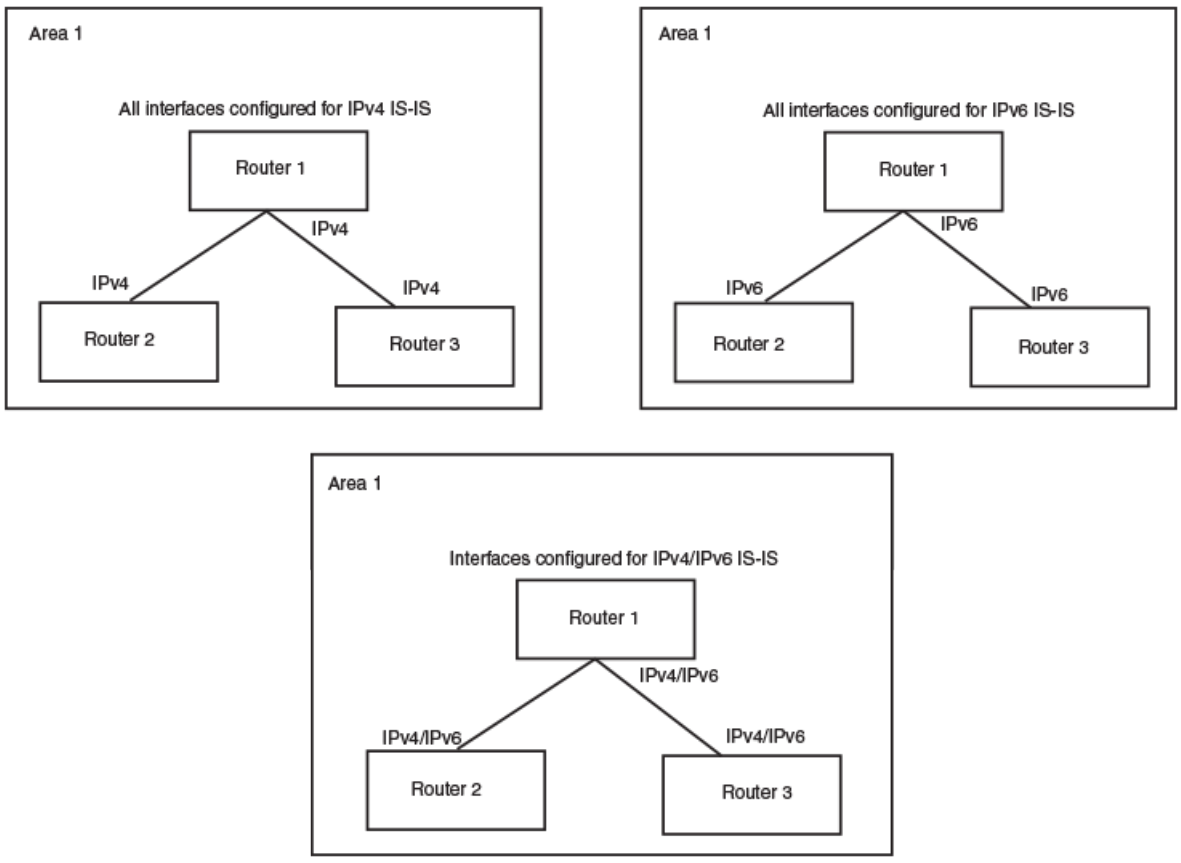
This example shows information about IS-IS packet counts..

# IS-IS (IPv6)

# IPv6 IS-IS single-topology mode

IPv6 IS-IS supports single-topology mode, which means that you can run IPv6 IS-IS concurrently with other network protocols such as IPv4 IS-IS throughout a topology. However, when implementing a single topology, all routers in an area (Level 1 routing) or domain (Level 2 routing) must be configured with the same set of network protocols on all its interfaces. You can configure IPv4 IS-IS only, IPv6 IS-IS only, or both IPv4 IS-IS and IPv6 IS-IS. For example, to successfully implement both IPv4 and IPv6 IS-IS in an area, you must configure both IPv4 and IPv6 IS-IS on all router interfaces in the area.

**FIGURE 47** IPv6 IS-IS in single-topology mode




A single shortest path first (SPF) per level computes the IPv4 and IPv6 routes. The use of a single SPF indicates that both IPv4 and IPv6 IS-IS routing protocols must share a common network topology

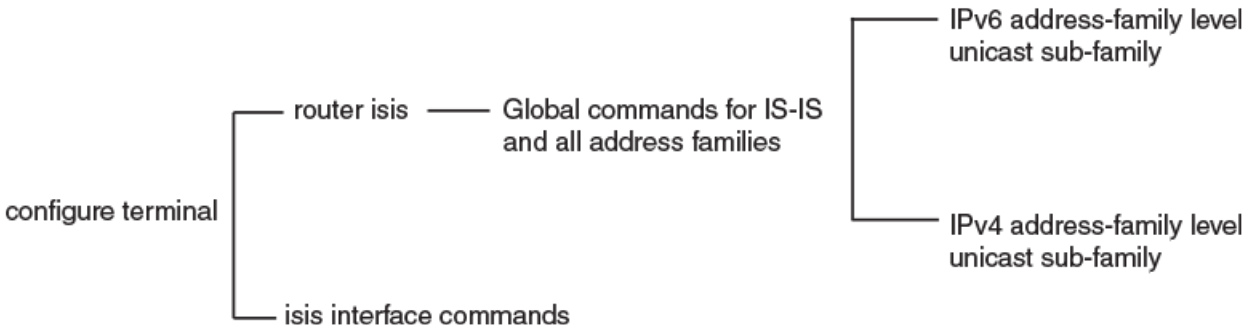
The implementation of IPv4 IS-IS supports type, length, and value (TLV) parameters to advertise reachability to IPv4 networks. The TLVs specify the types of data, the length of the data, and the valid values for the data. IPv6 IS-IS advertises its information using new TLV parameters. The new TLV parameters for IPv6 support an extended default metric value.

In a single topology, if both IPv4 and IPv6 are configured on an interface, metric-style must be set to wide in both address families. Narrow is the default for IPv4. Wide is the default for IPv6.

# IS-IS CLI levels

The CLI includes various levels of commands for IS-IS. The figure below shows these levels, including the levels used for IPv6 IS-IS.

**FIGURE 48** IPv6 IS-IS CLI levels




The IPv6 IS-IS CLI levels are as follows:

- A global level for the configuration of the IS-IS protocol. At this level, all IS-IS configurations at this level apply to IPv4 and IPv6. Enter this layer using the **router isis** command.
  - Under the global level, you specify an address family. Address families separate the IS-IS configuration IPv6 and IPv4. You enter this level by entering the **address-family** command at the router IS-IS level.
  - Under the address family level, you select a sub-address family, which is the type of routes for the configuration. For IS-IS, you specify **unicast**.
  - An interface level.

The implementation of IPv6 IS-IS includes a new configuration level: ISIS address-family IPv6 unicast configuration mode. You enter IS-IS definitions for IPv6 IS-IS under this level. Address-family allows you to create configurations for IPv6 IS-IS unicast routes that are separate and distinct from configurations for IPv4 IS-IS unicast routes.

NOTE

Each address family configuration level allows you to access commands that apply to that particular address family only. To enable a feature in a particular address family, you must specify any associated commands for that feature in that particular address family. A feature configured in IS-IS IPv4 unicast address family configuration mode does not work for the IPv6 IS-IS unicast address family unless it is explicitly configured in ISIS address-family IPv6 unicast configuration mode. .

# Enabling IPv6 IS-IS globally

When IS-IS is enabled on a device, the device enters IS-IS router configuration mode. Several commands can then be accessed that allow the configuration of IS-IS.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **ipv6 unicast-routing** command to enable the forwarding of IPv6 traffic on the device.

```
device(config)# ipv6 unicast-routing
```

3. Enter the **router isis** command to enter IS-IS router configuration mode and enable IS-IS on the device.

```
device(config)# router isis
ISIS: Please configure NET!
```

4. Enter the **net** command and specify a NSAP address to configure a NET for IS-IS.

```
device(config-isis-router)# net 49.2211.0000.00bb.cccc.00
```

The following example enables IPv6 IS-IS on a device.

```
device# configure terminal
device# ipv6 unicast-routing
device(config)# router isis
ISIS: Please configure NET!
device(config-isis-router)# net 49.2211.0000.00bb.cccc.00
```

# Configuring the IS-IS IPv6 unicast address family

The IS-IS IPv6 unicast address family allows you to configure IPv6 IS-IS unicast settings that are separate and distinct from IPv4 IS-IS unicast settings. The following task enables ISIS IPv6 address family unicast configuration mode where a variety of IS-IS features can be configured.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **router isis** command to enter IS-IS router configuration mode and enable IS-IS on the device.

```
device(config)# router isis
```

3. Enter the **address-family ipv6 unicast** command to enable ISIS address-family IPv6 unicast configuration mode.

```
device(config-isis-router)# address-family ipv6 unicast
```

The following example enables ISIS address-family IPv6 unicast configuration mode.

```
device# configure terminal
device(config)# router isis
device(config-isis-router)# address-family ipv6 unicast
device(config-isis-router-ipv6u)#
```

# Configuring IPv6 IS-IS single topology

If your IS-IS single topology supports both IPv6 and IPv4, you can configure both IPv6 and IPv4 on an IS-IS interface for Level 1, Level 2, or both Level 1 and Level 2. However, if you configure both IPv6 and IPv4 on an IS-IS interface, they must be configured to run on the same level. For example, you can configure IPv6 to run on Level 1 on an interface and IPv4 to also run on Level 1 on the same interface. However, you cannot configure IPv6 to run on Level 1 on an interface and IPv4 to run to Level 2 on the same interface.

To configure an IPv6 IS-IS single topology, you must perform the tasks listed below.

1. Globally enable IS-IS and configure at least one Network Entity Title (NET). The NET is the device's network interface with IS-IS. You can configure up to three NETs on a device.
2. Configure the desired device interfaces with an IPv6 address and enable IPv6 IS-IS on the device interfaces.
3. Configure IS-IS parameters.

# Setting the overload bit

The overload bit can be configured so that when an IS is overloaded, other ISs will not use the overloaded IS to forward traffic. The following task specifies that the overload bit is set upon system startup and remains set until BGP has converged and specifies that the device that 240 seconds is the maximum time that IS-IS will wait for BGP convergence to complete.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **router isis** command to enter IS-IS router configuration mode and enable IS-IS on the device.

```
device(config)# router isis
```

3. Enter the **set-overload-bit** command with the **on-startup** and **wait-for-bgp** parameters, specifying a value, to configure the device to set the overload bit upon system startup. The overload bit remains set until BGP has converged. 240 seconds is the maximum time that IS-IS waits for BGP convergence to complete.

```
device(config-isis-router)# set-overload-bit on-startup wait-for-bgp 240
```

The following example configures the overload bit on system startup until BGP has converged. The maximum time that the device waits for BGP convergence to complete is 86400 seconds.

```
device# configure terminal
device(config)# router isis
device(config-isis-router)# set-overload-bit on-startup wait-for-bgp 86400
```

# Configuring authentication

A device can be configured to authenticate packets sent to or received from an end system (ES) or other intermediate system (IS). The following task configures IS-IS authentication mode, an IS-IS authentication key and temporarily disables IS-IS authentication checking for Level 1 packets.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **router isis** command to enter IS-IS router configuration mode and enable IS-IS on the device.

```
device(config)# router isis
```

3. Enter the **auth-mode** command with the **md5** and **level-1** parameters to configure MD5 authentication for Level 1 packets.

```
device(config-isis-router)# auth-mode MD5 level-1
```

4. Enter the **no auth-check** command with the **level-1** parameter to temporarily disable authentication checking globally for Level 1 packets.

```
device(config-isis-router)# no auth-check level-1
```

5. Enter the **auth-key** command with the **level-1** parameters, specifying an authentication password to configure an authentication key globally for Level 1 packets.

```
device(config-isis-router)# auth-key 0 mysecurekey level-1
```

The following example configures IS-IS authentication mode, an IS-IS authentication key and temporarily disables IS-IS authentication checking for Level 1 packets.

```
device# configure terminal
device(config)# router isis
device(config-isis-router)# auth-mode MD5 level-1
device(config-isis-router)# no auth-check level-1
device(config-isis-router)# auth-key 0 mysecurekey level-1
```

# Changing the IS-IS level globally

By default, a device can operate as both a Level 1 and Level 2 router. This task globally changes the level supported from Level 1 and Level 2 to Level 1 only.

1. Enter the **configure terminal** command to access global configuration mode.

   ```
   device# configure terminal
   ```

2. Enter the **router isis** command to enter IS-IS router configuration mode and enable IS-IS on the device.

   ```
   device(config)# router isis
   ```

3. Enter the **is-type** command with the **level-1** parameter to globally change the IS-IS level supported from Level-1 and Level-2 to Level-1 only.

   ```
   device(config-isis-router)# is-type level-1
   ```

The following example globally changes the IS-IS level supported from Level-1 and Level-2 to Level-1 only.

```
device# configure terminal
device(config)# router isis
device(config-isis-router)# is-type level-1
```

# Logging adjacency changes

You can configure a device to log changes in the status of an adjacency with another IS. The following task enables the logging of adjacency changes.

1. Enter the **configure terminal** command to access global configuration mode.

   ```
   device# configure terminal
   ```

2. Enter the **router isis** command to enter IS-IS router configuration mode and enable IS-IS on the device.

   ```
   device(config)# router isis
   ```

3. Enter the **log adjacency** command to enable the logging of adjacency changes.

   ```
   device(config-isis-router)# log adjacency
   ```

The following example enables the logging of adjacency changes.

```
device# configure terminal
device(config)# router isis
device(config-isis-router)# log adjacency
```

# Configuring the CSNP interval

By default the Complete Sequence Numbers PDU (CSNP) interval is 10 seconds. The following task configures CSNP interval of 25 seconds for Level 1 and Level 2 packets.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **router isis** command to enter IS-IS router configuration mode and enable IS-IS on the device.

```
device(config)# router isis
```

3. Enter the **csnp-interval** command and specify a value to globally configure the CSNP interval.

```
device(config-isis-router)# csnp-interval 25
```

The following example configures a CSNP interval of 25 seconds.

```
device# configure terminal
device(config)# router isis
device(config-isis-router)# csnp-interval 25
```

# Changing the maximum LSP lifetime

This task changes the maximum number of seconds an unrefreshed LSP remains in a device's LSP database from the default of 1200 seconds (20 minutes) to 2400 seconds (40 minutes).

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **router isis** command to enter IS-IS router configuration mode and enable IS-IS on the device.

```
device(config)# router isis
```

3. Enter the **max-lsp-lifetime** command and specify a value to globally configure the maximum LSP lifetime.

```
device(config-isis-router)# max-lsp-lifetime 2400
```

The following example changes the maximum LSP lifetime to 2400 seconds.

```
device# configure terminal
device(config)# router isis
device(config-isis-router)# max-lsp-lifetime 2400
```

# Changing the LSP refresh interval

This task changes the maximum number of seconds a device waits between sending updated LSPs to its IS-IS neighbors from the default of 900 seconds to 1800 seconds.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **router isis** command to enter IS-IS router configuration mode and enable IS-IS on the device.

```
device(config)# router isis
```

3. Enter the **lsp-refresh-interval** command and specify a value to change the LSP refresh interval.

```
device(config-isis-router)# lsp-refresh-interval 1800
```

The following example changes the LSP refresh interval to 1800 seconds.

```
device# configure terminal
device(config)# router isis
device(config-isis-router)# lsp-refresh-interval 1800
```

# Changing the LSP generation interval

This task changes the minimum number of seconds the device waits between sending updated LSPs to its IS-IS neighbors from the default of 10 seconds to 60 seconds (one minute).

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **router isis** command to enter IS-IS router configuration mode and enable IS-IS on the device.

```
device(config)# router isis
```

3. Enter the **lsp-gen-interval** command and specify a value to change the LSP generation interval.

```
device(config-isis-router)# lsp-gen-interval 60
```

The following example changes the LSP generation interval to 60 seconds.

```
device# configure terminal
device(config)# router isis
device(config-isis-router)# lsp-gen-interval 60
```

# Changing the LSP interval and retransmit interval

The LSP interval is the rate of transmission, in milliseconds, of the LSPs. The retransmit interval is the time, in seconds, the device waits before it retransmits LSPs. This task changes the LSP interval to 45 milliseconds and changes the retransmission interval to 10 milliseconds.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **router isis** command to enter IS-IS router configuration mode and enable IS-IS on the device.

```
device(config)# router isis
```

3. Enter the **lsp-interval 45** command and specify a value to change the LSP interval from the default of 33 milliseconds.

```
device(config-isis-router)# lsp-interval 42
```

4. Enter the **retransmit-interval** command and specify a value to change the retransmission interval from the default of 5 seconds.

```
device(config-isis-router)# retransmit-interval 10
```

The following example changes the LSP interval to 45 milliseconds and changes the retransmission interval to 10 milliseconds.

```
device# configure terminal
device(config)# router isis
device(config-isis-router)# lsp-interval 42
device(config-isis-router)# retransmit-interval 10
```

# Disabling IS-IS name mapping capability

The implementation of IS-IS supports RFC 2763, which describes a mechanism for mapping IS-IS system IDs to the hostnames of the devices with those IDs. The following task disables IS-IS name mapping.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **router isis** command to enter IS-IS router configuration mode and enable IS-IS on the device.

```
device(config)# router isis
```

3. Enter the **no hostname** command to disable IS-IS name mapping.

```
device(config-isis-router)# no hostname
```

The following example globally disables IS-IS name mapping.

```
device# configure terminal
device(config)# router isis
device(config-isis-router)# no hostname
```

# Logging invalid LSP packets received

You can configure a device to provide logging of invalid LSP packets. The following task enables the logging of invalid LSP packets.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **router isis** command to enter IS-IS router configuration mode and enable IS-IS on the device.

```
device(config)# router isis
```

3. Enter the **log invalid-lsp-packets** command to enable the logging of invalid LSP packets.

```
device(config-isis-router)# log invalid-lsp-packets
```

The following example enables the logging of invalid LSP packets.

```
device# configure terminal
device(config)# router isis
device(config-isis-router)# log invalid-lsp-packets
```

# Changing the SPF timer

Every IS maintains a Shortest Path First (SPF) tree, which is a representation of the states of each of the IS's links to ESs and other ISs. If the IS is both a Level-1 and Level-2 IS, it maintains separate SPF trees for each level. To ensure that the SPF tree remains current, the IS updates the tree at regular intervals following a change in network topology or the link state database. This task changes the maximum interval in seconds between SPF recalculations, the initial SPF calculation delay, and the hold time between the first and second SPF calculation.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **router isis** command to enter IS-IS router configuration mode and enable IS-IS on the device.

```
device(config)# router isis
```

3. Enter the **spf-interval** command with the **level-1** parameter, specifying values for the *max-wait* , *initial-wait* , and *second-wait* parameters, to set the maximum interval in seconds between SPF recalculations, the initial SPF calculation delay, and the hold time between the first and second SPF calculation.

```
device(config-isis-router)# spf-interval level-1 15 10000 15000
```

The following example specifies that the maximum interval in seconds between SPF recalculations is 15 seconds for Level 1 packets. The initial SPF calculation delay is 10000 milliseconds and the hold time between the first and second SPF calculation is 15000 millseconds.

```
device# configure terminal
device(config)# router isis
device(config-isis-router)# spf-interval level-1 15 10000 15000
```

# Configuring the IS-IS flooding mechanism

You can configure IS-IS to flood Link State PDUs to other devices in the network before running SPF, thus improving database synchronization by allowing LSP changes to be propagated to neighbors before running SPF. The following task configures the IS-IS fast-flood feature.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **router isis** command to enter IS-IS router configuration mode and enable IS-IS on the device.

```
device(config)# router isis
```

3. Enter the **fast-flood** command and specify a value to implement IS-IS fast-flood.

```
device(config-isis-router)# fast-flood 10
```

The following example configures IS-IS to flood 10 LSPs before running SPF.

```
device# configure terminal
device(config)# router isis
device(config-isis-router)# fast-flood 10
```

# Configuring IS-IS PSPF exponential back-off

The following task changes the partial shortest path first (PSPF) interval, changing the maximum interval between SPF recalculations, the initial SPF calculation delay, and the hold time between the first and second SPF calculation.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **router isis** command to enter IS-IS router configuration mode and enable IS-IS on the device.

```
device(config)# router isis
```

3. Enter the **partial-spf-interval** command with the **level-1** parameter, specifying values for the *max-wait* , *initial-wait* , and *second-wait* parameters, to set the maximum interval between SPF recalculationss, the initial SPF calculation delay, and the hold time between the first and second SPF calculation.

```
device(config-isis-router)# partial-spf-interval 15 10000 15000
```

The following example specifies that the maximum interval in seconds between SPF recalculations is 15 seconds. The initial SPF calculation delay is 10000 milliseconds and the hold time between the first and second SPF calculation is 15000 milliseconds.

```
device# configure terminal
device(config)# router isis
device(config-isis-router)# partial-spf-interval 15 10000 15000
```

# Disabling hello padding globally

By default, a device adds extra data to the end of a hello packet to make the packet the same size as the maximum length of PDU the device supports. The following task disables hello padding globally.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **router isis** command to enter IS-IS router configuration mode and enable IS-IS on the device.

```
device(config)# router isis
```

3. Enter the **no hello padding** command to disable hello padding globally.

```
device(config-isis-router)# no hello padding
```

The following example disables hello padding globally.

```
device# configure terminal
device(config)# router isis
device(config-isis-router)# no hello padding
```

# Disabling partial SPF optimizations

You can configure IS-IS to perform a full SPF calculation when any network (non-topology) change occurs so that IS-IS always runs full SPF for all such network changes. This task disables partial SPF optimizationss.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **router isis** command to enter IS-IS router configuration mode and enable IS-IS on the device.

```
device(config)# router isis
```

3. Enter the **disable-partial-spf-opt** command to disable partial SPF optimizations.

```
device(config-isis-router)# disable-partial-spf-opt
```

The following example disables partial SPF optimizations.

```
device# configure terminal
device(config)# router isis
device(config-isis-router)# disable-partial-spf-opt
```

# Disabling incremental SPF optimizations

IS-IS is configured to use an incremental shortcut LSP SPF optimization algorithm by default. This task disables incremental shortcut LSP SPF optimization.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **router isis** command to enter IS-IS router configuration mode and enable IS-IS on the device.

```
device(config)# router isis
```

3. Enter the **disable-inc-stct-spf-opt** command to disable incremental shortcut LSP SPF optimization.

```
device(config-isis-router)# disable-inc-stct-spf-opt
```

The following example disables incremental shortcut LSP SPF optimization.

```
device# configure terminal
device(config)# router isis
device(config-isis-router)# disable-inc-stct-spf-opt
```

# Disabling incremental shortcut SPF optimizations

You can configure IS-IS to perform a full SPF calculation when any network topology change occurs so that IS-IS always runs full SPF for all such network changes. This task disables incremental SPF optimizations so that IS-IS always runs full SPF for all such network changes.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **router isis** command to enter IS-IS router configuration mode and enable IS-IS on the device.

```
device(config)# router isis
```

3. Enter the **disable-partial-spf-opt** command to disable partial SPF optimizations.

```
device(config-isis-router)# disable-incremental-spf-opt
```

The following example disables incremental SPF optimizations.

```
device# configure terminal
device(config)# router isis
device(config-isis-router)# disable-incremental-spf-opt
```

# Maximum number of load sharing paths

You can change the number of paths IPv6 IS-IS can calculate and install in the IPv6 forwarding table. If you change the number of paths to one, the device does not load share multiple route paths learned from IPv4 IS-IS.

NOTE
The maximum number of paths supported by the BR-MLX-10Gx24-DM module is 16.

By default, IPv6 IS-IS can calculate and install four equal-cost paths into the IPv4 forwarding table.

## Changing the maximum number of load sharing paths

You can specify the number of paths IS-IS can calculate and install in the IP forwarding table. The following task specifies that the number of paths IS-IS can calculate and install in the IP forwarding table is 5 for the IS-IS IPv6 unicast address family.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **router isis** command to enter IS-IS router configuration mode and enable IS-IS on the device.

```
device(config)# router isis
```

3. Enter the **address-family ipv6 unicast** command to enable ISIS address-family IPv6 unicast configuration mode.

```
device(config-isis-router)# address-family ipv6 unicast
```

4. Enter the **maximum-paths** command and specify a value to specify the number of paths IS-IS can calculate and install in the IP forwarding table.

```
device(config-isis-router-ipv6u)# maximum-paths 5
```

The following example specifies that the number of paths IS-IS can calculate and install in the IP forwarding table is 5 for the IS-IS IPv6 unicast address family.

```
device# configure terminal
device(config)# router isis
device(config-isis-router)# address-family ipv6 unicast
device(config-isis-router-ipv6u)# maximum-paths 5
```

# Default route advertisement

By default, a device does not generate or advertise a default route to its neighboring ISs. A default route is not advertised even if the device's IPv6 route table contains a default route. You can enable the device to advertise a default route to all neighboring ISs.

NOTE
This feature requires the presence of a default route in the IPv6 route table.

You can also use a route map to configure the device to advertise a default route. The route map must be configured before you can use the route map to configure the device to advertise the default route.

## Enabling advertisement of a default route

You can enable the device to advertise a default route to all neighboring ISs. The following task enables the advertisement of a default route.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **router isis** command to enter IS-IS router configuration mode and enable IS-IS on the device.

```
device(config)# router isis
```

3. Enter the **address-family ipv6 unicast** command to enable ISIS address-family IPv6 unicast configuration mode.

```
device(config-isis-router)# address-family ipv6 unicast
```

4. Enter the **default-information-originate** command to enable the advertisement of a default route.

```
device(config-isis-router-ipv6u)# default-information-originate
```

The following example enables the advertisement of a default route for the IS-IS IPv6 unicast address-family.

```
device# configure terminal
device(config)# router isis
device(config-isis-router)# address-family ipv6 unicast
device(config-isis-router-ipv6u)# default-information-originate
```

## Using a route map to advertise a default route

You can use a route map to configure the device to advertise a default route to all neighboring ISs. The following task enables the advertisement of a default route if the route map "myroutemap" is satisfied for the IS-IS IPv6 address family.

The route-map "myroutemap" must already be configured.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **router isis** command to enter IS-IS router configuration mode and enable IS-IS on the device.

```
device(config)# router isis
```

3. Enter the **address-family ipv6 unicast** command to enable ISIS address-family IPv6 unicast configuration mode.

```
device(config-isis-router)# address-family ipv6 unicast
```

4. Enter the **default-information-originate** command with the **route-map** parameter and specify a route-map parameter to use the route map to enable the advertisement of a default route.

```
device(config-isis-router-ipv6u)# default-information-originate route-map myroutemap
```

The following example enables the advertisement of a default route if the route map "myroutemap" is satisfied for the IPv6 address family.

```
device# configure terminal
device(config)# router isis
device(config-isis-router)# address-family ipv6 unicast
device(config-isis-router-ipv6u)# default-information-originate route-map myroutemap
```

# IPv6 IS-IS administrative distance

When the device has paths from multiple routing protocols to the same destination, it compares the administrative distances of the paths and selects the path with the lowest administrative distance to place in the IPv6 route table.

For example, if the device has a path from iBGP, from OSPFv3, and IPv6 IS-IS to the same destination, and all the paths are using their protocols' default administrative distances, the device selects the OSPFv3 path, because that path has a lower administrative distance than the iBGP and IPv6 IS-IS paths.

The default IPv6 administrative distances on the device are:

- Directly connected - 0 (this value is not configurable)
- Static - 1 (applies to all static routes, including default routes)
- eBGP - 20
- OSPFv3 - 110
- IPv6 IS-IS - 115
- RIPng - 120
- iBGP - 200
- Local BGP - 200
- Unknown - 255 (the device will not use this route)

Lower administrative distances are preferred over higher distances. For example, if the device receives routes for the same network from IPv6 IS-IS and from iBGP, it will prefer the IPv6 IS-IS route by default.

## Changing the administrative distance for IPv6 IS-IS

You can change the administrative distance for IPv6 IS-IS routes. The following task changes the administrative distance for the IS-IS IPv6 unicast address family.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **router isis** command to enter IS-IS router configuration mode and enable IS-IS on the device.

```
device(config)# router isis
```

3. Enter the **address-family ipv6 unicast** command to enable ISIS address-family IPv6 unicast configuration mode.

```
device(config-isis-router)# address-family ipv6 unicast
```

4. Enter the **distance** command and specify a value to change the administrative distance for the IPv6 IS-IS unicast address family.

```
device(config-isis-router-ipv6u)# distance 60
```

The following example sets an administrative distance of 60 for the IPv6 unicast address family.

```
device# configure terminal
device(config)# router isis
device(config-isis-router)# address-family ipv6 unicast
device(config-isis-router-ipv6u)# distance 60
```

# Configuring summary prefixes

You can configure summary prefixes to aggregate IS-IS IPv6 route information. Summary prefixes can enhance performance by reducing the size of the Link State database, reducing the amount of data the device needs to send to its neighbors, and reducing the CPU cycles used for IPv6 IS-IS. The following task configures a summary prefix 2001:db8::/32 for Level 1 routes.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **router isis** command to enter IS-IS router configuration mode and enable IS-IS on the device.

```
device(config)# router isis
```

3. Enter the **address-family ipv6 unicast** command to enable ISIS address-family IPv6 unicast configuration mode.

```
device(config-isis-router)# address-family ipv6 unicast
```

4. Enter the **summary-prefix** command with the **level-1** parameter, specifying an IPv6 address and network mask, to configure a summary prefix and network mask for Level 1 routes.

```
device(config-isis-router-ipv6u)# summary-prefix 2001:db8::/32 level-1
```

The following example configures asummary-prefix of 2001:db8::/32 for Level 1 routes.

```
device# configure terminal
device(config)# router isis
device(config-isis-router)# address-family ipv6 unicast
device(config-isis-router-ipv6u)# summary-prefix 2001:db8::/32 level-1
```

# Redistributing routes into IPv6 IS-IS

Routes can be redistributed into IPv6 IS-IS by performing the following configuration tasks:

- Changing the default redistribution metric (optional).
- Configuring the redistribution of a particular route type into IPv6 IS-IS (mandatory).

The device can redistribute routes from the following route sources into IPv6 IS-IS:

- BGP4+
- RIPng

- OSPFv3
- Static IPv6 routes
- IPv6 routes learned from directly connected networks

The device can also can redistribute Level 1 IPv6 IS-IS routes into Level 2 IPv6 IS-IS routes, and Level 2 IPv6 IS-IS routes into Level 1 IPv6 IS-IS routes.

Route redistribution from other sources into IPv6 IS-IS is disabled by default. When you enable redistribution, the device redistributes routes only into Level 2 by default. You can specify Level 1 only, Level 2 only, or Level 1 and Level 2 when you enable redistribution.

The device automatically redistributes Level 1 routes into Level 2 routes. Thus, you do not need to enable this type of redistribution. You also can enable redistribution of Level 2 routes into Level 1 routes.

The device attempts to use the redistributed route's metric as the route's IPv6 IS-IS metric. For example, if an OSPFv3 route has an OSPF cost of 20, the device uses 20 as the route's IPv6 IS-IS metric. The device uses the redistributed route's metric as the IPv6 IS-IS metric unless the route does not a have a valid metric. In this case, the device assigns the default metric value to the route.

An IS-IS LSP can hold around 32,000 IPv4 routes or 11,000 IPv6 routes. This number can vary depending on the prefix length of the routes.

# Redistributing routes into IPv6 IS-IS

Routes can be redistributed for IPv6-IS-IS, and the routes to be redistributed can be specified.

The redistribution of static routes into IPv6 IS-IS is configured on device1. The redistribution of BGP routes into IPv6 IS-IS is configured on device2, and the BGP routes to be redistributed are specified.

1. On device1, enter the **configure terminal** command to access global configuration mode.

   ```
   device1# configure terminal
   ```

2. Enter the **router isis** command to enter IS-IS router configuration mode and enable IS-IS on the device.

   ```
   device1(config)# router isis
   ```

3. Enter the **address-family ipv6 unicast** command to enable ISIS address-family IPv6 unicast configuration mode.

   ```
   device1(config-isis-router)# address-family ipv6 unicast
   ```

4. Enter the **redistribute** command with the **static** parameter to redistribute static routes.

   ```
   device(1config-isis-router-ipv6u)# redistribute static
   ```

5. On device2, enter the **configure terminal** command to access global configuration mode.

   ```
   device2# configure terminal
   ```

6. Enter the **router isis** command to enter IS-IS router configuration mode and enable IS-IS on the device.

   ```
   device2(config)# router isis
   ```

7. Enter the **address-family ipv6 unicast** command to enable ISIS address-family IPv6 unicast configuration mode.

   ```
   device2(config-isis-router)# address-family ipv6 unicast
   ```

8. Enter the **redistribute** command with the **bgp** and **route-map** parameters to redistribute BGP routes and specify a route map.

   ```
   device2(config-isis-router-ipv6u)# redistribute bgp route-map myroutemap
   ```

The following example redistributes static routes into IPv6 IS-IS on a device.

```
device# configure terminal
device(config)# router isis
device(config-isis-router)# address-family ipv6 unicast
device(config-isis-router-ipv6u)# redistribute static
```

The following example redistributes BGP routes into IPv6 IS-IS on a device and specifies a route map.

```
device# configure terminal
device(config)# router isis
device(config-isis-router)# address-family ipv6 unicast
device(config-isis-router-ipv6u)# redistribute bgp route-map myroutemap
```

# Default redistribution metric

When IPv6 IS-IS redistributes a route from another route source (such as OSPv3F, BGP4+, or a static IPv6 route) into IPv6 IS-IS, the route's metric value is used as its metric when the metric is not modified by a route map or metric parameter and the default redistribution metric is set to its default value of 0.

The default metric value can be changed.

NOTE
The implementation of IS-IS does not support the optional metric types Delay, Expense, or Error.

## Changing the default redistribution metric

You can change the default metric for the IS-IS routing protocol. The following task changes the default metric value for the IPv6 unicast address family.

1. Enter the **configure terminal** command to access global configuration mode.

   ```
   device# configure terminal
   ```

2. Enter the **router isis** command to enter IS-IS router configuration mode and enable IS-IS on the device.

   ```
   device(config)# router isis
   ```

3. Enter the **address-family ipv6 unicast** command to enable ISIS address-family IPv6 unicast configuration mode.

   ```
   device(config-isis-router)# address-family ipv6 unicast
   ```

4. Enter the **default-metric** command and specify a value to change the default metric value for the IPv6 unicast address family.

   ```
   device(config-isis-router-ipv6u)# default-metric 45
   ```

The following example sets the default value to 45 for the IPv6 unicast address family.

```
device# configure terminal
device(config)# router isis
device(config-isis-router)# address-family ipv6 unicast
device(config-isis-router-ipv6u)# default-metric 45
```

# Configuring the default link metric value globally

You can configure the metric value globally on all active IS-IS interfaces for the IS-IS IPv6 address family. The following task sets the IS-IS default link metric value for Level 1 for the IPv6 address family.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **router isis** command to enter IS-IS router configuration mode and enable IS-IS on the device.

```
device(config)# router isis
```

3. Enter the **address-family ipv6 unicast** command to enable ISIS address-family IPv6 unicast configuration mode.

```
device(config-isis-router)# address-family ipv6 unicast
```

4. Enter the **default-link-metric** command with the **level-1** parameter, and specify a value, to set the IS-IS default link metric value for Level 1 for the IPv6 address family.

```
device(config-isis-router-ipv6u)# default-link-metric 45 level-1
```

The following example sets the IS-IS default link metric value for Level 1 for the IPv6 address family.

```
device# configure terminal
device(config)# router isis
device(config-isis-router)# address-family ipv6 unicast
device(config-isis-router-ipv6u)# default-link-metric 45 level-1
```

# IPv6 metric behavior with multi-topology configuration

The default-link-metric for IPv6 depends upon the multi-topology configuration.

**No multi-topology:** The IPv6 default-link-metric is the same as that configured for IPv4 address-family.

**Multi-topology:** The IPv6 default-link-metric is equal to the value configured for IPv6 address-family.

**Multi-topology transition:** The IPv6 default-link-metric will be equal to the value configured for IPv6 address-family.

# IS-IS metric styles

There are two types of metric styles in IS-IS:

- narrow metric
- wide metric

The range of the metric value is different for both of these styles. If there is a change in the metric-style configuration, the default-link-metric changes with it so that the new value of the default-link-metric is equal to the minimum of both the configured value and the maximum value supported for the new metric-style.

If the metric style changes from narrow metric to wide metric, no change in the value of default-link-metric occurs.

If the metric style changes from wide metric to narrow metric, and if the value of default-link-metric is greater than 63, the default-link-metric takes the value 63, as it is the maximum supported in the narrow metric.

# IPv6 IS-IS non-stop routing

IPv6 IS-IS can continue operation without interruption during a processor switchover or hitless-reload event when the non-stop routing (NSR) feature is enabled.

In general, a device restart causes its peer to remove the routes originated from the router and reinstalls them. The IPv6 IS-IS NSR feature enables the device to maintain neighbors and LSA database with its peer in the event of a restart. In IS-IS NSR, the processor switchovers and the hitless-reloads are treated the same as they are during startup and the overload bit is set in the same way as it is after a reboot. IPv6 IS-IS NSR is compatible with IPv4 IS-IS NSR.

NOTE
IPv6 IS-IS NSR is not supported on the CES 2000 Series and CER 2000 Series platforms.

NOTE
IPv6 IS-IS NSR is independent of Graceful Restart (GR) and GR help role mechanisms.

## Enabling non-stop routing

IS-IS non-stop routing (NSR) can be re-enabled if it has been disabled. The following task re-enables NSR for IS-IS.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **router isis** command to enter IS-IS router configuration mode and enable IS-IS on the device.

```
device(config)# router isis
```

3. Enter the **nonstop-routing** command to re-enable NSR.

```
device(config-isis-router)# nonstop-routing
```

The following example re-enables NSR.

```
device# configure terminal
device(config)# router isis
device(config-isis-router)# nonstop-routing
```

# Limitations of IPv6 IS-IS non-stop routing

- The IS-IS over GRE tunnel feature does not support IS-IS NSR. The GRE tunnel interface types are not supported.
- The IS-IS shortcuts are not supported because they depend on the MPLS tunnel.
- If the IS-IS hellos are forwarded at Layer 2 and the device executes a hitless-reload, hellos will not be forwarded for a brief time. The IS-IS adjacencies are lost for 12 seconds and there will be data traffic loss.
- The configuration events that occur close to switchover or hitless-reload may get lost due to CLI synchronization issues.
- The neighbor or interface state changes close to switchover or hitless-reload cannot be handled.
- The IS-IS neighbor hold timer is restarted upon IS-IS NSR switchover or hitless-reload.
- It is recommended to use the default IS-IS hello timer value. IS-IS neighbor sessions may flap during a linecard software upgrade if a shorter timer value is used.
- The traffic counters are not synchronized because the neighbor and LSP database counters are recalculated on the standby module during synchronization.

- When IS-IS NSR is enabled, after switchover or hitless-reload to standby MP, IS-IS routes, LSP database and neighbor adjacencies are maintained so that there will be no loss of existing traffic to the IS-IS destinations.
- The IS-IS NSR hitless failover event may not be completely invisible to the network because, after switchover, additional flooding of CSNP packets will occur in the directly connected neighbors.

# IPv6 protocol-support consistency checks

An IS-IS single topology must be configured to run the same set of network protocols (IPv4 IS-IS only, IPv6 IS-IS only, or both IPv4 IS-IS and IPv6 IS-IS).

By default, IS-IS performs consistency checks on hello packets. If a hello packet does not have the same configured network protocols, IS-IS rejects the packet. For example, a hello packet from a device running IPv4 and IPv6 IS-IS will be rejected by a device running either IPv4 IS-IS only or IPv6 IS-IS only, and the two devices will not become adjacent.

You can configure two devices running different sets of network protocols to form an adjacency.

# Enabling IS-IS and assigning an IPv6 address to an interface

IS-IS can be enabled for a specified interface for a device. Several commands can then be accessed that allow the configuration of IS-IS on the specified interface. This task enables IS-IS routing for an interface Ethernet.

1. Enter the **configure terminal** command to access global configuration mode.

   ```
   device# configure terminal
   ```

2. Enter the **interface ethernet** command and specify an interface.

   ```
   device(config)# interface ethernet 1/2
   ```

3. Enter the **ipv6 address** command with the **eui-64** parameter, specifying an IPv6 address, to assign the IPv6 address to the interface and configure the IPv6 address with an EUI-64 interface ID in the low-order 64 bits.

   ```
   device(config-if-e1000-1/2)# ipv6 address 2001:db8:12d:1300::/64 eui-64
   ```

4. Enter the **ipv6 router isis** command to enable IS-IS for the interface.

   ```
   device(config-if-e1000-1/2)# ipv6 router isis
   ```

The following example enables IS-IS routing for an interface Ethernet and assigns an IPv6 address to the interface.

```
device# configure terminal
device(config)# interface ethernet 1/2
device(config-if-e1000-1/2)# ipv6 address 2001:db8:12d:1300::/64 eui-64
device(config-if-e1000-1/2)# ipv6 router isis
```

# Configuring authentication on an IS-IS interface

The following task configures IS-IS authentication mode, an IS-IS authentication key, and temporarily disables IS-IS authentication checking for Level 2 packets on an IS-IS Ethernet interface.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **interface ethernet** command and specify an interface.

```
device(config)# interface ethernet 1/2
```

3. Enter the **isis auth-mode** command with the **md5** and **level-2** parameters to configure MD5 authentication for Level 2 packets for the IS-IS interface.

```
device(config-if-e1000-1/2)# isis auth-mode MD5 level-2
```

4. Enter the **isis auth-key** command with the **level-2** parameter, specifying an authentication password, to configure an authentication key for Level 2 packets for the IS-IS interface.

```
device(config-if-e1000-1/2)# isis auth-key 0 mykey level-2
```

5. Enter the **no isis auth-check** command to temporarily disable authentication checking for Level 1 and Level 2 packets for the IS-IS interface.

```
device(config-if-e1000-1/2)# no isis auth-check
```

The following example configures IS-IS authentication mode, an IS-IS authentication key and temporarily disables IS-IS authentication checking for a specified IS-IS interface.

```
device# configure terminal
device(config)# interface ethernet 1/2
device(config-if-e1000-1/2)# isis auth-mode MD5 level-2
device(config-if-e1000-1/2)# isis auth-key 0 mykey level-2
device(config-if-e1000-1/2)# no isis auth-check
```

# Disabling hello padding for an IS-IS interface

By default, a device adds extra data to the end of a hello packet to make the packet the same size as the maximum length of PDU the device supports. The following task disables hello padding for an IS-IS Ethernet interface.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **interface ethernet** command and specify an interface.

```
device(config)# interface ethernet 1/2
```

3. Enter the **no isis hello padding** command disable hello padding for the IS-IS Ethernet interface.

```
device(config-if-e1000-1/2)# no isis hello padding
```

The following example disables hello padding for a specified IS-IS Ethernet interface.

```
device# configure terminal
device(config)# interface ethernet 1/2
device(config-if-e1000-1/2)# no isis hello padding
```

# Changing the IS-IS level on an IS-IS interface

By default, a device can operate as both a Level 1 and Level 2 router. The following task changes the level supported from Level 1 and Level 2 to Level 1 for an IS-IS interface.

If you change the IS-IS type on an individual interface, the type you specify must also be specified globally. For example, if you globally set the type to Level 2 only, you cannot set the type on an individual interface to Level 1. The setting is accepted but does not take effect.

1. Enter the **configure terminal** command to access global configuration mode.

   ```
   device# configure terminal
   ```

2. Enter the **interface ethernet** command and specify an interface.

   ```
   device(config)# interface ethernet 1/2
   ```

3. Enter the **isis circut-type** command with the **level-1** parameter to configure a Level 1 adjacency for the interface.

   ```
   device(config-if-e1000-1/2)# isis circut-type level-1

   device(conf-if-eth-1/2)# isis circut-type level-1
   ```

The following example configures a Level 1 adjacency on an IS-IS Ethernet interface.

```
device# configure terminal
device(config)# interface ethernet 1/2
device(config-if-e1000-1/2)# isis circut-type level-1
```

# Changing the hello multiplier for an IS-IS interface

The hello multiplier is the number by which an IS-IS interface multiplies the hello interval to obtain the hold time for Level 1 and Level 2 IS-to-IS hello PDUs. The following task changes the hello multiplier for Level 2 packets for an Ethernet interface.

1. Enter the **configure terminal** command to access global configuration mode.

   ```
   device# configure terminal
   ```

2. Enter the **interface ethernet** command and specify an interface.

   ```
   device(config)# interface ethernet 1/2
   ```

3. Enter the **isis hello-mulitplier** command with the **level-2** parameter, and specify a value, to change the hello multiplier for Level 2 packets for the interface.

   ```
   device(config-if-e1000-1/2)# isis hello-mulitplier 20 level-2
   ```

The following example changes the hello multiplier to 20 for Level 2 packets for an Ethernet interface.

```
device# configure terminal
device(config)# interface ethernet 1/2
device(config-if-e1000-1/2)# isis hello-mulitplier 20 level-2
```

# Changing the hello interval for an IS-IS interface

The hello interval specifies how often an IS-IS interface sends hello messages to its IS-IS neighbors. The following task changes the hello interval for Level 1 packets to 20 for an IS-IS interface.

1. Enter the **configure terminal** command to access global configuration mode.

   ```
   device# configure terminal
   ```

2. Enter the **interface ethernet** command and specify an interface.

   ```
   device(config)# interface ethernet 1/2
   ```

3. Enter the **isis hello-interval** command with the **level-1** parameter, and specify a value, to change the hello interval for Level 1 packets for the interface.

   ```
   device(config-if-e1000-1/2)# isis hello-interval 20 level-1
   ```

The following example changes the hello interval for Level 1 packets to 20 on a Loopback interface.

```
device# configure terminal
device(config)# interface ethernet 1/2
device(config-if-e1000-1/2)# isis hello-interval 20 level-1
```

# Changing the metric added to advertised routes for an IS-IS interface

When a device originates an IS-IS route or calculates a route, it adds a metric (cost) to the route. Each IS-IS interface has a separate metric value. The following task changes the IS-IS metric for an IS-IS Ethernet interface.

1. Enter the **configure terminal** command to access global configuration mode.

   ```
   device# configure terminal
   ```

2. Enter the **interface ethernet** command and specify an interface.

   ```
   device(config)# interface ethernet 1/2
   ```

3. Enter the **isis metric** command with the **level-1** parameter and specify a value to change the metric for the interface, specifying Level 1 packets.

   ```
   device(config-if-e1000-1/2)# isis metric level-1 38
   ```

The following example changes the metric for an IS-IS Ethernet interface to 38, specifying Level 1 packets. .

```
device# configure terminal
device(config)# interface ethernet 1/2
device(config-if-e1000-1/2)# isis metric level-1 38
```

# Disabling IPv6 protocol-support consistency checks

By default, IS-IS performs consistency checks on hello packets. If a hello packet does not have the same configured network protocols, IS-IS rejects the packet. The following task disables the IS-IS IPv6 protocol-support consistency checks so that two devices running different sets of network protocols can form an adjacency.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **router isis** command to enter IS-IS router configuration mode and enable IS-IS on the device.

```
device(config)# router isis
```

3. Enter the **address-family ipv6 unicast** command to enable ISIS address-family IPv6 unicast configuration mode.

```
device(config-isis-router)# address-family ipv6 unicast
```

4. Enter the **no adjacency-check** command and to disable IS-IS IPv6 protocol-support consistency checks.

```
device(config-isis-router-ipv6u)# no adjacency-check
```

The following example disables IS-IS IPv6 protocol-support consistency checks..

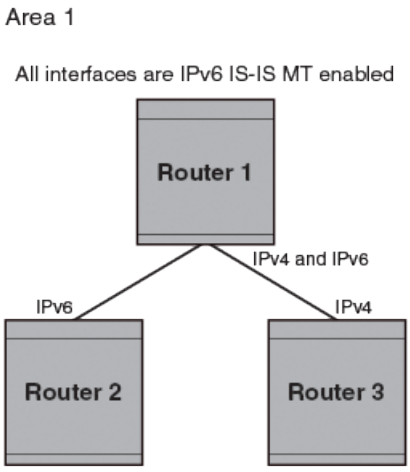
```
device# configure terminal
device(config)# router isis
device(config-isis-router)# address-family ipv6 unicast
device(config-isis-router-ipv6u)# no adjacency-check
```

# IPv6 IS-IS Multi-Topology

IPv6 IS-IS supports Multi-Topology (MT) mode, which allows you to configure both IPv4 and IPv6 topologies on the router interfaces in an area or a domain. However, when implementing an MT, all routers in an area (Level 1 routing) or a domain (Level 2 routing) can be configured with a set of independent topologies on all their interfaces, even on loopback interfaces. All routers in an area or a domain use the same type of IPv6 support, either single-topology or MT. In a network, the Shortest Path First (SPF) is calculated for each configured topology.

The figure below depicts a non-congruent topology with IPv6 IS-IS MT enabled. Router 1 is an IPv4 and IPv6 dual stack router, Router 2 is an IPv6 router, and Router 3 is an IPv4 router. All the routers (Router 1, Router 2, and Router 3) in the Area 1 are configured with a set of independent topologies.

**FIGURE 49** IS-IS non-congruent topology




# Configuration considerations for IPv6 IS-IS MT

The following configuration considerations apply for IPv6 IS-IS MT

- The wide metric style must be configured before enabling IPv6 IS-IS MT.
- IPv4, IPv6, or IPv4 and IPv6 configured on the same interface must run on the same IS-IS level.
- Enabling or disabling IPv6 IS-IS MT clears all adjacencies, LSP databases, and IPv6 IS-IS routes.
- All routers on a point-to-point or a broadcast interface must support at least one common topology (IPv4 or IPv6), when MT is enabled.

# Migrating to IPv6 IS-IS MT

The following steps must be performed to migrate from a non-MT environment to an MT environment.

1. Assume that the entire network is not an IPv6 IS-IS MT environment, and ensure that all the routes are correct.
2. Use the **multi-topology** command with the **transition** parameter to enable transition mode on each router one by one, and ensure that all the routes are correct.
3. After all the routers are in transition mode, use the **no multi-topology** command with the **transition** parameter to disable transition mode on each router one by one. Ensure that all the routes are correct.
4. Change the topology to make IPv4 and IPv6 different.

# Maintaining MT adjacencies

With the extension of IPv6 IS-IS MT, the new type, length, and value (TLV) parameters are added into the IS to IS hello (IIH) packets that advertise the topologies of the interface. In IPv6 IS-IS MT, the router advertises its information using the new TLV parameters such as MT ID TLV, MT IS Reachability TLV, MT Reachable IPv4 TLV, and MT Reachable IPv6 TLV. The TLVs specify the types of data, the maximum length of the data, and the valid values for the data.

## Forming adjacencies on the point-to-point interfaces

On a point-to-point interface, adjacencies are formed with IS-IS routers that do not implement MT extensions. If two peers share at least one common topology, then an adjacency is formed between the peers.

## Forming adjacencies on the broadcast interfaces

On a broadcast interface, all the MT-enabled routers advertise their MT capability TLV in their IIH packets. The MT-enabled IS-IS routers form adjacency with any IS-IS routers whether or not MT is enabled. A peering MT-disabled IS-IS router does not form adjacency when NLPID TLVs do not match.

# New TLV attributes

The new TLV parameters to support the IPv6 IS-IS MT extension are MT ID TLV, MT IS Reachability TLV, MT Reachable IPv4 TLV, and MT Reachable IPv6 TLV.

# Enabling IPv6 IS-IS MT

You can enable IPv6 IS-IS MT in an area or a domain so that the MT-enabled devices runs IPv6 IS-IS in multi SPF mode. The following task enables IPv6 IS-IS MT with transition support.

1. Enter the **configure terminal** command to access global configuration mode.

   ```
   device# configure terminal
   ```

2. Enter the **router isis** command to enter IS-IS router configuration mode and enable IS-IS on the device.

   ```
   device(config)# router isis
   ```

3. Enter the **address-family ipv6 unicast** command to enable ISIS address-family IPv6 unicast configuration mode.

   ```
   device(config-isis-router)# address-family ipv6 unicast
   ```

4. Enter the **multi-topology** command with the **transition** parameter to enable IPv6 IS-IS MT with transition support.

   ```
   device(config-isis-router-ipv6u)# multi-topology transition
   ```

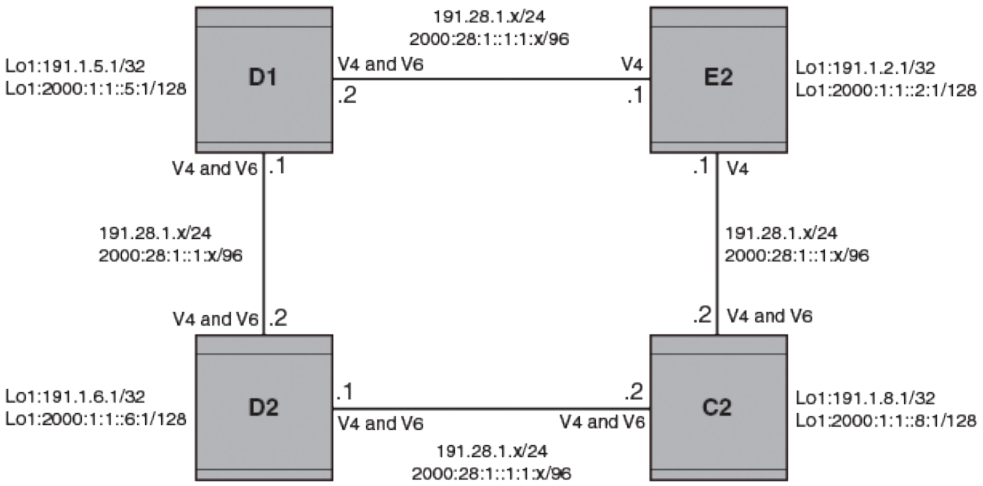
The following example enables IPv6 IS-IS MT with transition support.

```
device# configure terminal
device(config)# router isis
device(config-isis-router)# address-family ipv6 unicast
device(config-isis-router-ipv6u)# multi-topology transition
```

# Configuration example to deploy IPv6 IS-IS MT

The figure below shows an example of a non-congruent topology enabled with IPv6 IS-IS MT. Device D1 supports both the IPv4 and IPv6 topologies, device D2 supports both the IPv4 and IPv6 topologies, device E2 supports an IPv4 topology, and device C2 supports both the IPv4 and IPv6 topologies.

**FIGURE 50** IPv6 IS-IS MT configuration




# Configuration commands to enable IPv6 IS-IS MT on device D1

The following commands enable IPv6 IS-IS MT on device D1.

```
device# configure terminal
device(config)# router isis
device(config-isis-router)# net 00.0000.001b.ed03.1400.00
device(config-isis-router)# address-family ipv4 unicast
device(config-isis-router-ipv4u)# metric-style wide
device(config-isis-router-ipv4u)# exit-address-family
device(config-isis-router)# address-family ipv6 unicast
device(config-isis-router-ipv6u)# multi-topology
device(config-isis-router-ipv6u)# exit-address-family
```

# Configuration commands to enable IPv6 IS-IS MT on device D2

The following commands enable IPv6 IS-IS MT on device D2.

```
device# configure terminal
device(config)# router isis
device(config-isis-router)# net 00.0000.001b.ed04.4400.00
device(config-isis-router)# address-family ipv4 unicast
device(config-isis-router-ipv4u)# metric-style wide
device(config-isis-router-ipv4u)# exit-address-family
device(config-isis-router)# address-family ipv6 unicast
device(config-isis-router-ipv6u)# multi-topology
device(config-isis-router-ipv6u)# exit-address-family
```

# Configuration commands to enable IPv6 IS-IS MT on device E2

The following commands enable IPv6 IS-IS MT on device E2.

```
device# configure terminal
device(config)# router isis
device(config-isis-router)# net 00.0000.001b.ed04.4000.00
device(config-isis-router)# address-family ipv4 unicast
device(config-isis-router-ipv4u)# metric-style wide
device(config-isis-router-ipv4u)# exit-address-family
device(config-isis-router)# address-family ipv6 unicast
```

```
device(config-isis-router-ipv6u)# multi-topology
device(config-isis-router-ipv6u)# exit-address-family
```

## Configuration commands to enable IPv6 IS-IS MT on device C2

The following commands enable IPv6 IS-IS MT on device C2.

```
device# configure terminal
device(config)# router isis
device(config-isis-router)# net 00.0000.001b.ed04.0000.00
device(config-isis-router)# address-family ipv4 unicast
device(config-isis-router-ipv4u)# metric-style wide
device(config-isis-router-ipv4u)# exit-address-family
device(config-isis-router)# address-family ipv6 unicast
device(config-isis-router-ipv6u)# multi-topology
device(config-isis-router-ipv6u)# exit-address-family
```

# Displaying IS-IS statistics

Various **show isis** commands verify information about IPv4 and IPv6 IS-IS configurations.

Use one or more of the following commands to verify IS-IS information. The commands do not have to be entered in this order.

1. Enter the **show isis** command.

```
device# show isis

IS-IS Routing Protocol Operation State: Enabled
IS-Type: Level-1-2
System Id: 768e.f805.5812
Manual area address(es): 11
Level-1-2 Database State: On
Administrative Distance 115
Maximum Paths 8
Default redistribution metric 0
Default link metric for level-1 0 (conf)/ 10 (adv)
Default link metric for level-2 0 (conf)/ 10 (adv)
Protocol Routes Redistributed into IS-IS: None
Number of Routes Redistributed into IS-IS: 0
Level-1 Auth-mode: None
Level-2 Auth-mode: None
Metric Style Supported for Level-1: Narrow
Metric Style Supported for Level-2: Narrow
Graceful-Restart Helper Support: Enabled
ISIS Partial SPF Optimizations: Enabled
Timers:
L1 SPF: Max-wait 5s Init-wait 5000ms Second-wait 5000ms
L2 SPF: Max-wait 5s Init-wait 5000ms Second-wait 5000ms
L1 SPF is not scheduled
L2 SPF is not scheduled
PSPF: Max-wait 5000ms Init-wait 2000ms Second-wait 5000ms
PSPF is not scheduled
LSP: max-lifetime 1200s refresh-interval 900s gen-interval 10s
retransmit-interval 5s, lsp-interval 33ms
SNP: csnp-interval 10s psnp-interval 2s
Global Hello Padding: Enabled
Global Hello Padding For Point to Point Circuits: Enabled
Ptpt Three Way HandShake Mechanism: Enabled
BGP Ipv4 Converged: False BGP Ipv6 Converged: False
IS-IS Traffic Engineering Support: Disabled
No ISIS Shortcuts Configured
BFD: Disabled, BFD HoldoverInterval: 0
NSR: Disabled
LSP-SYNC: Not Globally Enabled
```

   This example output displays general IS-IS information..

2. Enter the **show isis config** command.

```
device# show isis config

router isis
net 11.768e.f805.5812.00
address-family ipv4 unicast
!
address-family ipv6 unicast
!
```

   This example shows the global IS-IS configuration commands that are in effect on the device..

3. Enter the **show isis counts** command.

```
device# show isis counts

Area Mismatch: 0
Max Area Mismatch: 0
System ID Length Mismatch: 0
LSP Sequence Number Skipped: 0
LSP Max Sequence Number Exceeded: 0
Level-1 Database Overload: 1
Level-2 Database Overload: 0
Our LSP Purged: 2
```

This example shows IS-IS error statistics..

4. Enter the **show isis hostname** command.

```
device# show isis hostname

Total number of entries in IS-IS Hostname Table: 1
System ID Hostname * = local IS
-----------------------------------------------------
* 768e.f805.5812 R1
```

This example shows the router-name-to-system-ID mapping table entries for the device.

5. Enter the **show isis interface** command.

```
device# show isis interface

Total number of IS-IS Interfaces: 11
Interface: Ve 301
Circuit State: UP Circuit Mode: Level 1-2
Circuit Type: BCAST Passive State: FALSE
Circuit Number: 2, MTU: 1500
Level-1 Auth-mode: NONE
Level-2 Auth-mode: NONE
Level-1 Metric: 10, Level-1 Priority: 64
Level-1 Hello Interval: 10, Level-1 Hello Multiplier: 3
Level-1 Designated IS: R1-02 Level-1 DIS Changes: 2
Level-2 Metric: 10, Level-2 Priority: 64
Level-2 Hello Interval: 10, Level-2 Hello Multiplier: 3
Level-2 Designated IS: R1-02 Level-2 DIS Changes: 2
Next IS-IS LAN Level-2 Hello in 11 seconds
Number of active Level-2 adjacencies: 0
Next IS-IS LAN Level-1 Hello in 1 seconds
Number of active Level-1 adjacencies: 0
Circuit State Changes: 1 Circuit Adjacencies State Changes: 0
Rejected Adjacencies: 0
Circuit Authentication L1 failures: 0
Circuit Authentication L2 failures: 0
Bad LSPs: 0
Control Messages Sent: 7577 Control Messages Received: 0
Hello Padding: Enabled
IP Enabled: TRUE
IP Addresses:
11.2.1.1/30
IPv6 Enabled: FALSE
MPLS TE Enabled: FALSE
BFD Enabled: FALSE
LDP-SYNC: Disabled, State:
…
```

This example shows information about IS-IS interfaces for a device.

6. Enter the **show isis interface** command, using the **brief** keyword.

```
device# show isis interface brief

Total number of IS-IS Interfaces: 11
Interface Type State Mode Passive MTU UpAdj DIS StateChg AdjStateChg
Ve 301 BCAST UP L12 FALSE 1500 0 None 1 0
Ve 302 BCAST UP L12 FALSE 1500 0 None 1 0
Ve 303 BCAST UP L12 FALSE 1500 0 None 1 0
Ve 304 BCAST UP L12 FALSE 1500 0 None 1 0
Ve 305 BCAST UP L12 FALSE 1500 0 None 1 0
Ve 306 BCAST UP L12 FALSE 1500 0 None 1 0
Ve 307 BCAST UP L12 FALSE 1500 0 None 1 0
Ve 308 BCAST UP L12 FALSE 1500 0 None 1 0
Ve 309 BCAST UP L12 FALSE 1500 0 None 1 0
Ve 310 BCAST UP L12 FALSE 1500 0 None 1 0
Lo 1 BCAST UP L12 TRUE 0 0 None 1 0
```

This example shows summarized information about IS-IS interfaces for a device.

7. Enter the **show isis neighbor** command.

```
device# show isis neighbor
```

This example shows IS-IS neighbor information.

8. Enter the **show isis spf-log** command.

```
device# show isis spf-log

ISIS Level-1 SPF Log
When Duration Nodes Count Last-Trigger-LSP Trigger
10h34m13s 0ms 1 10 R1.00-00 Interface State Change
10h34m38s 0ms 1 2 R1.00-00 Interface Config Change
10h34m43s 0ms 1 18 R1.00-00 Interface Config Change
10h34m48s 0ms 1 5 R1.00-00 Interface State Change
ISIS Level-2 SPF Log
When Duration Nodes Count Last-Trigger-LSP Trigger
10h34m13s 0ms 1 10 R1.00-00 Interface State Change
10h34m38s 0ms 1 2 R1.00-00 Interface Config Change
10h34m43s 0ms 1 18 R1.00-00 Interface Config Change
10h34m48s 0ms 1 5 R1.00-00 Interface State Change
```

This example shows IS-IS link-state packet (LSP) logging information.

9. Enter the **show isis traffic** command.

```
show isis traffic

                            Message Received     Message Sent
 Level-1 Hellos             0                    44912
 Level-2 Hellos             0                    44927
 PTP Hellos                 0                    0
 Level-1 LSP                0                    0
 Level-2 LSP                0                    0
 Level-1 CSNP               0                    0
 Level-2 CSNP               0                    0
 Level-1 PSNP               0                    0
 Level-2 PSNP               0                    0
```

This example shows information about IS-IS packet counts..

# Multi-VRF



# Multi-VRF overview

Virtual Routing and Forwarding (VRF) allows routers to maintain multiple routing tables and forwarding tables on the same router. A Multi-VRF router can run multiple instances of routing protocols with a neighboring router with overlapping address spaces configured on different VRF instances.

Some vendors also use the terms Multi-VRF CE or VRF-Lite for this technology. VRF-Lite provides a reliable mechanism for a network administrator to maintain multiple virtual routers on the same device. The goal of providing isolation among different VPN instances is accomplished without the overhead of heavyweight protocols (such as MPLS) used in secure VPN technologies. Overlapping address spaces can be maintained among the different VPN instances.

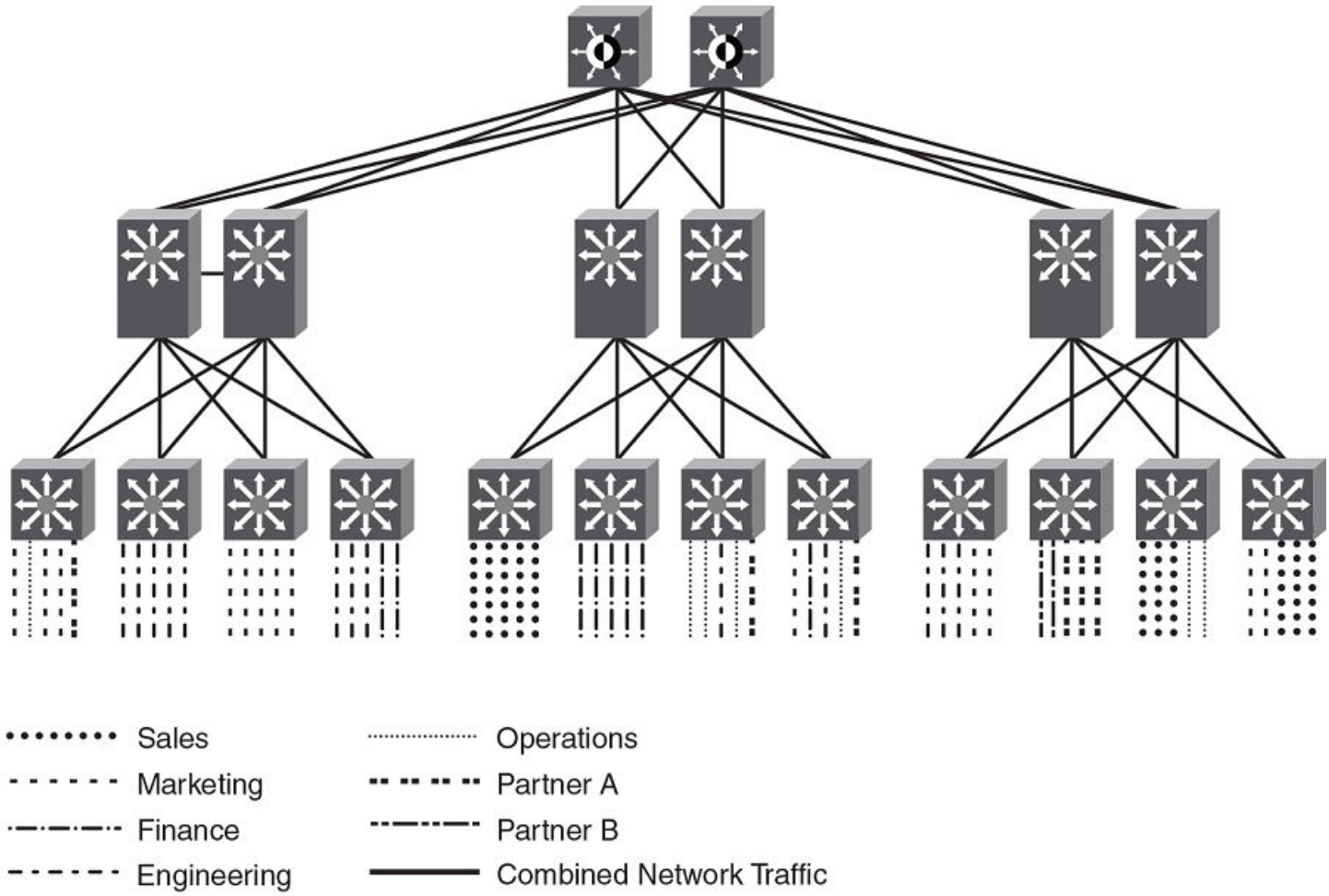
Central to VRF-Lite is the ability to maintain multiple VRF tables on the same Provider Edge (PE) Router. VRF-Lite uses multiple instances of a routing protocol such as OSPF or BGP to exchange route information for a VPN among peer PE routers. The VRF-Lite capable PE router maps an input customer interface to a unique VPN instance. The router maintains a different VRF table for each VPN instance on that PE router. Multiple input interfaces may also be associated with the same VRF on the router, if they connect to sites belonging to the same VPN. This input interface can be a physical interface or a virtual Ethernet interface on a port.

In Multi-VRF deployments:

- Two VRF-capable routers must be directly connected at Layer 3, deploying BGP, OSPF, or static routes.
- Each VRF maintains unique routing and forwarding tables.
- Each VRF can be assigned one or more Layer 3 interfaces on a router to be part of the VRF.
- Each VRF can be configured with IPv4 address family, IPv6 unicast address family, or both.
- A packet's VRF instance is determined based on the VRF index of the interface on which the packet is received.
- Separate routing protocol instances are required for each VRF instance.
- Overlapping address spaces can be configured on different VRF instances.

Multi-VRF deployments provide the flexibility to maintain multiple virtual routers, which are segregated for each VRF instance. The following illustrates a generic, high-level topology where different enterprise functions are assigned unique VRF instances.

**FIGURE 51** Example high-level Multi-VRF topology




A Multi-VRF instance can be configured on any of the following:

- Virtual interfaces
- Loopback interfaces
- Ethernet interfaces
- Tunnel interfaces - The tunnel can belong to any user-defined VRF, but the tunnel source and tunnel destination are restricted to the default VRF.

A Multi-VRF instance **cannot** be configured on any of the following:

- Physical interfaces
- Management interfaces

To configure Multi-VRF, perform the following steps:

- (Optional) Configure tagging on peer interfaces for security.
- Configure VRF instances.
- (Optional) Configure a Route Distinguisher (RD) for new VRF instances.
- Configure an IPv4 address family or IPv6 unicast address family (AF) for new VRF instances.

- Configure routing protocols for new Multi-VRF instances.
- Assign VRF instances to Layer 3 interfaces.

# Configuring Multi-VRF

## Configuring a VRF instance

Do the following to configure a VRF instance.

A device can be configured with more than one VRF instance. You should define each VRF instance before assigning the VRF to a Layer 3 interface. The range of the instance name is from 1 through 255 alphanumeric characters. Each VRF instance is identified by a unique Route Distinguisher (RD), which is prepended to the address being advertised. Because the RD provides overlapping client address space with a unique identifier, the same IP address can be used for different VRFs without conflict. The RD can be an AS number, followed by a colon (:) and a unique arbitrary number as shown below. Alternatively, it can be a local IP address followed by a colon (:) and a unique arbitrary number, as in "1.1.1.1:100." An optional router ID can also be assigned.

Use the **address-family** command in VRF configuration mode to specify an IPv4 or IPv6 address family. For a specific address family you can also configure static route, static ARP, IGMP, and multicast for IPv4, and static route, IPv6 neighbor, and multicast for IPv6.

ATTENTION
Using the **overwrite** option while downloading a configuration from a TFTP server to the running-config will lead to the loss of all VRF configurations when a VRF is configured on a routing interface.

1. In global configuration mode, enter the **vrf** command to create a VRF instance, "corporate" in this example.

```
device(config)# vrf corporate
```

2. (Optional) Assign a Route Distinguisher (RD).

```
device(config-vrf-corporate)# rd 11:11
```

3. (Optional) Assign a router ID.

```
device(config-vrf-corporate)# ip router-id 1.1.1.1
```

4. Use the **address-family unicast (VRF)** command to configure an address family on the VRF and exit. This example uses IPv4.

```
device(config-vrf-corporate)# address-family ipv4 unicast
device(config-vrf-corporate-ipv4)# exit
```

5. Verify the configuration.

```
device(config-vrf-corporate)# do show vrf
Total number of VRFs configured: 2
Status Codes - A:active, D:pending deletion, I:inactive
Name                 Default RD          vrf|v4|v6 Routes Interfaces
corporate            11:11               A  | A| I      0
guest                10:10               A  | A| I      0
Total number of IPv4 unicast route for all non-default VRF is 0
Total number of IPv6 unicast route for all non-default VRF is 0
```

# Starting a routing process for a VRF

You must enable a routing protocol for each VRF instance. This example uses OSPF.

1. In global configuration mode, enable OSPF for the VRF instance "corporate."

```
device(config)# router ospf vrf corporate
```

2. Configure the VRF to use OSPF Area 0.

```
device(config-ospf-router-vrf-corporate)# area 0
```

3. (Optional) Configure the VRF to ensure that essential OSPF neighbor state changes are logged, especially in the case of errors.

```
device(config-ospf-router-vrf-corporate)# log adjacency
```

# Assigning a Layer 3 interface to a VRF

The following example illustrates how a virtual Ethernet (VE) interface is assigned to a VRF, and how IP addresses and the OSPF protocol are configured.

> ATTENTION
> After you configure a VRF instance on the device, you must assign one or more Layer 3 interfaces (physical or virtual Ethernet) to the VRF. When you do this, all existing IP addresses are deleted; this action also triggers cache deletion, route deletion, and associated cleanup. After you assign an interface to the VRF, you must reconfigure the IP address and interface properties.

1. Enter global configuration mode.

```
device# configure terminal
```

2. In global configuration mode, enter the **interface ve** command to create a VE interface.

```
device(config)# interface ve 10
```

3. In VE configuration mode, enable forwarding for the VRF "guest".

```
device(config-vif-10)# vrf forwarding guest
Warning: All IPv4 and IPv6 addresses (including link-local) on this interface have been removed
have been removed
```

4. Configure an IPv4 address and mask on the VE interface.

```
device(config-vif-10)# ip address 192.168.1.254/24
```

5. Enable OSPF Area 0.

```
device(config-vif-10)# ip ospf area 0
```

6. Configure the interface as passive.

```
device(config-vif-10)# ip ospf passive
device(config-vif-10)# exit
```

7. Exit the configuration.

```
device(config-vif-10)# exit
```

# Assigning a loopback interface to a VRF

Do the following to assign a loopback interface to a nondefault VRF.

Because a loopback interface is always available as long as the device is available, it allows routing protocol sessions to stay up even if the outbound interface is down. Assigning a loopback interface to a VRF is similar to assigning any interface. A loopback interface that is not assigned to a nondefault VRF belongs to the default VRF.

1. Enter global configuration mode.

```
device# configure terminal
```

2. In global configuration mode, enter interface subtype configuration mode and assign a loopback interface.

```
device(config)# interface loopback 1
```

3. Use the **vrf forwarding** command to assign the interface to the VRF "customer-1" in this example.

```
device(config-lbif-1)# vrf forwarding customer-1
```

4. Assign an IPv4 address and mask to the loopback interface.

```
device(config-lbif-1)# ip address 10.0.0.1/24
```

# Verifying a Multi-VRF configuration

The following examples illustrate the use of a variety of show commands that are useful in verifying Multi-VRF configurations.

To verify all configured VRFs in summary mode, enter the **show vrf** command, as in the following example.

```
device# show vrf
Total number of VRFs configured: 2
Status Codes - A:active, D:pending deletion, I:inactive
Name Default RD vrf|v4|v6 Routes Interfaces
green 1:1 A | A| A 12 ve111 ve211 ve311*
red 10:12 A | A| A 4 ve1117 port-id tn1*
Total number of IPv4 unicast route for all non-default VRF is 8
Total number of IPv6 unicast route for all non-default VRF is 8
```

To verify a specific VRF in detail mode, enter the **show vrf detail** *vrf-name* command, as in the following example.

```
device# show vrf green
VRF green, default RD 1:1, Table ID 1
IP Router-Id: 1.1.1.1
Interfaces: ve111 ve211 ve311 ve1116 ve2115
Address Family IPv4
Max Routes: 5500
Number of Unicast Routes: 6
Address Family IPv6
Max Routes: 400
Number of Unicast Routes: 6
```

To verify all configured VRFs in detail mode, enter the **show vrf detail** command, as in the following example.

```
device# show vrf detail
Total number of VRFs configured: 2
VRF green, default RD 1:1, Table ID 1
IP Router-Id: 1.1.1.1
Interfaces: Use "show vrf green" to see the list of interfaces
Address Family IPv4
Max Routes: 5500
Number of Unicast Routes: 6
Address Family IPv6
Max Routes: 400
Number of Unicast Routes: 6
VRF red, default RD 10:12, Table ID 2
IP Router-Id: 1.1.17.1
Interfaces:
Use "show vrf red" to see the list of interfaces
Address Family IPv4
Max Routes: 300
Number of Unicast Routes: 2
Address Family IPv6
Max Routes: 70
Number of Unicast Routes: 2
Total number of IPv4 unicast route for all non-default VRF is 8
Total number of IPv6 unicast route for all non-default VRF is 8
```

The following commands display additional information about a specific application, protocol configuration, or protocol state for both the default VRF and user-defined VRFs.

**TABLE 46** Useful show commands

| Default VRF | User-defined VRF |
|---|---|
| **show ip route** | **show ip route vrf** *vrf-name* |
| **show ip ospf neighbor** | **show ip ospf vrf** *vrf-name* **neighbor** |
| **show ip bgp summary** | **show ip bgp vrf** *vrf-name* **summary** |
| Default VRF | User-defined VRF |
| **show ip route** | **show ip route vrf** *vrf-name* |
| **show ip ospf neighbor** | **show ip ospf vrf** *vrf-name* **neighbor** |
| **show ip rip interface** | **show ip rip vrf** *vrf-name* **interface** |
| **show ip bgp summary** | **show ip bgp vrf** *vrf-name* **summary** |

# Removing a VRF configuration

The following examples illustrate a variety of ways by which you can remove a VRF configuration: deleting a VRF instance from a port, deleting an address family from a VRF, and deleting the VRF globally.

To delete a VRF instance from a specific port, use the **no** form of the **vrf** command. This removes all Layer 3 interface bindings from the VRF, and returns the interface to default VRF mode. All IP addresses and protocol configuration on this Layer 3 interface are removed.

```
device(config-if-e1000-1/1)# no vrf forwarding1
All existing IP and IPv6 address will be removed from port 1/1
The port will be returned to default VRF
```

To delete an IPv4 or IPv6 address family from a VRF instance, use the **no** form of the **address-family** command. All configuration related to the address family on all ports of the VRF are removed. Routes allocated to the address family are returned to the global pool.

```
device(config-vrf-customer1)# no address-family ipv4
device(config-vrf-customer1)#
```

To delete a VRF instance globally, use the **no** form of the **vrf** command. All IPv4 or IPv6 addresses are removed from all interfaces.

```
device(config)# no vrf customer1
Warning: All IPv4 and IPv6 addresses (including link-local) from all interfaces in VRF customer1 have been
removed
```

## Configuring the maximum number of routes

You can use the **max-route** command to specify the number of routes held in the routing table per VRF instance, for an IPv4 or IPv6 VRF address family.

If this command is not used, the maximum number of routes is 4294967295. This number does not appear in a running configuration.

1. Enter global configuration mode.

   ```
   device# configure terminal
   ```

2. From global configuration mode, specify a VRF instance (in this example, "myvrf") and enter VRF configuration mode.

   ```
   device(config)# vrf myvrf
   ```

3. Enter the **address-family unicast** command, in this example for IPv4, and enter VRF address-family IPv4 unicast configuration mode.

   ```
   device(config-vrf-myvrf)# address-family ipv4 unicast
   ```

4. Enter the **max-route** command and specify the maximum number of routes to be held in the routing table for this VRF instance, 3600 in this example. (The range is from 1 through 4294967295.)

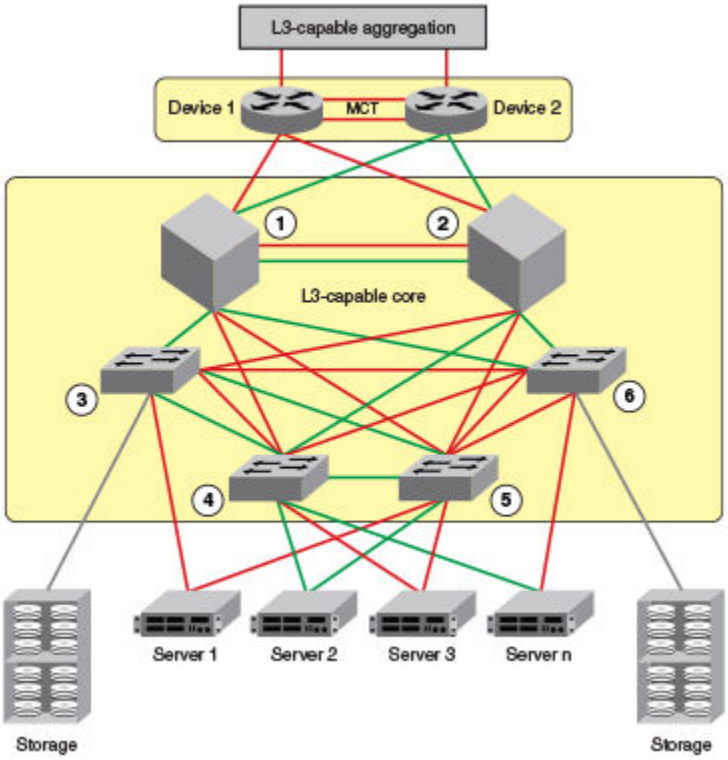
   ```
   device(vrf-myvrf-ipv4-unicast)# max-route 3600
   ```

# Multi-VRF configuration example

The following is an example of a basic Multi-VRF configuration that uses eBGP with OSPF.

The following example topology shows a typical network that uses the Multi-VRF feature to implement Layer 3 VPNs across two directly connected (at Layer 3) Provider Edge (PE) devices. The Customer Edge (CE) devices can be any router or Layer 3 switch that is capable of running one or many dynamic routing protocols such as BGP, OSPF, or RIP, or even simple static routing. In this example, we use two devices that interconnect all four CE routers with a single link between the two of them.

**FIGURE 52** eBGP configured between PE1 and PE2 with OSPF (Area 0) configured between PEs and CEs




1. PE1
2. PE2
3. CE1
4. CE2
5. CE3
6. CE4

Topology details are listed below.

**TABLE 47** Topology details

| Node | Description | Networks | Carries routes . . . | Interfaces |
|---|---|---|---|---|
| PE1 | Aggregation | 10.1.1.0/24<br>10.3.1.0/24 | | 1/0/1<br>1/0/2<br>1/0/3 |
| PE2 | Aggregation | 10.2.1.0/24<br>10.3.1.0/24 | | 2/0/1<br>2/0/2<br>2/0/3 |
| CE1 | Edge | 10.1.1.0/24 | 10.1.2.0/24<br>10.1.3.0/24 | 3/0/1 |
| CE2 | Edge | 10.1.1.0/24 | 10.1.2.0/24 | 4/0/1 |

**TABLE 47** Topology details (continued)

| Node | Description | Networks | Carries routes . . . | Interfaces |
|---|---|---|---|---|
| | | | 10.1.3.0/24 | |
| CE3 | Edge | 10.2.1.0/24 | 10.2.2.0/24<br>10.2.3.0/24 | 5/0/1 |
| CE4 | Edge | 10.2.1.0/24 | 10.2.2.0/24<br>10.2.3.0/24 | 6/0/1 |

- Traffic is separated by VRF "green" on VLAN 10 and VRF "red" on VLAN 20.
- eBGP and OSPF (Area 0) are used to connect aggregation switches PE1 and PE2 to a Layer 3-capable aggregation VCS Fabric.
- iBGP and OSPF (Area 0) are used to connect the aggregation switches to the CE switches.
- Alternatively, with only OSPF used, Areas 1 and 2 could carry traffic between the PEs and CEs.

The following configuration examples for PE1, PE2, CE1, CE2, CE3, and CE4 implement the topology above.

NOTE
The single link between the two PEs could also be replaced by a Layer 2 switched network if direct physical connection between the PEs is not possible. The only requirement for the connections is that the two PEs be directly connected at Layer 3.

In the example topology, because two different VLANs (10 and 20) have overlapping IP address ranges, communication within each customer's VPN across the two PE routers (that is, between CE1 and CE4, and between CE2 and CE3) must be separated by means of two different VRFs ("green" and "red").

# Multi-VRF with eBGP and OSPF: Configuring PE1

VLANs 10 and 20 are created as links on a tagged port (e 1/1) between PE1 and PE2.

Two VRFs ("red" and "green") are defined, with each having a unique (optional) Route Distinguisher (RD). VRF "green" is assigned an RD value of 10:10, and VRF "red" is assigned an RD value of 20:20.

In the eBGP configuration, PE1 is defined in Local AS 1. VRFs "green" and "red" are configured, and both have the same IP network address assigned (10.3.1.2/24). This is possible because each of the BGP VRF instances has its own BGP tables. This is also the same IP network address that will be assigned to VRFs "green" and "red" on PE2 within Local AS 2. Redistribution of OSPF routes from PE1's CE peers is enabled to all for their advertisement to PE2.

Both VRFs are configured in Area 0 and are directed to redistribute their routes to BGP. The physical interfaces to the CEs are assigned to the appropriate VRF and are configured with the same network address (10.1.1.1/24) and OSPF Area 0.

The virtual Interfaces (Ve 10 and Ve 20) are configured with the same network address (10.3.1.1/24) and for VRF forwarding in the appropriate VRF ("green" or "red").

1. In global configuration mode, create VLANs 10 and 20.

```
device(config)# vlan 10
device(config-vlan-10)# exit
device(config)# vlan 20
```

2. (Optional) Configure the VLAN as tagged on an Ethernet interface, providing communication security for peers.

```
device(config-vlan-10)# tagged e 1/1
```

3. In interface subtype configuration mode, create a virtual Ethernet routing interface for the VLAN.

```
device(config-vlan-10)# router-interface ve 10
```

4. Repeat the above steps as appropriate for remaining physical, VLAN, and virtual Ethernet interfaces.
5. Create VRF "green" and assign a Route Distinguisher (optional).

NOTE
This is done for every VRF instance. A Route Distinguisher is optional. Use the **address-family ipv6 unicast** command for IPv6 addresses. Also, you can use the **max-route** command, which helps restrict the maximum number of routes per VRF.

6. Configure VRF "red" and exit the VRF configuration.

```
device(config-vrf-green)# vrf red
device(config-vrf-red)# rd 20:20
device(config-vrf-red)# exit
```

7. In global configuration mode, enable BGP routing and configure the following in this IPv4 example.
   a) Enable BGP routing.

```
device(config)# router bgp
```

   b) Assign a Local AS number.

```
device(config-bgp)# local-as 1
```

   c) Enable IPv4 unicast address-family mode for VRF "green."

```
device(config-bgp)# address-family ipv4 unicast vrf green
```

   d) Assign Remote AS 2 as a neighbor with the specified address.

```
device(config-bgp-ipv4u-vrf)# neighbor 10.3.1.2 remote-as 2
```

   e) Assign the appropriate network.

```
device(config-bgp-ipv4u-vrf)# network 10.3.1.0/24
```

   f) Redistribute the OSPF routes into BGP4, specifying the types of routes to be distributed, then exit the address family configuration.

```
device(config-bgp-ipv4u-vrf)# redistribute ospf match internal
device(config-bgp-ipv4u-vrf)# redistribute ospf match external1
device(config-bgp-ipv4u-vrf)# redistribute ospf match external2
device(config-bgp-ipv4u-vrf)# exit
```

8. Repeat as above VRF "red."

```
device(config)# router bgp
device(config-bgp)# address-family ipv4 unicast vrf red
device(config-bgp-ipv4u-vrf)# neighbor 10.3.1.2 remote-as 2
device(config-bgp-ipv4u-vrf)# network 10.3.1.0/24
device(config-bgp-ipv4u-vrf)# redistribute ospf match internal
device(config-bgp-ipv4u-vrf)# redistribute ospf match external1
device(config-bgp-ipv4u-vrf)# redistribute ospf match external2
device(config-bgp-ipv4u-vrf)# exit
```

9. Enable OSPF routing for VRF "green" and configure the following.
   a) Enable OSPF.

```
device(config)# router ospf vrf green
```

   b) Assign Area 0.

```
device(config-ospf-router-vrf-green)# area 0
```

   c) Redistribute the OSPF routes into BGP4 and exit the VRF configuration.

```
device(config-ospf-router-vrf-green)# redistribute bgp
device(config-ospf-router-vrf-green)# exit
device(config-ospf-router)# exit
```

10. Repeat as above for VRF "red".

```
device(config)# router ospf vrf red
device(config-ospf-router-vrf-red)# area 0
device(config-ospf-router-vrf-red)# redistribute bgp
device(config-ospf-router-vrf-red)# exit
```

11. Configure the Ethernet interfaces as appropriate, as in the following example.
   a) Assign an interface to VRF instance "green" and enable forwarding.

```
device(config)# interface ethernet 1/2
device(config-if-e1000-1/2)# vrf forwarding green
```

   b) Assign Area 0.

```
device(config-if-e1000-1/2)# ip ospf area 0
```

   c) Assign an IP network.

```
device(config-if-e1000-1/2)# ip address 10.1.1.1/24
```

   d) Repeat as above for another Ethernet interface and VRF "red" and exit the interface configuration.

```
device(config-if-e1000-1/2)# interface ethernet 1/3
device(config-if-e1000-1/3)# vrf forwarding red
device(config-if-e1000-1/3)# ip ospf area 0
device(config-if-e1000-1/3)# ip address 10.1.1.1/24
device(config-if-e1000-1/3)# exit
```

12. Configure the VE interfaces for the appropriate VRF and network.
   a) Configure VE 10, corresponding to VLAN 10.

```
device(config)# interface ve 10
device(config-vif-10)# vrf forwarding green
device(config-vif-10)# ip address 10.3.1.1/24
```

   b) Repeat the above for VE 20, corresponding to VLAN 20.

```
device(config-vif-10)# interface ve 20
device(config-vif-20)# vrf forwarding red
device(config-vif-20)# ip address 10.3.1.1/24
```

# Multi-VRF with eBGP and OSPF: Configuring PE2

The PE2 configuration is a mirror image of the PE1 configuration. The only difference is that the BGP neighbor on the corresponding interface has an IP address of 10.3.1.1. This is used in the BGP configuration.

The following summarizes the configuration on PE2.

```
device(config)# vlan 10
device(config-vlan-10)# tagged e 1/1
device(config-vlan-10)# router-interface ve 10
device(config-vlan-10)# vlan 20
device(config-vlan-20)# tagged e 1/1
device(config-vlan-20)# router-interface ve 20
device(config-vlan-20)# exit-vrf
device(config)# vrf green
device(config-vrf-green) rd 10:10
device(config-vrf-green) vrf red
device(config-vrf-red) rd 20:20
device(config-vrf-red) exit
device(config)# router bgp
device(config-bgp)# local-as 1
device(config-bgp)# address-family ipv4 unicast vrf green
device(config-bgp-ipv4u-vrf)# neighbor 10.3.1.1 remote-as 2
device(config-bgp-ipv4u-vrf)# network 10.3.1.0/24
device(config-bgp-ipv4u-vrf)# redistribute ospf match internal
device(config-bgp-ipv4u-vrf)# redistribute ospf match external1
device(config-bgp-ipv4u-vrf)# redistribute ospf match external2
device(config-bgp-ipv4u-vrf)# exit
device(config)# router bgp
device(config-bgp)# address-family ipv4 unicast vrf red
device(config-bgp-ipv4u-vrf)# neighbor 10.3.1.1 remote-as 2
device(config-bgp-ipv4u-vrf)# network 10.3.1.0/24
device(config-bgp-ipv4u-vrf)# redistribute ospf match internal
device(config-bgp-ipv4u-vrf)# redistribute ospf match external1
device(config-bgp-ipv4u-vrf)# redistribute ospf match external2
device(config-bgp-ipv4u-vrf)# exit
device(config)# router ospf vrf green
device(config-ospf-router-vrf-green)# area 0
device(config-ospf-router-vrf-green)# redistribute bgp
device(config-ospf-router-vrf-green)# exit
device(config)# router ospf vrf red
device(config-ospf-router-vrf-red)# area 0
device(config-ospf-router-vrf-red)# redistribute bgp
device(config-ospf-router-vrf-red)# exit
device(config)# interface ethernet 1/2
device(config-if-e1000-1/2)# vrf forwarding green
device(config-if-e1000-1/2)# ip ospf area 0
device(config-if-e1000-1/2)# ip address 10.1.1.1/24
device(config-if-e1000-1/2)# interface ethernet 1/3
device(config-if-e1000-1/3)# vrf forwarding red
device(config-if-e1000-1/3)# ip ospf area 0
device(config-if-e1000-1/3)# ip address 10.1.1.1/24
device(config-if-e1000-1/3)# exit
device(config)# interface ve 10
device(config-vif-10)# vrf forwarding green
device(config-vif-10)# ip address 10.3.1.1/24
device(config-vif-10)# interface ve 20
device(config-vif-10)# vrf forwarding red
device(config-vif-10)# ip address 10.3.1.1/24
```

# Multi-VRF with eBGP and OSPF: Configuring CE1 and CE2

The CE1 and CE2 router configurations are exactly the same. Both are configured in OSPF Area 0 with route redistribution enabled. The IP addresses: 10.1.2.1/32 and 10.1.3.1/32 are configured for the Loopback1 interface allowing them to carry routes from these networks.

1. Enable OSPF routing.

```
device(config)# router ospf
```

2. Assign Area 0.

```
device(config-ospf-router)# area 0
```

3. Redistribute connected routes into OSPF and exit the OSPF configuration.

```
device(config-ospf-router)# redistribute connected
device(config-ospf-router)# exit
device(config)#
```

4. Configure a loopback interface to support the appropriate networks.

```
device(config)# interface loopback 1
device(config-lbif-1)# ip address 10.1.2.1/32
device(config-lbif-1)# ip address 10.1.3.1/32
```

5. Configure an Ethernet interface, assign it to Area 0, and assign it to the appropriate network.

```
device(config-lbif-1)# interface ethernet 1/1
device(config-if-e1000-1/1)# ip ospf area 0
device(config-if-e1000-1/1)# ip address 10.1.1.2/24
```

# Multi-VRF with eBGP and OSPF: Configuring CE3 and CE4

The CE3 and CE4 router configurations are exactly the same. Both are configured in OSPF Area 0 with route redistribution enabled. The IP addresses: 10.2.2.1/32 and 10.2.3.1/32 are configured for the Loopback1 interface allowing them to carry routes from these networks.

The following summarizes the configuration.

```
device(config)# router ospf
device(config-ospf-router)# area 0
device(config-ospf-router)# redistribute connected
device(config-ospf-router)# exit
device(config)# interface loopback 1
device(config-lbif-1)# ip address 10.2.2.1/32
device(config-lbif-1)# ip address 10.2.3.1/32
device(config-lbif-1)# interface ethernet 1/1
device(config-if-e1000-1/1)# ip ospf area 0
device(config-if-e1000-1/1)# ip address 10.2.1.2/24
```

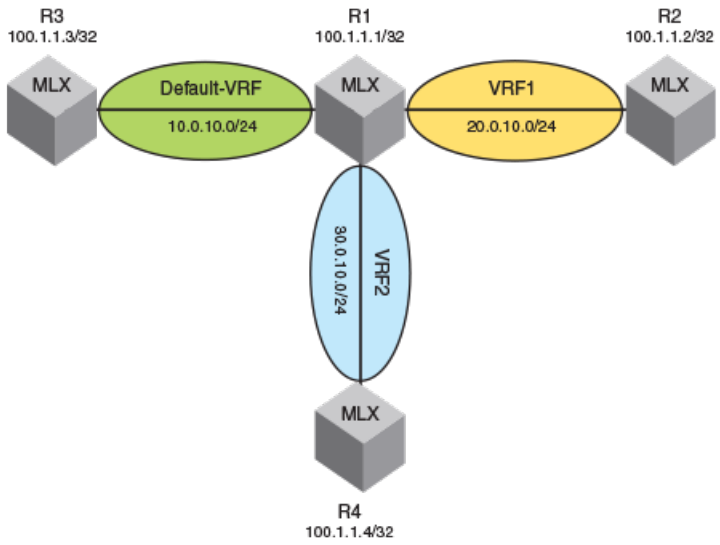
# Inter-VRF Routing



## Inter-VRF routing overview

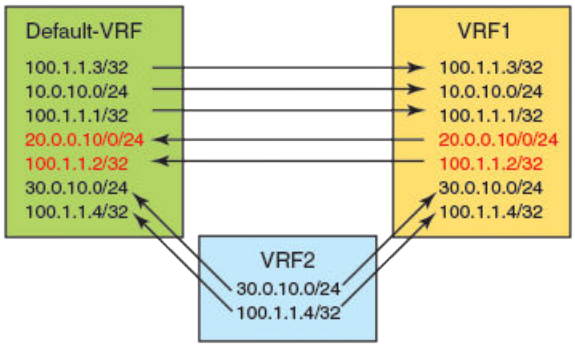
Inter-VRF routing feature permits routes from one VRF to import into other VRFs. This feature is useful in cases where all the VRFs share the same path to reach the external domain, but each VRF can still keep its internal routing information limited to its own VRF. Currently, the devices permit static routes to be configured across VRFs. User can configure to import routes from one VRF to other VRFs through configuration. The following figure depicts a network using inter-VRF to provide connectivity among sites that belong to multiple VPNs. To share the VPN routing table information with remote PEs, each PE creates separate virtual interfaces and runs different instances of the PE-PE routing protocol for each VRF.

**FIGURE 53** A Network deploying Inter-VRF




Following are the route entries and routing tables in Router R1.

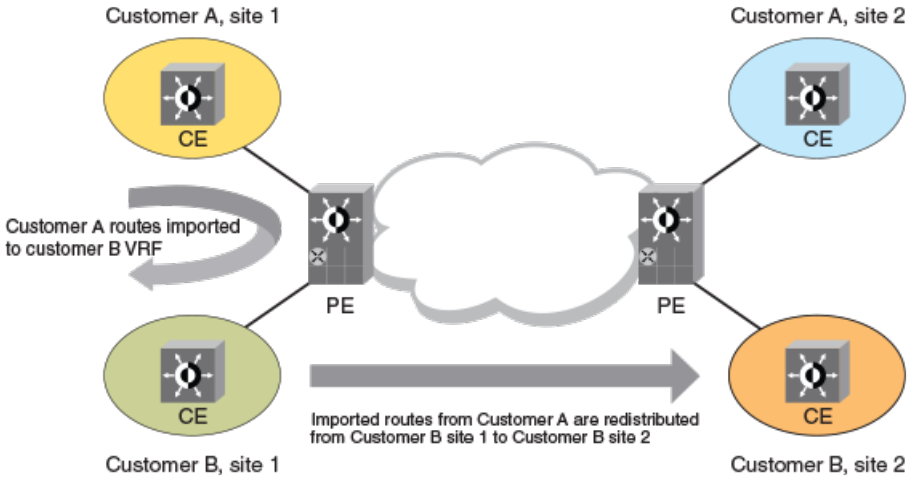
**FIGURE 54** Router R1 route-entries and route-tables




# Features and benefits

Inter-VRF routing feature allows customers to selectively access each other's networks through configuration. It allows all VRFs to share the same path to the external domain while keeping internal routing information separate.

**FIGURE 55** Inter-VRF routing topology




To import routes from multiple VRFs, multiple import commands need to be defined. The filtering criteria for routes to be imported are specified using route-maps by the user. Routes can be filtered based on BGP attributes, interfaces, IP addresses, next hops, metrics, metric types, protocols, route types and tags which are supported by our existing route-map infrastructure.

Routes from multiple (default 50) VRFs can be selectively imported into a target destination VRF. Imported routes can be redistributed using routing protocols.

Configuring IPv6 inter-VRF routing is very similar to configuring inter-VRF routing for IPv4. The behavior and CLI syntax of route-maps is the same between IPv4 and IPv6 address families.

The route-map attributes that are used for filtering the routes are:

**TABLE 48** IPv4 Route-map handling

| Attributes used for filtering routes | Attributes set using a route map |
|---|---|
| IP address (Prefix list, Access list) | |
| Next hop (Prefix list, Access list) | Metric value |
| Metric value | Nexthop |
| Tag type | Distance |
| Route type | Tag |
| BGP attributes (AS path, Community, Ext community access list) | |
| Interface type | |
| Protocol type | |

**TABLE 49** IPv6 Route-map handling

| Attributes used for filtering routes | Attributes set using a route map |
|---|---|
| IP address (Prefix list) | |
| Next hop (Prefix list) | Metric value |
| Metric value | Nexthop |
| Tag type | Distance |
| Route type | Tag |
| BGP attributes (AS path, Community, Ext community access list) | |
| Interface type | |
| Protocol type | |

# Configuration considerations

These are the things to consider while configuring the device:

- Import configuration commands allow specifying a non-existing source VRF. Routes will be imported dynamically when the source VRF is created.
- If the route-map is empty or does not exist, importing will happen but no actions are applied since there are no rules in the route-map.
- If you change the configuration of a route-map, then all the VRFs which are configured to use this route-map will be processed again.

## Tie breaker rules

The rules in the sequence below apply to break the tie when the same routes are imported from multiple VRFs including the local route:

- If routes are originated from different protocols, then the protocol with the best administrative distance will be used to break the tie.
- If the routes' origin (protocol) are the same, then the metric value will be used to break the tie.
- If the metric value is the same, then routes learned in local VRF will be used to break the tie.
- If the metric value is the same, and a local VRF route is not available, then the lowest nexthop address will be used to break the tie.
- If the nexthop address is the same, then the oldest route will be used to break the tie.

# Maximum route limitations

When importing routes from other VRFs, there may be a chance that routes are not added due to a limitation on the number of routes that the destination VRF can support. This may happen in the following cases:

1. The source VRF is importing more routes than the destination VRF can support.
2. The destination VRF is configured to limit the number of routes with a configuration command such as address-family **ipv4 max-route** or **address-family ipv6 max-route** .
3. While processing the route-map changes, you exceed the number of routes that the VRF can support because you process the new set of routes before deleting the old set of routes.

   For any of the above situations, execute the **clear ip route VRF dest-vrf-name** followed by the **import***src-vrf-name* command to recover.

# BGP L3VPN configuration

## No advertising of inter-vrf-leaked routes out to a Layer 3 VPN

When a route is imported from one VRF to another VRF using the inter-VRF route leaking feature, the imported route in the destination VRF can be redistributed into VRF-BGP. It can also be advertised out to the Layer 3 VPN network.

The advertised Layer 3 VPN route originally imported from a different VRF uses the export route-target(s) from the destination VRF. This feature does not automatically block inter-VRF leaked routes from being advertised out to a Layer 3 VPN network. To block inter-VRF leaked routes use the **no** version of the export-vrf-leaked-routes command. The default behavior is backward compatible. A BGP option has been added to disable backward compatibility.

Starting in 5.8.00d and 5.9.00a, the export-vrf-leaked-routes command also disables inter-VRF-leaking of BGP routes with LSP next-hop for IPv4 routes. Inter-VRF-leaking of BGP routes with LSP next-hop for IPv6 routes is not supported.

# Configuring Inter-VRF routing

The following configuration steps allow the VRF VPN to import IPv4 routes from the **default-vrf brcd-sj**.

```
device(config)#vrf vpn
device(config-vrf-vpn)#address-family ipv4
device(config-vrf-vpn-ipv4)#import routes vrf default-vrf route-map brcd-sj
```

The following configuration allows the default-vrf to import IPv4 routes from the non-default VRF VPN after satisfying conditions specified in the route-map brcd-sj.

```
device(config)#ip import routes vrf vpn route-map brcd-sj
```

From non-default VRF, user can configure the command in address-family mode.

**Syntax: import routes vrf** *vrf-name* **route-map** *route-map-name*

This command imports the IPv4 routes from src-vrf to dest-vrf. The route-map import-map is applied while importing the routes.

**Syntax: import routes vrf** *src-vrf* **route-map** *import-map*

From default VRF the commands for IPv4 and IPv6 are as below.

These commands import the IPv4 and IPv6 routes from src-vrf to default-vrf using the route-map import-map.

```
device(config)#ip import routes vrf src-vrf route-map import-map>
device(config)#ipv6 import routes vrf src-vrf route-map import-map
```

**TABLE 50** Route-map for non-default and default VRF

| This field... | Displays |
|---|---|
| src-vrf | The source VRF from where the routes have been imported to the destination VRF. The "-" in the src-vrf column output denotes the route is local route. |
| import-map | Route-map which has clauses to filter the routes coming from src-vrf. |
| Defaults | By default no routes will be imported in to dest-vrf. |
| Range | User can configure a maximum of 50 import commands for a given VRF per address-family. If user tries to configure more than 50 commands then the configuration will be rejected and an error message will be generated. |
| No | The no command removes the configuration and all the routes imported from src-vrf will be removed in the dest-vrf. |

NOTE
Warning message will be displayed when import route is issued for non-existing VRF as follows: "VRF *vrf-name* is not created yet"

NOTE
Warning message will be displayed when VRF is getting deleted by the user and exports routes to other VRFs: "All IPv4 and IPv6 export routes in VRF one have been removed"

# Blocking inter-VRF leaked routes from being advertised for the IPv4 VPN unicast address-family

This task blocks inter-VRF leaked routes from being advertised out to a Layer 3 VPN network. for the IPv4 VPN unicast address-family.

1. Enter the **configure terminal** command to access global configuration mode.

   ```
   device# configure terminal
   ```

2. Enter the **router bgp** command to enable BGP routing.

   ```
   device(config)# router bgp
   ```

3. Enter the **local-as** command to configure the autonomous system number (ASN) in which your device resides.

   ```
   device(config-bgp)# local-as 65520
   ```

4. Enter the **address-family vpnv4 unicast** command to enter BGP address-family IPv4 VPN unicast configuration mode .

   ```
   device(config-bgp)# address-family vpnv4 unicast
   ```

5. Enter the **no export-vrf-leaked-routes** command to block inter-VRF leaked routes from being advertised out to a Layer 3 VPN network.

   ```
   device(config-bgp-vpn4u)# no export-vrf-leaked-routes
   ```

This example blocks inter-VRF leaked routes from being advertised out to a Layer 3 VPN network. for the IPv4 VPN unicast address-family.

```
device# configure terminal
device(config)# router bgp
device(config-bgp)# local-as 65520
device(config-bgp)# address-family vpnv4 unicast
device(config-bgp-vpn4u)# no export-vrf-leaked-routes
```

## Blocking inter-VRF leaked routes from being advertised for the IPv6 VPN unicast address-family

This task blocks inter-VRF leaked routes from being advertised out to a Layer 3 VPN network. for the IPv6 VPN unicast address-family.

1. Enter the **configure terminal** command to access global configuration mode.

   ```
   device# configure terminal
   ```

2. Enter the **router bgp** command to enable BGP routing.

   ```
   device(config)# router bgp
   ```

3. Enter the **local-as** command to configure the autonomous system number (ASN) in which your device resides.

   ```
   device(config-bgp)# local-as 65520
   ```

4. Enter the **address-family vpnv6 unicast** command to enter BGP address-family IPv6 VPN unicast configuration mode .

   ```
   device(config-bgp)# address-family vpnv6 unicast
   ```

5. Enter the **no export-vrf-leaked-routes** command to block inter-VRF leaked routes from being advertised out to a Layer 3 VPN network.

   ```
   device(config-bgp-vpn4u)# no export-vrf-leaked-routes
   ```

This example blocks inter-VRF leaked routes from being advertised out to a Layer 3 VPN network. for the IPv6 VPN unicast address-family.

```
device# configure terminal
device(config)# router bgp
device(config-bgp)# local-as 65520
device(config-bgp)# address-family vpnv6 unicast
device(config-bgp-vpnv6)# no export-vrf-leaked-routes
```

## Show commands

Use the following commands to display the IPv4 and IPv6 routing configuration on the device and to include the maximum allowed import VRFs.

```
device# show ip
```

This command will display the maximum IPv4 allowed import VRFs and list of import VRFs to default-vrf. The following is an example of this enhancement to **show ip** .

```
device(config)#show ip
Global Settings
IP CAM Mode: static IPVPN CAM Mode: static
  ttl: 64, arp-age: 10, bootp-relay-max-hops: 4, icmp-error-rate: 400
  IP Router-Id: 10.0.0.1   load-sharing path: 4
  enabled : UDP-Broadcast-Forwarding  ICMP-Redirect  ICMP-MPLS-Response  Source-Route  Load-Sharing
```

```
 RARP  RIP  BGP4  IS-IS  OSPF  VRRP
  disabled: Directed-Broadcast-Forwarding  drop-arp-pending-packets  IRDP  Proxy-ARP  RPF-Check  RPF
-Exclude-Default  VRRP-Extended  VSRP
Configured Static Routes: 15
Maximum allowed import VRFs: 2048
```

device **# show ipv6**

This command will display the maximum IPv6 allowed import VRFs and list of IPv6 import VRFs to default-vrf.

## *Displaying the IP route table for a specified VRF*

To display the IP routes for a specified VRF, enter the following command at any CLI level for IPv4 and IPv6 respectively.

```
device# show ip route vrf one
device# show ipv6 route vrf one
```

**Syntax: show ip route vrf vrf-name** [ *num* ] | [ *ip-addr* ] | [ **bgp** ] | [ **connected** ] | [ **isis** ] | [ **ospf** ] | [ **rip** ] | [ **static** ] | [ **tags** ]

**Syntax: show ipv6 route vrf vrf-name** [ *num* ] | [ *ip-addr* ] | [ **bgp** ] | [ **connected** ] | [ **isis** ] | [ **ospf** ] | [ **rip** ] | [ **static** ] | [ **tags** ] | **nexthop** *nexthop_id* | **ref-routes**

The *vrf-name* parameter specifies the VRF for which you want to display the IP routes.

The **nexthop** option displays the next-hop information for all next hops in the routing table or for a specific entry. The *nexthop_id* parameter is a specific nexthop entry from the next hop table.

The **ref-routes** option allows you to display IPv6 routes in the forwarding table that refer to the specified nexthop entry.

The following table lists the information displayed by the **show ip/ipv6 route vrf** command.

**TABLE 51** CLI display of IP route-table

| This field... | Displays |
|---|---|
| Total number of IP routes | The total number of IP routes that are in the specified VRP routing-table. |
| Destination | The destination network of the route. |
| NetMask | The network mask of the destination address. |
| Gateway | The next-hop router. |
| Port | The port through which this Extreme device sends packets to reach the route's destination. |
| Cost | The route's cost. |
| Type | The route type, which can be one of the following:<br>• B - The route was learned from BGP.<br>• D - The destination is directly connected to this Extreme device.<br>• R - The route was learned from RIP.<br>• S - The route is a static route.<br>• * - The route is a candidate default route.<br>• O - The route is an OSPF route. Unless you use the **ospf** option to display the route table, "O" is used for all OSPF routes. If you do use the **ospf** option, the following type codes are used:<br>– O - OSPF intra area route (within the same area).<br>– IA - The route is an OSPF inter area route (a route that passes from one area into another).<br>– E1 - The route is an OSPF external type 1 route.<br>– E2 - The route is an OSPF external type 2 route. |
| Uptime | The amount of time since the route was last modified. The format of this |

**TABLE 51** CLI display of IP route-table (continued)

| This field... | Displays |
|---|---|
| | display parameter may change depending upon the age of the route to include<br>the seconds (s), minutes (m), hours (h), and days (d), as described in the<br>following:<br>400d - Only days (d) displayed<br>20d23h - days (d) and hours (h) displayed<br>14h33m - hours (h) and minutes (m) displayed<br>10m59s - minutes (m) and seconds (s) displayed |
| src-vrf | The source VRF from where the routes have been imported to the destination VRF. The "-" in the src-vrf column output denotes the route is local route. |

## *Displaying the IP route table for a specified VRF import, local or summary*

To display the IP routes for a specified VRF import or from local VRF or all VRFs summary, enter the following command at any CLI level for IPv4 and IPv6 respectively.

```
device# show ip route vrf one import/local/summary
device# show ipv6 route vrf one import/local/summary
```

**Syntax: show ip route vrf** [ **import** | **local** | **summary** ] **vrf-name** [ *num* ] | [ *ip-addr* ] | [ **bgp** ] | [ **connected** ] | [ **isis** ] | [ **ospf** ] | [ **rip** ] | [ **static** ] | [ **tags** ]

**Syntax: show ipv6 route vrf** [ **import** | **local** | **summary** ] **vrf-name** [ *num* ] | [ *ip-addr* ] | [ **bgp** ] | [ **connected** ] | [ **isis** ] | [ **ospf** ] | [ **rip** ] | [ **static** ] | [ **tags** ] | **nexthop** *nexthop_id* | **ref-routes**

The *vrf-name* parameter specifies the VRF for which you want to display the IP routes.

The **nexthop** option displays the next-hop information for all next hops in the routing table or for a specific entry. The *nexthop_id* parameter is a specific nexthop entry from the next hop table.

The **ref-routes** option allows you to display IPv6 routes in the forwarding table that refer to the specified nexthop entry.

Displaying the IP route table for a specified VRF on page 495 lists the information displayed by these commands.

## *Displaying IPv4 routes in VRF one*

To display IP information for a specified VRF, enter the following command at any level of the CLI.

```
device(config)#show ip route vrf one
Total number of IP routes: 4
Type Codes - B:BGP D:Connected I:ISIS O:OSPF R:RIP S:Static; Cost - Dist/Metric
BGP  Codes - i:iBGP e:eBGP
ISIS Codes - L1:Level-1 L2:Level-2
OSPF Codes - i:Inter Area 1:External Type 1 2:External Type 2 s:Sham Link
        Destination        Gateway         Port          Cost          Type Uptime  src-vrf
1       10.0.0.0/8         10.25.104.1     eth1/1          20          O       4d1h       c
2       10.1.0.0/16        10.25.103.1     eth2/1          20          O       3d1h       b
3       10.20.0.0/8        10.25.104.1     eth1/1          20          O       4d1h       -
4       10.40.0.0/8        10.25.105.1     eth2/1          20          O       3d1h       default
```

## Displaying IPv4 routes in VRF one import

To display IP information for a specified VRF import, enter the following command at any level of the CLI.

```
device(config)#show ip route vrf one import
Type Codes - B:BGP D:Connected I:ISIS O:OSPF R:RIP S:Static; Cost - Dist/Metric
BGP  Codes - i:iBGP e:eBGP
ISIS Codes - L1:Level-1 L2:Level-2
OSPF Codes - i:Inter Area 1:External Type 1 2:External Type 2 s:Sham Link
        Destination        Gateway         Port          Cost          Type Uptime  src-vrf
1       10.0.0.0/8         10.25.104.1     eth1/1          20          O       4d1h       c
2       10.1.0.0/16        10.25.103.1     eth2/1          20          O       3d1h       b
3       10.40.0.0/8        10.25.105.1     eth2/1          20          O       3d1h       default
```

## Displaying outputs routed from other VRF

To display all the other VRFs from where the routes will be imported into this VRF, enter the following command at any level of the CLI.

**Syntax: show ip vrf** *vrf-name*

```
device#show ip vrf one
VRF one, default RD 1001:11, Table ID 2 IFL ID 131070
Label: 500000,  Label-Switched Mode: OFF
IP Router-Id: 10.1.1.2
  Interfaces:
    e2/8
  No Export VPN route-target communities
  No Import VPN route-target communities
  No import route-map
  No export route-map
  Address Family IPv4
    Max Routes: 5120
    Imports routes from VRF: a, b, c, d
    No Export VPN route-target communities
    No Import VPN route-target communities
 Address Family IPv6
    Max Routes: 128
    Imports routes from VRF: a, b
    No Export VPN route-target communities
    No Import VPN route-target communities
```

## Configuring routes from multiple VRFs

This example shows the sequence of commands in configuring routes from multiple VRFs.

```
device# vrf a
device# rd 1111:11
device# address-family ipv4
device# import routes vrf b route-map import-map
device# import routes vrf c route-map import-map
device# exit-address-family
device# address-family ipv6
device# import routes vrf b route-map import-v6map
device# exit-address-family
device# exit-vrf
device# route-map import-map permit 10
device# match ip address prefix-list export
device# route-map import-map permit 15
device# match ip address prefix-list loop
```

NOTE

If the configuration of a route-map is changed, then the VRFs which are configured to use the respective route-map will be processed again.

### *Displaying IPv6 routes in VRF one from local*

To display IP information for a specified VRF for IPv6 routes from the local VRF, enter the following command at any level of the CLI.

```
device(config)#show ipv6 route vrf one local
Type Codes - B:BGP C:Connected I:ISIS L:Local O:OSPF R:RIP S:Static
BGP  Codes - i:iBGP e:eBGP
ISIS Codes - L1:Level-1 L2:Level-2
OSPF Codes - i:Inter Area 1:External Type 1 2:External Type 2
Destination       Gateway                   Port   Cost   Type   Uptime   src-vrf
   3000:/8  fe80::768e:f8ff:fe2a:d063 eth1/3   20     O      4d1h       -
```

### *Displaying IPv6 imported routes summary*

To display IP information for all VRFs summary for IPv6 routes, enter the following command at any level of the CLI.

```
device(config)#show ipv6 route vrf one import vrf summary
IPv6 Routing Table - 10 entries:
  0 connected, 0 static, 0 RIP, 10 OSPF, 0 BGP, 0 ISIS
  Number of prefixes:
  /64:10
```

**NOTE**

An error will be displayed when an attempt to match the source VRF name with the import VRF name.

### *Displaying IPv6 routes in VRF one imported from another VRF*

To display IP information for a specified VRF import routes from VRF two, enter the following command at any level of the CLI.

```
device(config)#show ipv6 route vrf one import vrf b
Type Codes - B:BGP C:Connected I:ISIS L:Local O:OSPF R:RIP S:Static
BGP  Codes - i:iBGP e:eBGP
ISIS Codes - L1:Level-1 L2:Level-2
OSPF Codes - i:Inter Area 1:External Type 1 2:External Type 2
Destination       Gateway                   Port   Cost   Type   Uptime   src-vrf
        2000:/8  fe80::768e:f8ff:fe2a:d062 eth1/2 20     O      4d1h       -
```

# Clearing IP routes

You can clear the entire routing-table or specific individual routes as needed.

To clear all routes from the IPv4 routing-table, enter the following command at any level of the CLI.

```
device# clear ip route
```

To clear route 10.157.22.0/24 from the IPv4 routing table, enter:

```
device# clear ip route 10.157.22.0/24
```

**Syntax: clear** [ **ip** | **ipv6** ] **route** [ *ip-addrip-mask* | *ip-addr/mask-bits* ] [ **import** | **local** | **import vrf** *vrf-name* ] | **nexthop** *nexthop_id*

The following examples illustrate the use of the **clear route** command:

```
device# clear ip route vrf one [<IP address
> <Mask
>] import
```

Clears the imported IPv4 routes from all other VRFs. When this command is issued with an IP address and mask, then the imported routes matching the address and mask from other VRFs are cleared.

```
device# clear ip route vrf one [<IP address
> <Mask
> local]
```

Clears the IPv4 routes from specific VRF. When this command is issued with an IP address and mask only, then the local routes matching the address and mask are cleared, otherwise this option is not available.

```
device# clear ip route vrf one [<IP address
> <Mask
>] import vrf two
```

Clears the imported IPv4 routes from VRF two. When this command is issued with an IP address and mask, then imported routes matching the address and mask from VRF two are cleared.

```
device# clear ip route vrf one [<IP address
> <Mask
>] import vrf default-vrf
```

Clears the imported IPv4 routes from the default-vrf. When this command is issued with an IP address and mask, then imported routes matching the address and mask from the default VRF are cleared.

```
device# clear ipv6 route
```

Clears all routes from the IPv6 routing-table.

```
device# clear ipv6 route 10.157.22.0/24
```

Clears route 10.157.22.0/24 from the IPv6 routing-table.

```
device# clear ipv6 route vrf one [IPv6 addr/Prefix length
] import vrf two
```

Clears the imported IPv6 routes from VRF two. When this command is issued with an IPv6 address and prefix length, then the imported routes matching the IPv6 address and prefix length from VRF two are removed.

```
device# clear ipv6 route vrf one [IPv6 addr/Prefix length
] import vrf default-vrf
```

Clears the imported IPv6 routes from the default-vrf. When this command is issued with an IPv6 address and prefix length, then the imported routes matching the IPv6 address and prefix length from the default VRF are removed.

```
device# clear ipv6 route nexthop nexthop_ID
```

Clears the imported IPv6 routes for the specified nexthop ID on the interface module (LP).

# Configuring the number of VRFs for IPv4 and IPv6

To limit the number of imported IPv4 or IPv6 routes into any VRF including the default VRF, the following command is available in the global configuration mode and not available in any individual VRF mode. Changes in the value in the global configuration mode will be effective in all VRFs.

```
device(config)# [ip|ipv6] max-import-vrfs 1-2048
```

The **no** command will set the value to the default value, which is 50.

If you configure **ip max-import-vrfs** to a number which is less than the currently imported routes in the IPv4 or IPv6 address family for any VRF, then the following error will be displayed and the configuration will not be accepted.

```
device(config)# [ip|ipv6
] max-import-vrfs 2
Error: VRF one has 3 import commands configured in ipv4/ipv6 address families
```

The configured non-default value of **ip/ipv6 max-import-vrfs** may be displayed using the **show ip** or **show ip vrf-name** commands.

NOTE
A system maximum of 1000 import commands (including all VRFs, IPv4 and IPv6 address families) can be defined.

NOTE
Using the **system-max ip-vrf-route** command, the number of IPv4 routes per VRF instance is limited to 1024. Using the **system-max ipv6-vrf-route** command, the number of IPv6 routes per VRF instance is limited to 8192.

# Modified CLI commands

The Inter-vrf routing feature makes it possible to import OSPF routes from one VRF to another VRF. There may be a need to advertise the imported OSPF routes back to the OSPF domain as external routes. This requires redistribution of OSPF into OSPF again, which was not supported In prior releases. With the introduction of redistribution, the following configuration is supported:

```
device(config-ospf-router)#redistribute ospf route-map <route-map-name>
  bgp         Border Gateway Protocol (BGP)
  connected   Connected
  isis        Intermediate System to Intermediate System (IS-IS)
  rip         Routing Information Protocol (RIP)
  static      Static routes
```

ospf OSPF routes (new addition)

This command is applicable to OSPF, RIP and BGP. Currently IS-IS does not support VRFs and it is not possible to have IS-IS running in multiple VRFs.

If you configure to import the same protocol routes into the same protocol, then RTM will send back protocol routes belonging to other VRFs.

Redistribution of a protocol into itself is supported for the following protocols:

- IPv4

1. OSPF->OSPF
   a) route-map option
2. BGP->BGP
   a) route-map option
   b) Metric option
3. RIP->RIP
   a) route-map option
   b) Metric option
      - IPv6

# OSPFv2

# OSPFv2 overview

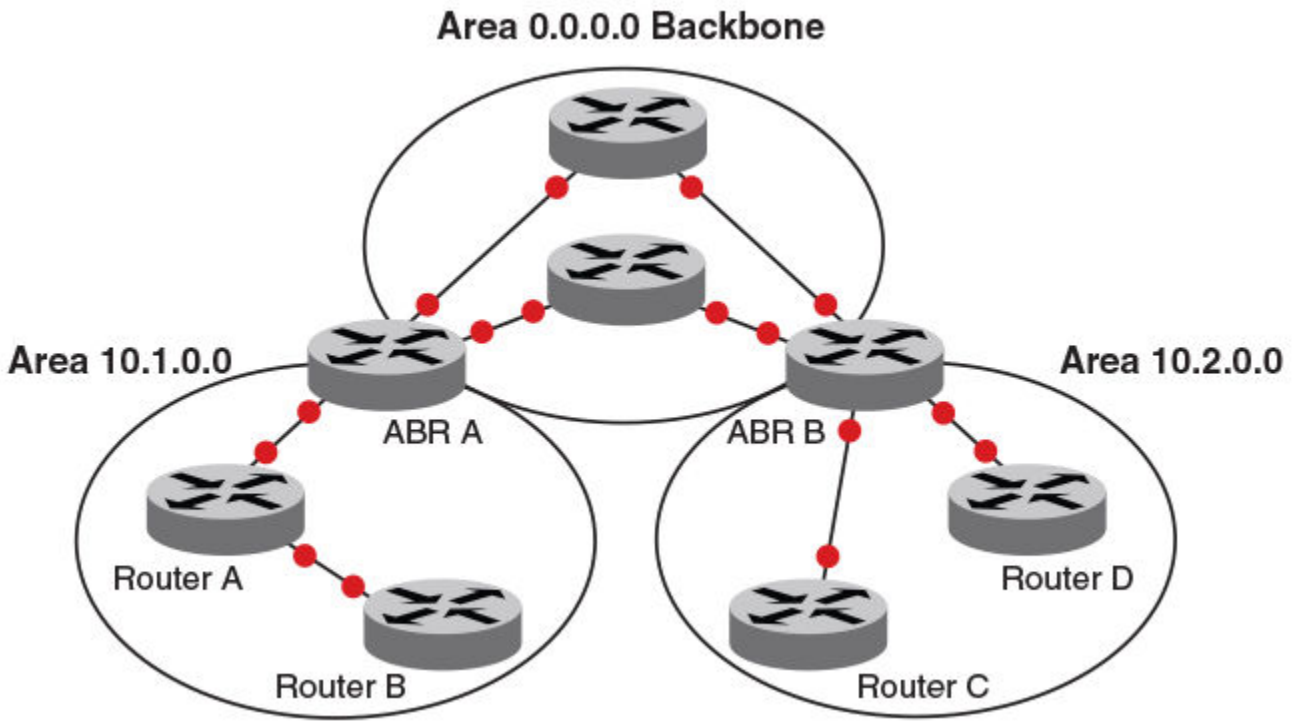
Open Shortest Path First Version 2 (OSPFv2) is a link-state routing protocol that uses link-state advertisements (LSAs) to update neighboring routers about a router's interfaces. Each router maintains an identical area-topology database to determine the shortest path to any neighboring router.

OSPF is built upon a hierarchy of network components and areas. The highest level of the hierarchy is the autonomous system. An autonomous system is defined as a number of networks, all of which share the same routing and administration characteristics. A backbone area forms the core of the network, connecting all other areas. Details of these and other OSPF components are provided below.

# Autonomous System

An Autonomous System can be divided into multiple areas. Each area represents a collection of contiguous networks and hosts. Areas limit the amount of advertisements sent within the network. This is known as flooding. An area is represented in OSPFv2 by either an IP address or a number.

**FIGURE 56** OSPF operating in a network




NOTE

For details of components and virtual links, refer to OSPFv2 components and roles on page 503 and Virtual links on page 512, respectively.

Once OSPFv2 is enabled on the system, the user assigns an IP address or number as the *area ID* for each area. The area ID is representative of all IP addresses (subnets) on a router port. Each port on a router can support one area.

# OSPFv2 components and roles

OSPFv2 can be configured on either a point-to-point or broadcast network.

Devices can take a variety of roles in an OSPFv2 topology, as discussed below.

## Area Border Routers

An OSPF router can be a member of multiple areas. Routers with membership in multiple areas are known as Area Border Routers (ABRs). All ABRs must have either a direct or indirect link to an OSPF backbone area (also known as area 0 or area 0.0.0.0). Each ABR maintains a separate topological database for each area the router is in. Each topological database contains all LSA databases for each router within a given area. The routers within the same area have identical topological databases. An ABR is responsible for forwarding routing information or changes among its border areas.

For more information on OSPFv2 areas, refer to the *OSPFv2 areas* section.

## Autonomous System Boundary Routers

An Autonomous System Boundary Router (ASBR) is a router that is running multiple protocols and serves as a gateway to routers outside the OSPF domain and those operating with different protocols. The ASBR is able to import and translate different protocol routes into OSPF through a process known as redistribution.

## Designated routers

In an OSPF broadcast network, OSPF elects one router to serve as the designated router (DR) and another router on the segment to act as the backup designated router (BDR). This minimizes the amount of repetitive information that is forwarded on the network. OSPF forwards all messages to the designated router.

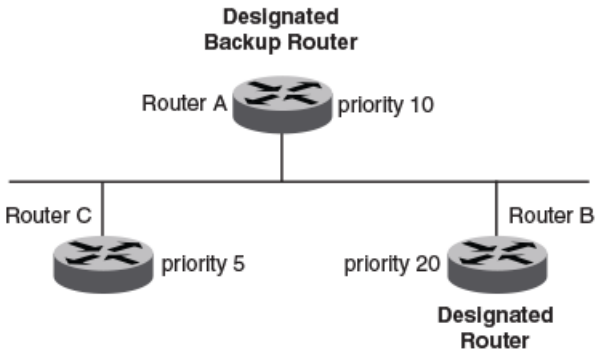
On broadcast networks such as LAN links, all routers on the LAN other than the DR and BDR form full adjacencies with the DR and BDR and pass LSAs only to them. The DR forwards updates received from one neighbor on the LAN to all other neighbors on that same LAN. One of the main functions of a DR is to ensure that all the routers on the same LAN have identical LSDBs. Therefore, on broadcast networks, an LSDB is synchronized between a DROther (a router that is not a DR or a BDR) and its DR and BDR.

> NOTE
> In an OSPF point-to-point network, where a direct Layer 3 connection exists between a single pair of OSPF routers, there is no need for designated or backup designated routers.

Without the need for Designated and Backup Designated routers, a point-to-point network establishes adjacency and converges faster. The neighboring routers become adjacent whenever they can communicate directly. In contrast, in broadcast and non-broadcast multi-access (NBMA) networks, the Designated Router and Backup Designated Router become adjacent to all other routers attached to the network.

In a network with no designated router and no backup designated router, the neighboring router with the highest priority is elected as the DR, and the router with the next highest priority is elected as the BDR, as shown in the figure below. Priority is a configurable option at the interface level; refer to the **ip ospf priority** command in the *Extreme NetIron Command Reference*.

FIGURE 57 Designated and backup router election




If the DR goes off line, the BDR automatically becomes the DR. The router with the next highest priority becomes the new BDR.

If two neighbors share the same priority, the router with the highest router ID is designated as the DR. The router with the next highest router ID is designated as the BDR. The DR and BDRs are recalculated after the OSPF protocol is disabled and re-enabled by means of the **[no] router ospf** command.

NOTE
By default, the device's router ID is the IP address configured on the lowest numbered loopback interface. If the device does not have a loopback interface, the default router ID is the lowest numbered IP address configured on the device.

When multiple routers on the same network are declaring themselves DRs, then both the priority and router ID are used to select the designated router and backup designated routers.

The DR and BDR election process is performed when one of the following events occurs:

- An interface is in a waiting state and the wait time expires.
- An interface is in a waiting state and receives a hello packet that addresses the BDR.
- A change in the neighbor state occurs, such as the following:
  - A neighbor state transitions from ATTEMPT state to a higher state.
  - Communication to a neighbor is lost.
  - A neighbor declares itself to be the DR or BDR for the first time.

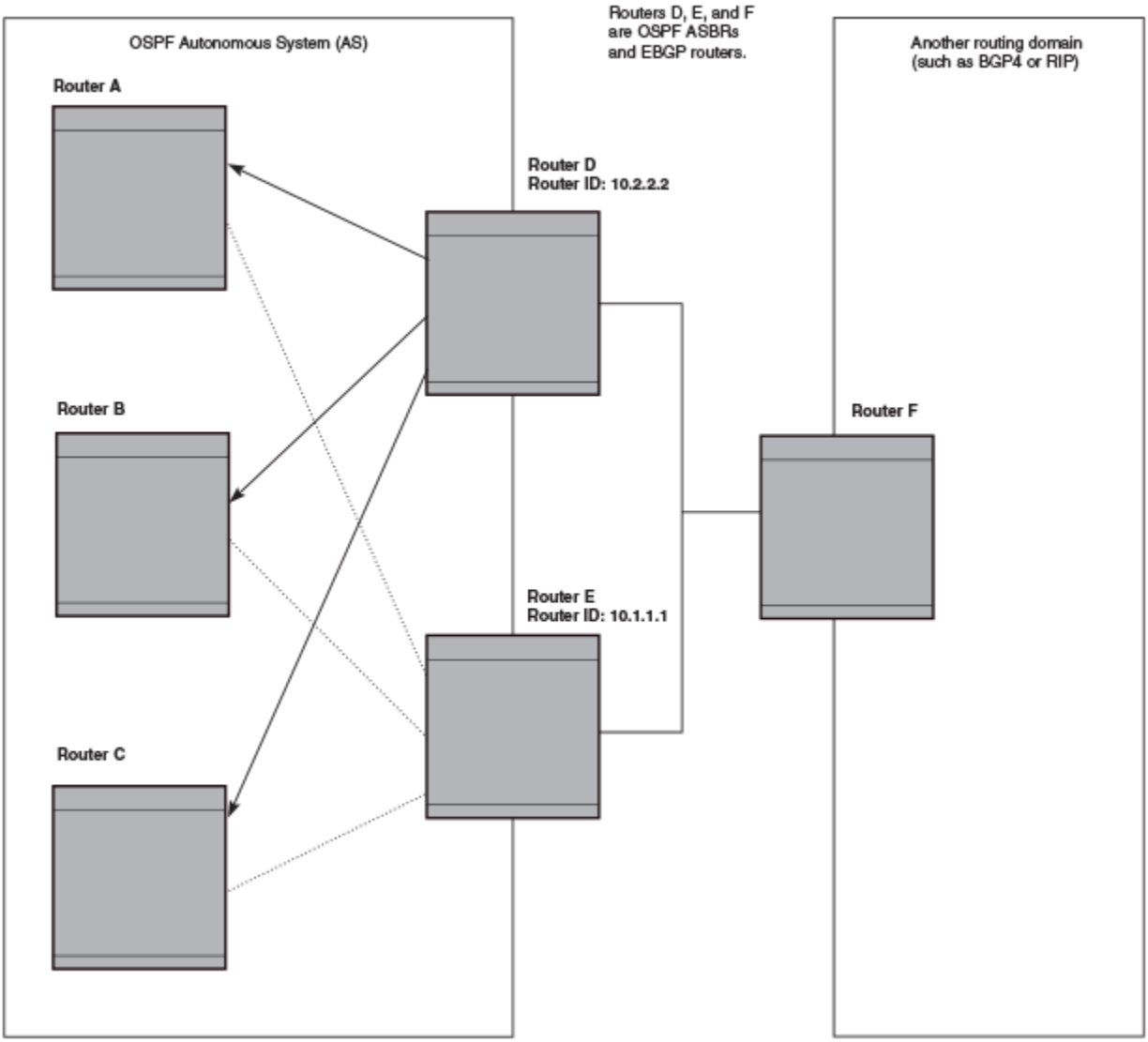
# Reduction of equivalent AS external LSAs

An OSPF ASBR uses AS External link advertisements (AS External LSAs) to originate advertisements of a route learned from another routing domain, such as a BGP4 or RIP domain. The ASBR advertises the route to the external domain by flooding AS External LSAs to all the other OSPF devices (except those inside stub networks) within the local OSPF Autonomous System (AS).

In some cases, multiple ASBRs in an AS can originate equivalent LSAs. The LSAs are equivalent when they have the same cost, the same next hop, and the same destination. The device optimizes OSPF by eliminating duplicate AS External LSAs in this case. The device with the lower router ID flushes the duplicate External LSAs from its database and thus does not flood the duplicate External LSAs into the OSPF AS. AS External LSA reduction, therefore, reduces the size of the link state database on the device. The AS External LSA reduction is described in RFC 2328

In this example, Routers D and E are OSPF ASBRs, and thus communicate route information between the OSPF AS, which contains Routers A, B, and C, and another routing domain, which contains Router F. The other routing domain is running another routing protocol, such as BGP4 or RIP. Routers D, E, and F, therefore, are each running both OSPF and either BGP4 or RIP.

**FIGURE 58** AS external LSA reduction




Notice that both Router D and Router E have a route to the other routing domain through Router F.

OSPF eliminates the duplicate AS External LSAs. When two or more devices are configured as ASBRs have equal-cost routes to the same next-hop router in an external routing domain, the ASBR with the highest router ID floods the AS External LSAs for the external domain into the OSPF AS, while the other ASBRs flush the equivalent AS External LSAs from their databases. As a result, the overall volume of route advertisement traffic within the AS is reduced and the devices that flush the duplicate AS External LSAs have more memory for other OSPF data. Because Router D has a higher router ID than Router E, Router D floods the AS External LSAs for Router F to Routers A, B, and C. Router E flushes the equivalent AS External LSAs from its database.

# Algorithm for AS external LSA reduction

The AS external LSA reduction example shows the normal AS External LSA reduction feature. The behavior changes under the following conditions:

- There is one ASBR advertising (originating) a route to the external destination, but one of the following happens:
  - A second ASBR comes on-line
  - A second ASBR that is already on-line begins advertising an equivalent route to the same destination.

In either case above, the router with the higher router ID floods the AS External LSAs and the other router flushes its equivalent AS External LSAs. For example, if Router D is offline, Router E is the only source for a route to the external routing domain. When Router D comes on-line, it takes over flooding of the AS External LSAs to Router F, while Router E flushes its equivalent AS External LSAs to Router F.

- One of the ASBRs starts advertising a route that is no longer equivalent to the route the other ASBR is advertising. In this case, the ASBRs each flood AS External LSAs. Since the LSAs either no longer have the same cost or no longer have the same next-hop router, the LSAs are no longer equivalent, and the LSA reduction feature no longer applies.
- The ASBR with the higher router ID becomes unavailable or is reconfigured so that it is no longer an ASBR. In this case, the other ASBR floods the AS External LSAs. For example, if Router D goes off-line, then Router E starts flooding the AS with AS External LSAs for the route to Router F.

# Enabling OSPFv2

A number of steps are required when enabling OSPFv2 on a device.

Consider the following when enabling OSPFv2 on a device.

- Redistribution must be enabled on devices configured to operate as ASBRs.
- All device ports must be assigned to one of the defined areas on an OSPF device. When a port is assigned to an area, all corresponding subnets on that port are automatically included in the assignment.

1. Enter the **router ospf** command in global configuration mode to enable OSPF on the device.
2. Assign the areas to which the device will be attached.
3. Assign individual interfaces to the OSPF areas.
4. Assign a virtual link to any ABR that does not have a direct link to the OSPF backbone area.
5. Refer to Changing default settings on page 535.

# Backbone area

The backbone area (also known as area 0 or area 0.0.0.0) forms the core of OSPF networks. All other areas should be connected to the backbone area either by a direct link or by virtual link configuration. Routers that have interfaces in both backbone area and (at least one) non-backbone area are called Area Border Routers (ABR). Inter area routing happens via ABRs.

The backbone area is the logical and physical structure for the OSPF domain and is attached to all non-zero areas in the OSPF domain.

The backbone area is responsible for distributing routing information between non-backbone areas. The backbone must be contiguous, but it does not need to be physically contiguous; backbone connectivity can be established and maintained through the configuration of virtual links.

# Assigning OSPFv2 areas

Areas can be assigned as OSPFv2 areas.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **router ospf** command to enter OSPF router configuration mode and enable OSPFv2 on the device.

```
device(config)# router ospf
```

3. Enter the **area** command to define an OSPFv2 area ID.

```
device(config-ospf-router)# area 0
```

4. Enter the **area** command to define a second OSPFv2 area ID.

```
device(config-ospf-router)# area 10.1.1.1
```

The following example assigns an OSPFv2 ID to two areas. One of the areas is assigned by decimal number. The second area is assigned by IP address.

```
device# configure terminal
device(config)# router ospf
device(config-ospf-router)# area 0
device(config-ospf-router)# area 10.1.1.1
```

# Area range

You can further consolidate routes at an area boundary by defining an area range. The area range allows you to assign an aggregate address to a range of IP and IPv6 addresses.

This aggregate value becomes the address that is advertised instead of all the individual addresses it represents being advertised. Only this aggregate or summary address is advertised into other areas instead of all the individual addresses that fall in the configured range. Area range configuration can considerably reduce the number of Type 3 summary LSAs advertised by a device. You have the option of adding the cost to the summarized route. If you do not specify a value, the cost value is the default range metric calculation for the generated summary LSA cost. You can temporarily pause route summarization from the area by suppressing the type 3 LSA so that the component networks remain hidden from other networks.

You can assign up to 32 ranges in an OSPF area.

## Assigning an area range

Ranges for an area can be assigned. Ranges allow a specific IP address and mask to represent a range of IP addresses within an area, so that only that reference range address is advertised to the network, instead of all the addresses within that range. Each area can have up to 32 range addresses.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **router ospf** command to enter OSPF router configuration mode and enable OSPFv2 on the device.

```
device(config)# router ospf
```

3. Enter the **area range** command, specifying an area ID, and enter the range. Repeat as necessary.

```
device(config-ospf-router)# area 10.0.0.10 range 10.45.0.0 10.255.0.0
device(config-ospf-router)# area 10.0.0.20 range 10.45.0.0 10.255.0.0
```

The following example defines an area range for subnets on 10.0.0.10 and 10.0.0.20.

```
device# configure terminal
device(config)# router ospf
device(config-ospf-router)# area 10.0.0.10 range 10.45.0.0 10.255.0.0
device(config-ospf-router)# area 10.0.0.20 range 10.45.0.0 10.255.0.0
```

# Area types

OSPFv2 areas can be normal, a stub area, a totally stubby area (TSA), or a not-so-stubby area (NSSA).

- Normal: OSPFv2 devices within a normal area can send and receive external link-state advertisements (LSAs).
- Stub: OSPFv2 devices within a stub area cannot send or receive external LSAs. In addition, OSPFv2 devices in a stub area must use a default route to the area's Area Border Router (ABR) to send traffic out of the area.
- NSSA: The Autonomous System Boundary Router (ASBR) of an NSSA can import external route information into the area.
  - ASBRs redistribute (import) external routes into the NSSA as type 7 LSAs. Type 7 External LSAs are a special type of LSA generated only by ASBRs within an NSSA, and are flooded to all the routers within only that NSSA.
  - ABRs translate type 7 LSAs into type 5 External LSAs, which can then be flooded throughout the autonomous system. The NSSA translator converts a type 7 LSA to a type 5 LSA if F-bit and P-bit are set and there is a reachable forwarding address. You can configure summary-addresses on the ABR of an NSSA so that the ABR converts multiple type 7 external LSAs received from the NSSA into a single type 5 external LSA.

  When an NSSA contains more than one ABR, OSPFv2 elects one of the ABRs to perform the LSA translation for NSSA. OSPFv2 elects the ABR with the highest router ID. If the elected ABR becomes unavailable, OSPFv2 automatically elects the ABR with the next highest router ID to take over translation of LSAs for the NSSA. The election process for NSSA ABRs is automatic.
- TSA: Similar to a stub area, a TSA does not allow summary routes in addition to not having external routes.

# Stub area and totally stubby area

A stub area is an area in which advertisements of external routes are not allowed, reducing the size of the database. A totally stubby area (TSA) is a stub area in which summary link-state advertisement (type 3 LSAs) are not sent. A default summary LSA, with a prefix of 0.0.0.0/0 is originated into the stub area by an ABR, so that devices in the area can forward all traffic for which a specific route is not known, via ABR.

A stub area disables advertisements of external routes. By default, the ABR sends summary LSAs (type 3 LSAs) into stub areas. You can further reduce the number of LSAs sent into a stub area by configuring the device to stop sending type 3 LSAs into the area. You can disable the summary LSAs to create a TSA when you are configuring the stub area or after you have configured the area.

The ABR of a totally stubby area disables origination of summary LSAs into this area, but still accepts summary LSAs from OSPF neighbors and floods them to other neighbors.

When you enter the **area stub** command with the **no-summary** keyword and specify an area to disable the summary LSAs, the change takes effect immediately. If you apply the option to a previously configured area, the device flushes all the summary LSAs it has generated (as an ABR) from the area with the exception of the default summary LSA originated. This default LSA is needed for the internal routers, since external routes are not propagated to them.

NOTE
Stub areas and TSAs apply only when the device is configured as an Area Border Router (ABR) for the area. To completely prevent summary LSAs from being sent to the area, disable the summary LSAs on each OSPF router that is an ABR for the area.

## Disabling summary LSAs for a stub area

LSAs can be disabled for a stub area.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **router ospf** command to enter OSPF router configuration mode and enable OSPFv2 on the device.

```
device(config)# router ospf
```

3. Enter the **area stub** command, specifying an area and a cost, followed by the **no-summary** parameter to set an additional cost on a specified stub area and prevent any Type 3 and Type 4 summary LSAs from being injected into the area.

```
device(config-ospf-router)# area 40 stub 99 no-summary
```

The following example configures a stub area, specifying a cost of 99 and preventing any Type 3 and Type 4 summary LSAs from being injected into the area.

```
device# configure terminal
device(config)# router ospf
device(config-ospf-router)# area 40 stub 99 no-summary
```

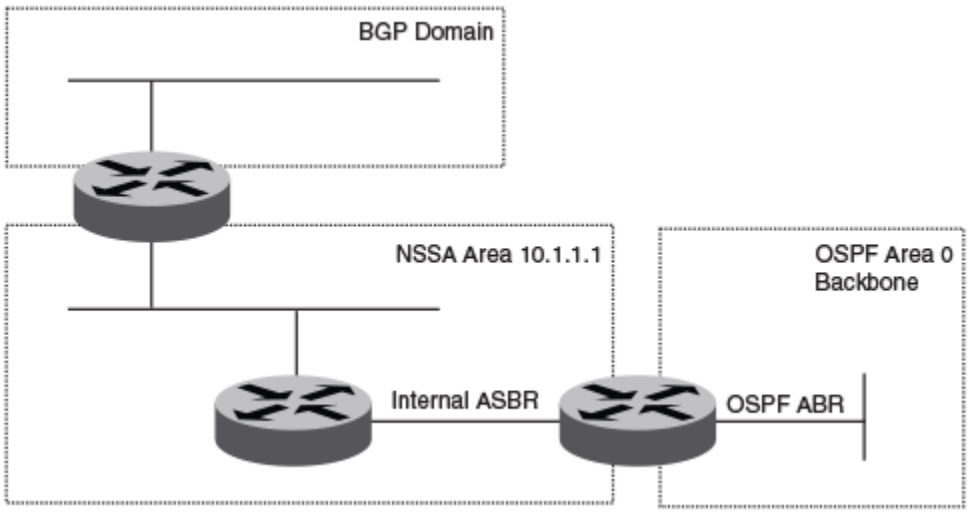
# Not-so-stubby area (NSSA)

The OSPFv2 not-so-stubby area (NSSA) enables you to configure OSPFv2 areas that provide the benefits of stub areas, but that also are capable of importing external route information. OSPFv2 does not flood external routes from other areas into an NSSA, but does translate and flood route information from the NSSA into other areas such as the backbone. Since external routes are not published, a Type 7 default LSA with a prefix of ::/0 and a cost of 10 is originated into the NSSA area by the ABR to ensure that traffic passes through.

NSSAs are especially useful when you want to summarize type 5 External LSAs (external routes) before forwarding them into an OSPFv2 area. The OSPFv2 specification prohibits summarization of type 5 LSAs and requires OSPFv2 to flood type 5 LSAs throughout a routing domain. When you configure an NSSA, you can specify a summary-address for aggregating the external routes that the NSSA's ABR exports into other areas.

The figure below shows an example of an OSPFv2 network containing an NSSA.

**FIGURE 59** OSPF network containing an NSSA




This example shows two routing domains, a BGP domain and an OSPF domain. The ASBR inside the NSSA imports external routes from BGP into the NSSA as type 7 LSAs, which the ASBR floods throughout the NSSA.

The ABR translates the type 7 LSAs into type 5 LSAs. If a summary-address is configured for the NSSA, the ABR also summarizes the LSAs into an aggregate LSA before flooding the type 5 LSAs into the backbone.

Because the NSSA is partially stubby the ABR does not flood external LSAs from the backbone into the NSSA. To provide access to the rest of the Autonomous System (AS), the ABR generates a default type 7 LSA into the NSSA.

ABRs of an NSSA area can be configured with the no-summary parameter to prevent the generation of type 3 and type 4 summary LSAs into the area. The only exception is the default type 3 LSA, with a prefix of 0.0.0.0/0. The default type 7 LSA is not originated in this case.

# Configuring an NSSA

OSPFv2 areas can be defined as NSSA areas with modifiable parameters.

1. Enter the **configure terminal** command to access global configuration mode.

   ```
   device# configure terminal
   ```

2. Enter the **router ospf** command to enter OSPF router configuration mode and enable OSPFv2 on the device.

   ```
   device(config)# router ospf
   ```

3. Enter the **area nssa** command and specify an area address and a cost.

   ```
   device(config-ospf-router)# area 10.1.1.1 nssa 1
   ```

   Area 10.1.1.1 is defined as an NSSA.

The following example configures OSPF area 10.1.1.1 as an NSSA.

```
device# configure terminal
device(router ospf
device(config-ospf-router)# area 10.1.1.1 nssa 1
```

## Configuring a summary-address for the NSSA

If you want the ABR that connects the NSSA to other areas to summarize the routes in the NSSA before translating them into type 5 LSAs and flooding them into the other areas, configure an address range summary-address. The ABR creates an aggregate value based on the address range. The aggregate value becomes the address that the ABR advertises instead of advertising the individual addresses represented by the aggregate. You can configure up to 32 ranges in an OSPFv2 area.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **router ospf** command to enter OSPF router configuration mode and enable OSPFv2 on the device.

```
device(config)# router ospf
```

3. Enter the **area nssa** command, specifying an area and a cost.

```
device(config-ospf-router)# area 10.1.1.1 nssa 10
```

4. Enter the **summary-address** command, followed by the IP address and mask for the summary route.

```
device(config-ospf-router)# summary-address 10.10.1.0    10.10.2.0
```

The following example configures a summary-address in NSSA 10.1.1.1.

```
device# configure terminal
device(config)# router ospf
device(config-ospf-router)# area 10.1.1.1 nssa 10
device(config-ospf-router)# summary-address 10.10.1.0    10.10.2.0
```

# Assigning interfaces to an area

Once you define OSPFv2 areas, you can assign interfaces to the areas. All device ports must be assigned to one of the defined areas on an OSPFv2 device. When a port is assigned to an area, all corresponding subnets on that port are automatically included in the assignment.

To assign a loopback interface to an area with the IP address of 10.5.0.0, perform the following task:

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **interface** command and specify an interface.

```
device(config)# interface loopback 2
```

3. Enter the **ip ospf area** command followed by the IP address of the area.

```
device(config-lbif-2)# ip ospf area 10.5.0.0
```

   If you want to set an interface to passive mode, use the **ip ospf passive** command. If you want to block flooding of outbound LSAs on specific OSPF interfaces, use the **ip ospf database-filter all out** command.(Refer to the *Extreme NetIron Command Reference* for details.)

The following example assigns a loopback interface to an area with the IP address of 10.5.0.0.

```
device# configure terminal
device(config)# interface loopback 2
device(config-lbif-2)# ip ospf area 10.5.0.0
```

# Link state advertisements

Extreme devices support the following types of LSAs, which are described in RFC 2328 and 3101:

- Router link
- Network link
- Summary link
- Autonomous system summary link
- AS external link
- Not-So-Stubby Area (NSSA) external link
- Grace LSAs

Communication among areas is provided by means of link state advertisements (LSAs). The LSAs supported for each area type are as follows:

- Backbone (area 0) supports LSAs 1, 2, 3, 4, 5, and 7.
- Nonbackbone area supports LSAs 1, 2, 3, 4, and 5.
- Stub area supports LSAs 1, 2, and 3.
- Totally stubby area (TSA) supports LSAs 1 and 2, and also supports a single LSA 3 per ABR, advertising a default route.
- No so stubby area (NSSA) supports LSAs 1, 2, 3, and 7.

# Virtual links

All ABRs must have either a direct or indirect link to the OSPFv2 backbone area (0.0.0.0 or 0). If an ABR does not have a physical link to the area backbone, the ABR can configure a virtual link to another router within the same area, which has a physical connection to the area backbone.

The path for a virtual link is through an area shared by the neighbor ABR (router with a physical backbone connection), and the ABR requires a logical connection to the backbone.

Two parameters fields must be defined for all virtual links—transit area ID and neighbor router:

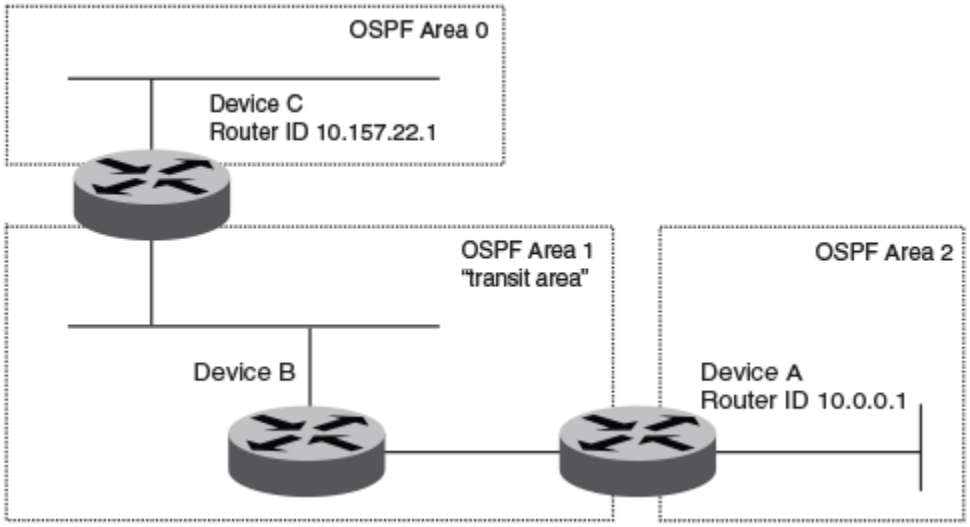
- The transit area ID represents the shared area of the two ABRs and serves as the connection point between the two routers. This number should match the area ID value.
- The neighbor router field is the router ID (IP address) of the router that is physically connected to the backbone, when assigned from the router interface requiring a logical connection. When assigning the parameters from the router with the physical connection, be aware that the router ID is the IP address of the router requiring a logical connection to the backbone.

  NOTE
  By default, a device's router ID is the IP address configured on the lowest numbered loopback interface. If the device does not have a loopback interface, the default router ID is the lowest numbered IP address configured on the device. When you establish an area virtual link, you must configure it on both of the routers (both ends of the virtual link).

Virtual links cannot be configured in stub areas and NSSAs.

The following figure shows an OSPF area border router, Device A, that is cut off from the backbone area (area 0). To provide backbone access to Device A, you can add a virtual link between Device A and Device C using Area 1 as a transit area. To configure the virtual link, you define the link on the router that is at each end of the link. No configuration for the virtual link is required on the routers in the transit area.

**FIGURE 60** Defining OSPF virtual links within a network




# Configuring virtual links

If an Area Border Router (ABR) does not have a physical link to a backbone area, a virtual link can be configured between that ABR and another device within the same area that has a physical link to a backbone area.

A virtual link is configured, and a virtual link endpoint on two devices, ABR1 and ABR2, is defined.

1. On ABR1, enter the **configure terminal** command to access global configuration mode.

   ```
   device# configure terminal
   ```

2. Enter the **router ospf** command to enter OSPF router configuration mode and enable OSPFv2 on the device.

   ```
   device(config)# router ospf
   ```

3. Enter the **area** command to assign an OSPFv2 area ID.

   ```
   device(config-ospf-router)# area 0
   ```

4. Enter the **area** command to assign an OSPFv2 area ID.

   ```
   device(config-ospf-router)# area 1
   ```

5. Enter the **area virtual-link** command and the ID of the OSPFv2 device at the remote end of the virtual link to configure the virtual link endpoint.

   ```
   device(config-ospf-router)# area 1 virtual-link 10.2.2.2
   ```

6. On ABR2, enter the **configure terminal** command to access global configuration mode.

   ```
   device# configure terminal
   ```

7. Enter the **router ospf** command to enter OSPFv2 router configuration mode and enable OSPFv2 on the device.

   ```
   device(config)# router ospf
   ```

8. Enter the **area** command to assign an OSPFv2 area ID.

```
device(config-ospf-router)# area 1
```

9. Enter the **area** command to assign an OSPFv2 area ID.

```
device(config-ospf-router)# area 2
```

10. Enter the **area virtual-link** command and the ID of the OSPFv2 device at the remote end of the virtual link to configure the virtual link endpoint.

```
device(config-ospf-router)# area 1 virtual-link 10.1.1.1
```

The following example configures a virtual link between two devices.

```
ABR1:
device1# configure terminal
device1(config)# router ospf
device1(config-ospf-router)# area 0
device1(config-ospf-router)# area 1
device1(config-ospf-router)# area 1 virtual-link 10.2.2.2

ABR2:
device2# configure terminal
device2(config)# router ospf
device2(config-ospf-router)# area 1
device2(config-ospf-router)# area 2
device2(config-ospf-router)# area 1 virtual-link 10.1.1.1
```

# Default route origination

When the device is an OSPFv2 Autonomous System Boundary Router (ASBR), you can configure it to automatically generate a default external route into an OSPFv2 routing domain.

By default, a device does not advertise the default route into the OSPFv2 domain. If you want the device to advertise the OSPFv2 default route, you must explicitly enable default route origination. When you enable OSPFv2 default route origination, the device advertises a type 5 default route that is flooded throughout the autonomous system, with the exception of stub areas.

The device advertises the default route into OSPFv2 even if OSPFv2 route redistribution is not enabled, and even if the default route is learned through an iIBGP neighbor when default-information-originate is configured. The device does not, however, originate the default route if the active default route is learned from an OSPFv2 device in the same domain.

> NOTE
> The device does not advertise the OSPFv2 default route, regardless of other configuration parameters, unless you explicitly enable default route origination.

If default route origination is enabled and you disable it, the default route originated by the device is flushed. Default routes generated by other OSPFv2 devices are not affected. If you re-enable the default route origination, the change takes effect immediately and you do not need to reload the software.

# External route summarization

An ASBR can be configured to advertise one external route as an aggregate for all redistributed routes that are covered by a specified address range.

When you configure a summary address range, the range takes effect immediately. All the imported routes are summarized according to the configured summary address range. Imported routes that have already been advertised and that fall within the range are flushed out of the autonomous system and a single route corresponding to the range is advertised.

If a route that falls within a configured summary address range is imported by the device, no action is taken if the device has already advertised the aggregate route; otherwise, the device advertises the aggregate route. If an imported route that falls within a configured summary address range is removed by the device, no action is taken if there are other imported routes that fall within the same summary address range; otherwise, the aggregate route is flushed.

You can configure up to 32 summary address ranges. The device sets the forwarding address of the aggregate route to 0 and sets the tag to 0. If you delete a summary address range, the advertised aggregate route is flushed and all imported routes that fall within the range are advertised individually. If an external link-state database (LSDB) overflow condition occurs, all aggregate routes and other external routes are flushed out of the autonomous system. When the device exits the external LSDB overflow condition, all the imported routes are summarized according to the configured summary address ranges.

NOTE
If you use redistribution filters in addition to summary address ranges, the device applies the redistribution filters to routes first, and then applies them to the summary address ranges.

NOTE
If you disable redistribution, all the aggregate routes are flushed, along with other imported routes.

NOTE
This option affects only imported, type 5 external LSA routes. A single type 5 LSA is generated and flooded throughout the autonomous system for multiple external routes. Type 7-route redistribution is not affected by this feature. All type 7 routes will be imported (if redistribution is enabled). To summarize type 7 LSAs or exported routes, use NSSA address range summarization.

# SPF timers

The device uses an SPF delay timer and an SPF hold-time timer to calculate the shortest path for OSPFv2 routes. The values for both timers can be changed.

- SPF delay: When the device receives a topology change, it waits before starting a Shortest Path First (SPF) calculation. By default, the device waits zero seconds. You can configure the SPF delay to a value from 0 through 65535 seconds. If you set the SPF delay to 0 seconds, the device immediately begins the SPF calculation after receiving a topology change.
- SPF hold time: The device waits a specific amount of time between consecutive SPF calculations. By default, it waits zero seconds. You can configure the SPF hold time to a value from 0 through 65535 seconds. If you set the SPF hold time to 0 seconds, the device does not wait between consecutive SPF calculations.

You can set the SPF delay and hold time to lower values to cause the device to change to alternate paths more quickly if a route fails. Note that lower values for these parameters require more CPU processing time.

You can change one or both of the timers.

NOTE
If you want to change only one of the timers, for example, the SPF delay timer, you must specify the new value for this timer as well as the current value of the SPF hold timer, which you want to retain. The device does not accept only one timer value.

NOTE
If you configure SPF timers between 0 through 100, they default to 0.

# Modifying Shortest Path First timers

The Shortest Path First (SPF) throttle timers can be modified.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **router ospf** command to enter OSPF router configuration mode and enable OSPFv2 globally.

```
device(config)# router ospf
```

3. Enter the **timers** command with the **throttle spf** keyword and specify the SPF delay, the hold time, and the maximum wait time.

```
device(config-ospf-router)# timers throttle spf 100 500 5000
```

The following example sets the SPF initial delay to 100 milliseconds, the hold time to 500 milliseconds, and the maximum wait time to 5000 milliseconds.

```
device# configure terminal
device(config)# router ospf
device(config-ospf-router)# timers throttle spf 100 500 5000
```

# OSPFv2 administrative distance

Devices can learn about networks from various protocols and select a route based on the source of the route information. This decision can be influenced if the default administrative distance for OSPFv2 routes is changed. Consequently, the routes to a network may differ depending on the protocol from which the routes were learned.

You can influence the device's decision by changing the default administrative distance for OSPFv2 routes. You can configure a unique administrative distance for each type of OSPFv2 route. For example, you can configure the Extreme device to prefer a static route over an OSPFv2 inter-area route and to prefer OSPFv2 intra-area routes over static routes. The distance you specify influences the choice of routes when the device has multiple routes to the same network from different protocols. The device prefers the route with the lower administrative distance.

You can specify unique default administrative distances for the following OSPFv2 route types:

- External routes
- Intra-area routes
- Inter-area routes
- Route maps

NOTE
The choice of routes within OSPFv2 is not influenced. For example, an OSPFv2 intra-area route is always preferred over an OSPFv2 inter-area route, even if the intra-area route's distance is greater than the inter-area route's distance.

# OSPFv2 LSA refreshes

To prevent a refresh from being performed each time an individual LSA's refresh timer expires, OSPFv2 LSA refreshes are delayed for a specified time interval. This pacing interval can be altered.

The device paces OSPFv2 LSA refreshes by delaying the refreshes for a specified time interval instead of performing a refresh each time an individual LSA's refresh timer expires. The accumulated LSAs constitute a group, which the device refreshes and sends out together in one or more packets.

The pacing interval, which is the interval at which the device refreshes an accumulated group of LSAs, is configurable in a range from 10 through 1800 seconds (30 minutes). The default is 240 seconds (4 minutes). Thus, every four minutes, the device refreshes the group of accumulated LSAs and sends the group together in the same packets.

The pacing interval is inversely proportional to the number of LSAs the device is refreshing and aging. For example, if you have approximately 10,000 LSAs, decreasing the pacing interval enhances performance. If you have a very small database (40 to 100 LSAs), increasing the pacing interval to 10 to 20 minutes may enhance performance only slightly.

## Configuring the OSPFv2 LSA pacing interval

The interval between OSPFv2 LSA refreshes can be modified.

1. Enter the **configure terminal** command to access global configuration mode.

   ```
   device# configure terminal
   ```

2. Enter the **router ospf** command to enter OSPF router configuration mode and enable OSPFv2 globally.

   ```
   device(config)# router ospf
   ```

3. Enter the **timers** command with the **lsa-group-pacing** parameter.

   ```
   device(config-ospf-router)# timers lsa-group-pacing 120
   ```

   The OSPFv2 LSA pacing interval is changed to 120 seconds (2 minutes).

The following example changes the OSPFv2 LSA pacing interval is changed to 120 seconds (2 minutes).

```
device# configure terminal
device(config)# router ospf
device(config-ospf-router)# timers lsa-group-pacing 120
```

# Support for OSPF RFC 2328 Appendix E

Extreme devices support Appendix E in OSPF RFC 2328. Appendix E describes a method to ensure that an OSPF device generates unique link state IDs for type-5 (External) link state advertisements (LSAs) in cases where two networks have the same network address but different network masks.

> NOTE
> Support for Appendix E of RFC 2328 is enabled automatically and cannot be disabled. No user configuration is required.

Normally, an OSPF device uses the network address alone for the link state ID of the link state advertisement (LSA) for the network. For example, if the device needs to generate an LSA for network 10.1.2.3 255.0.0.0, the device generates ID 10.1.2.3 for the LSA.

However, suppose that an OSPF device needs to generate LSAs for all the following networks:

- 10.0.0.0 255.0.0.0
- 10.0.0.0 255.255.0.0
- 10.0.0.0 255.255.255.0

All three networks have the same network address, 10.0.0.0. Without support for RFC 2328 Appendix E, an OSPF device uses the same link state ID, 10.0.0.0, for the LSAs for all three networks. For example, if the device generates an LSA with ID 10.0.0.0 for network 10.0.0.0 255.0.0.0, this LSA conflicts with the LSA generated for network 10.0.0.0 255.255.0.0 or 10.0.0.0 255.255.255.0. The result is multiple LSAs that have the same ID but that contain different route information.

When appendix E is supported, the device generates the link state ID for a network as the following steps.

1. Does an LSA with the network address as its ID already exist?
   - – No - Use the network address as the ID.
   - – Yes - Go to "Support for OSPF RFC 2328 Appendix E".
2. Compare the networks that have the same network address, to determine which network is more specific. The more specific network is the one that has more contiguous one bits in its network mask. For example, network 10.0.0.0 255.255.0.0 is more specific than network 10.0.0.0 255.0.0.0, because the first network has 16 ones bits (255.255.0.0) whereas the second network has only 8 ones bits (255.0.0.0).
   - – For the less specific network, use the networks address as the ID.
   - – For the more specific network, use the network's broadcast address as the ID. The broadcast address is the network address, with all ones bits in the host portion of the address. For example, the broadcast address for network 10.0.0.0 255.255.0.0 is 10.0.255.255.

   If this comparison results in a change to the ID of an LSA that has already been generated, the device generates a new LSA to replace the previous one. For example, if the device has already generated an LSA for network with ID 10.0.0.0 for network 10.0.0.0 255.255.255.0, the device must generate a new LSA for the network, if the device needs to generate an LSA for network 10.0.0.0 255.255.0.0 or 10.0.0.0 255.0.0.0.

# OSPFv2 graceful restart

The graceful restart (GR) feature provides a routing device with the capability to inform its neighbors when it is performing a restart.

Neighboring devices, known as GR helpers, are informed via protocol extensions that the device is undergoing a restart and assist in the restart. For the duration of the graceful restart, the restarting device and its neighbors continue forwarding packets ensuring there is no disruption to network performance or topology. Disruptions in forwarding are minimized and route flapping diminished. When the restart is complete, the device is able to quickly resume full operation due to the assistance of the GR helpers. The adjacent devices then return to normal operation.

There are two types of OSPFv2 graceful restart:

- **Planned restart**: The restarting routing device informs its neighbors before performing the restart. The GR helpers act as if the routing device is still within the network topology, continuing to forward traffic to the restarting routing device. A defined interval, known as a "grace period" is set to specify when the neighbors should consider the restart complete and the restarting routing device as part of the network topology again.
- **Unplanned restart**: The routing device restarts without warning due to a software fault.

  NOTE
  In order for a graceful restart on a routing device to be successful, the OSPFv2 neighbors must have GR-helper mode enabled. GR-helper mode is enabled by default.

OSPFv2 Graceful Restart can be enabled in the following configurations:

- Configuring OSPFv2 Graceful Restart for the Global Instance – In this configuration all OSPFv2 neighbors other than those used by VRFs are made subject to the Graceful Restart capability. The restart timer set globally does not apply to Graceful Restart on a configured VRF.
- Configuring OSPFv2 Graceful Restart per VRF – In this configuration all OSPFv2 neighbors for the specified VRF are made subject to the Graceful Restart capability. The restart timer set for a specific VRF only applies to that VRF.

  NOTE
  If a 32-slot XMR Series or MLX Series system running a version 03.6.00 or later application image is configured for OSPFv2 graceful restart and intended to be used in switchover or hitless upgrade, the OSPFv2 dead-interval needs to be changed to 60 seconds on OSPFv2 interfaces to ensure that the graceful restart process succeeds without a timeout.

# Hitless upgrade support for OSPF graceful restart

OSPFv2 graceful restart experiences minimal packet loss during hitless upgrade on a non-default VRF. On a default VRF, there is no packet loss during hitless upgrade.

# Disabling OSPFv2 graceful restart

OSPFv2 graceful restart (GR) is enabled by default, and can be disabled on a routing device.

1. Enter the **configure terminal** command to access global configuration mode.

   ```
   device# configure terminal
   ```

2. Enter the **router ospf** command to enter OSPF router configuration mode and enable OSPFv2 globally.

   ```
   device(config)# router ospf
   ```

3. Enter the **no graceful restart** command to disable GR on the device.

   ```
   device(config-ospf-router)# no graceful-restart
   ```

The following example disables GR.

```
device# configure terminal
device(config)# router ospf
device(config-ospf-router)# no graceful-restart
```

# Re-enabling OSPFv2 graceful restart

If you disable OSPFv2 graceful restart (GR), you can re-enable it. You can also change the maximum restart wait time from the default value of 120 seconds.

NOTE
GR is mutually exclusive to NSR.

1. Enter the **configure terminal** command to access global configuration mode.

   ```
   device# configure terminal
   ```

2. Enter the **router ospf** command to enter OSPF router configuration mode and enable OSPFv2 globally.

```
device(config)# router ospf
```

3. Enter the **graceful restart** command to re-enable GR on the device.

```
device(config-ospf-router)# graceful-restart
```

4. Enter the **graceful restart** command with the **restart-time** parameter and specify a value to change the maximum restart wait time from the default value of 120 seconds.

```
device(config-ospf-router)# graceful-restart restart-time 240
```

The following example re-enables GR and changes the maximum restart wait time from the default value of 120 seconds to 240 seconds.

```
device# configure terminal
device(config)# router ospf
device(config-ospf-router)# graceful-restart
device(config-ospf-router)# graceful-restart restart-time 240
```

## Disabling OSPFv2 graceful restart helper

The OSPFv2 graceful restart (GR) helper is enabled by default, and can be disabled on a routing device.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **router ospf** command to enter OSPF router configuration mode and enable OSPFv2 on the device.

```
device(config)# router ospf
```

3. Enter the **graceful-restart** command using the **helper-disable** keyword to disable the GR helper.

```
device(config-ospf-router)# graceful-restart helper-disable
```

The following example disables the GR helper.

```
device# configure terminal
device(config)# router ospf
device(config-ospf-router)# graceful-restart helper-disable
```

# OSPFv2 stub router advertisement

OSPFv2 stub router advertisement is an open standard based feature and it is specified in RFC 3137. This feature provides a user with the ability to gracefully introduce and remove an OSPFv2 device from the network, by controlling when the data traffic can start and stop flowing through the device in cases where there are other OSPFv2 devices present on the network providing alternative paths for the traffic. This feature does not work if there is no alternative for the traffic through other OSPFv2 routers. The device can control the data traffic flowing through it by changing the cost of the paths passing through the configured device. By setting the path cost high the traffic will be redirected to other OSPFv2 devices providing a lower cost path. This change in path cost is accomplished by setting the metric of the links advertised in the Router LSA to a maximum value. When the OSPFv2 device is ready to forward the traffic, the links are advertised with the real metric value instead of the maximum value.

OSPFv2 stub router advertisement is useful for avoiding a loss of traffic during short periods when adjacency failures are detected and traffic is rerouted. Using this feature, traffic can be rerouted before an adjacency failure occurs due to common services interruptions such as a router being shutdown for maintenance.

OSPFv2 stub router advertisement is also useful during startup because it gives the device enough time to build up its routing table before forwarding traffic. This can be useful where BGP is enabled on the device because it takes time for the BGP routing table to converge.

You can also configure and set a metric value for the following LSA types:

- Summary (type 3 and type 4)
- External (type 5 and type 7)
- Opaque (type 10, TE link)

# OSPFv2 Shortest Path First throttling

Rapid triggering of SPF calculations with exponential back-off to offer the advantages of rapid convergence without sacrificing stability. As the delay increases, multiple topology changes can occur within a single SPF. This dampens network activity due to frequent topology changes.

This scheduling method starts with an initial value after which a configured delay time is followed. If a topology change event occurs the SPF is schedule after the time specified by the initial value, the device starts a timer for the time period specified by a configured hold time value. If no topology events occur during this hold time, the router returns to using the initial delay time.

If a topology event occurs during the hold time period, the next hold time period is recalculated to a value that is double the initial value. If no topology events occur during this extended hold time, the device resets to its initial value. If an event occurs during this extended hold time, the next hold time is doubled again. The doubling occurs as long as topology events occur during the calculated hold times until a configured maximum delay time value is reached or no event occurs (which resets the router to the initial hold time). The maximum value is then held until the hold time expires without a topology change event occurring. At any time that a hold time expires without a topology change event occurring, the router reverts to the initial hold value and begins the process all over again.

For example, if you set the initial delay timer to 100 milliseconds, the hold timer to 300 and the maximum hold timer to 2000 milliseconds, the following will occur:

If a topology change occurs the initial delay of 100 milliseconds will be observed. If a topology change occurs during the hold time of 300 milliseconds the hold time is doubled to 600 milliseconds. If a topology change event occurs during the 600 millisecond period, the hold time is doubled again to 1200 milliseconds. If a topology change event occurs during the 1200 millisecond period, the hold time is doubled to 2400 milliseconds. Because the maximum hold time is specified as 2000, the value will be held at 2000. This 2000 millisecond period will then repeat as long as topology events occur within the maximum 2000 millisecond hold time. When a maximum hold time expires without a topology event occurring, the router reverts to the initial delay time and the cycle repeats as described.

Therefore, longer SPF scheduling values can be used during network topology instability.

# IETF RFC and internet draft support

The implementation of OSPF Graceful Restart supports the following IETF RFC:

- RFC 3623: Graceful OSPF Restart

  NOTE
  A secondary management module must be installed for the device to function as a graceful restart device. If the device functions as a graceful restart helper device only, there is no requirement for a secondary management module.

# OSPFv2 non-stop routing

OSPFv2 can continue operation without interruption during hitless failover when the OSPFv2 non-stop routing (NSR) feature is enabled.

During graceful restart (GR), the restarting neighbors must help build routing information during a failover. However, GR may not be supported by all devices in a network. NSR eliminates this dependency.

NSR does not require support from neighboring devices to perform hitless failover, and OSPF can continue operation without interruption.

> NOTE
> NSR does not support IPv6-over-IPv4 tunneling and virtual links, so traffic loss is expected while performing hitless failover.

If the active management module fails, the standby management module takes over and maintains the current OSPF routes, link-state advertisements (LSAs), and neighbor adjacencies, so that there is no loss of existing traffic to the OSPF destination.

> NOTE
> NSR and Graceful Restart (GR) are mutually exclusive.

## Limitations of NSR

- Configurations that occur before the switchover are lost due to the CLI synchronization.
- Sham links are not supported.
- OSPF adjacency over GRE tunnels is not supported.
- Changes in the neighbor state or interface state before or during a switchover do not take effect.
- Traffic counters are not synchronized because the neighbor and LSA database counters are recalculated on the standby module during synchronization.
- LSA acknowledging is delayed because it has to wait until standby acknowledging occurs.
- Depending on the sequence of redistribution or new LSAs (from neighbors), the LSAs accepted within the limits of the database may change after switchover.
- In NSR hitless failover, after switchover, additional flooding-related protocol traffic is generated to the directly connected neighbors.
- OSPF startup timers, database overflow, and max-metric, are not applied during NSR switchover.
- Devices may generate OSPF log messages or reset OSPF neighbor timers, but these issues do not cause any OSPF or traffic disruption.

## BFD with OSPF NSR

Bidirectional forwarding detection (BFD) supports MP switchover and all BFD sessions for OSPF with graceful OSPF NSR, which are in the up state after the switchover. The BFD sessions for OSPF that do not use OSPF NSR are cleared before the switchover and then re-established on the new active MP after the MP switchover.

In case the active MP learns an OSPF neighbor and then restarts before a new BFD session is established, the standby module will not have a BFD session for the new OSPF neighbor. To overcome this and to support OSPF NSR with BFD, the following functions are supported when the active MP restarts:

- During MP switchover, BFD checks whether OSPF NSR is enabled. If OSPF NSR is enabled, the existing BFD sessions for OSPF is maintained during the switchover.

- OSPF sets up or clears the BFD sessions after OSPF neighbor transition.
- After the switchover, BFD sessions correspond with the active OSPF neighbor.

## Enabling OSPFv2 NSR

OSPFv2 non-stop routing (NSR) can be re-enabled if it has been disabled. The following task re-enables NSR for OSPFv2.

NOTE
GR is mutually exclusive to NSR.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **router ospf** command to enter OSPF router configuration mode and enable OSPFv2 globally.

```
device(config)# router ospf
```

3. Enter the **nonstop-routing** command to re-enable NSR on the device.

```
device(config-ospf-router)# nonstop-routing
```

The following example re-enables NSR for OSPFv2.

```
device# configure terminal
device(config)# router ospf
device(config-ospf-router)# nonstop-routing
```

# Synchronization of critical OSPFv2 elements

All types of LSAs and the neighbor information are synchronized to the standby module using the NSR synchronization library and IPC mechanism to transmit and receive packets.

## Link state database synchronization

To ensure non-stop routing, when the active management module fails the standby management module takes over from the active management module, with the identical OSPF link state database it had before the failure. The next shortest path first (SPF) run after the switchover yields the same result in routes as the active module had before the failure. The OSPF protocol requires that all devices in the network have identical databases.

## LSA delayed acknowledging

When an OSPF device receives LSAs from its neighbor, it acknowledges the LSAs. After the acknowledgement is received, the neighbor removes this device from its retransmission list and stops resending the LSAs.

In the case of NSR, the device fails after receiving the LSA from its neighbor and acknowledges that neighbor upon receipt of an LSA. The LSA synchronization to the standby module is then completed. In this case the standby module, when taking over from the active module, does not have that LSA in its database and the already acknowledged neighbor does not retransmit that LSA. For this reason, the NSR-capable device waits for LSA synchronization of the standby module to complete (Sync-Ack) before acknowledging the neighbor that sent the LSA.

## LSA syncing and packing

When the LSA processing is completed on the active management module and the decision is made to install the LSA in its link state database (LSDB), OSPF synchronizes that LSA to the standby module. OSPF checks the current state of the database entry, whether or not it is marked for deletion. After checking the database state, OSPF packs the LSA status and other necessary information needed for direct installation in the standby OSPF LSDB, along with the LSA portion. When the LSA reaches the standby module, OSPF checks the database entry state in the buffer and takes appropriate action, such as adding, overwriting, updating, or deleting the LSA from the LSDB.

## Neighbor device synchronization

When the neighbor device is added in the active management module, it is synchronized and added to the standby module. When the neighbor is deleted in the active module, it is synchronized to the standby module and deleted in the standby module. When the neighbor device state becomes 2way or full, the neighbor device is synchronized to the standby module. The following attributes of the neighbor device are synchronized to the standby module:

- Neighbor device ID
- Neighbor device IP address
- Destination device or backup destination device information
- Neighbor state 2way or full
- MD5 information
- Neighbor priority

## Synchronization limitations

- If a neighbor device is inactive for 30 seconds, and if the standby module takes over in another 10 seconds, the neighbor device cannot be dropped. The inactivity timer starts again and takes another 40 seconds to drop the neighbor device.
- In standby module, the valid neighbor states are loading, down, 2way, and full. If the active management processor (MP) fails when the neighbor state is loading, the standby module cannot continue from loading, but the standby can continue from 2way and tries to establish adjacency between the neighboring devices.
- The minimum OSPF dead-interval timer value is 40 seconds. When the dead-interval value is configured to less than this minimum value, OSPF NSR cannot be supported.

## Interface synchronization

Interface information is synchronized for interfaces such as PTPT, broadcast, and non-broadcast. Interface wait time is not synchronized to the standby module. If an interface waits for 30 seconds to determine the identity of the designated router (DR) or the backup designated router (BDR), and if the standby module takes over, the wait timer starts again and takes another 40 seconds for the interface state to change from waiting to BDR, DR, or DROther.

# Standby module operations

The standby management module with OSPF configuration performs the following functions:

## Neighbor database

Neighbor information is updated in the standby module based on updates from the active module. Certain neighbor state and interface transitions are synchronized to the standby module. By default, the neighbor timers on the standby module are disabled.

## LSA database

The standby module processes LSA synchronization events from the active module and unpacks the LSA synchronization information to directly install it in its LSDB, as the LSA has already been processed on the active module. The information required to install all types of LSAs (and special LSAs such as Grace LSAs) is packed by OSPF on the active module in the synchronization buffer, so that you can directly install LSAs on the standby module without extra processing.

The standby module is not allowed to originate any LSAs of its own. This is to maintain all information consistently from the active module. The active module synchronizes self-originated LSAs to the standby module.

LSA aging is not applicable on the standby module. During synchronization from the active module, the current LSA age is recorded and the new database timestamp is created on the standby module to later derive the LSA age as needed.

When the active module sends the LSAs to the standby module, based on the message, the standby module deletes or updates its LSDB with the latest information.

LSA acknowledging or flooding are not done on the standby module. When the LSA synchronization update arrives from the active module, it will be directly installed into the LSDB.

# OSPFv2 distribute list

A distribution list can be configured to explicitly deny specific routes from being eligible for installation in the IP route table. By default, all OSPFv2 routes in the OSPFv2 route table are eligible for installation in the IP route table. Receipt of LSAs are not blocked for the denied routes. The device still receives the routes and installs them in the OSPFv2 database. The denied OSPFv2 routes cannot be installed into the IP route table.

The OSPFv2 distribution list can be managed using ACLs or route maps to identify routes to be denied as described in the following sections:

- Configuring an OSPFv2 Distribution List using ACLs
- Configuring an OSPFv2 Distribution List using route maps

## Configuring an OSPFv2 distribution list using ACLs

To configure an OSPFv2 distribution list using ACLs:

- Configure an ACL that identifies the routes you want to deny. Using a standard ACL allows you deny routes based on the destination network, but does not filter based on the network mask. To also filter based on the network mask of the destination network, use an extended ACL.
- Configure an OSPFv2 distribution list that uses the ACL as input.

### *Examples*

In the following configuration example, the first three commands configure a standard ACL that denies routes to any 10.x.x.x destination network and allows all other routes for eligibility to be installed in the IP route table. The last three commands change the CLI to the OSPFv2 configuration level and configure an OSPFv2 distribution list that uses the ACL as input. The distribution list prevents routes to

any 10.x.x.x destination network from entering the IP route table. The distribution list does not prevent the routes from entering the OSPFv2 database.

```
device(config)# ip access-list standard no_ip
device(config-std-nacl)# deny 10.0.0.0 0.255.255.255
device(config-std-nacl)# permit any
device(config)# router ospf
device(config-ospf-router) # area 0
device(config-ospf-router) # distribute-list no_ip in
```

In the following example, the first three commands configure an extended ACL that denies routes to any 10.31.39.x destination network and allows all other routes for eligibility to be installed in the IP route table. The last three commands change the CLI to the OSPFv2 configuration level and configure an OSPFv2 distribution list that uses the ACL as input. The distribution list prevents routes to any 10.31.39.x destination network from entering the IP route table. The distribution list does not prevent the routes from entering the OSPFv2 database.

```
device(config)# ip access-list extended DenyNet39
device(config-ext-nacl)# deny ip 10.31.39.0 0.0.0.255 any
device(config-ext-nacl)# permit ip any any
device(config)# router ospf
device(config-ospf-router) # area 0
device(config-ospf-router) # distribute-list DenyNet39 in
```

In the following example, the first command configures a numbered ACL that denies routes to any 10.31.39.x destination network and allows all other routes for eligibility to be installed in the IP route table. The last three commands change the CLI to the OSPFv2 configuration level and configure an OSPF distribution list that uses the ACL as input. The distribution list prevents routes to any 10.31.39.x destination network from entering the IP route table. The distribution list does not prevent the routes from entering the OSPFv2 database.

```
device(config)# ip access-list 100 deny ip 10.31.39.0 0.0.0.255 any
device(config)# ip access-list 100 permit ip any any
device(config)# router ospf
device(config-ospf-router) # area 0
device(config-ospf-router) # distribute-list 100 in
```

# Configuring an OSPFv2 distribution list using route maps

You can manage an OSPFv2 distribution list using route maps that apply match operations as defined by an ACL or an IP prefix list. You can also use other options available within the route maps and ACLs to further control the contents of the routes that OSPFv2 provides to the IP route table. This section describes an example of an OSPFv2 distribution list using a route map to specify an OSPFv2 administrative distance for routes identified by an IP prefix list.

To configure an OSPFv2 distribution list using route maps:

- Configure a route map that identifies the routes you want to manage
- Optionally configure an OSPFv2 administrative distance to apply to the OSPFv2 routes
- Configure an OSPFv2 distribution list that uses the route map as input

In the following example, the first two commands identify two routes using the **ip prefix-list test1** command. Next, a route map is created using the **prefix-list test1** command to identify the two routes and the **set distance** command to set the OSPFv2 administrative distance of those routes to 200. A distribution list is then configured under the OSPFv2 configuration that uses the route map named "setdistance" as input.

```
device(config)# ip prefix-list test1 seq 5 permit 10.100.1.0/24
device(config)# ip prefix-list test1 seq 10 permit 10.100.2.0/24
device(config)# route-map setdistance permit 1
device(config-routemap setdistance)# match ip address prefix-list test1
device(config-routemap setdistance)# set distance 200
device(config-routemap setdistance)# exit
device(config)# route-map setdistance permit 2
```

```
device(config-routemap setdistance)# exit
device(config)# router ospf
device(config-ospf-router)# area 0
device(config-ospf-router)# area 1
device(config-ospf-router)# distribute-list route-map setdistance in
device(config-ospf-router)# exit
```

Once this configuration is implemented, the routes identified by the **ip prefix-list** command and matched in the route map will have their OSPFv2 administrative distance set to 200. This is displayed in the output from the **show ip route** command, as shown below.

```
device# show ip route
Total number of IP routes: 4
Type Codes - B:BGP D:Connected I:ISIS S:Static R:RIP O:OSPF; Cost - Dist/Metric
      Destination        Gateway     Port       Cost          Type
1     10.1.1.0/24        DIRECT      eth 1/1    0/0                    D
2     10.100.1.0/24      10.1.1.1    eth 1/1    200/2       O
3     10.100.2.0/24      10.1.1.1    eth 1/1    200/10      02
4     10.100.6.0/24      10.1.1.1    eth 1/1    110/2       O
```

Routes 1 and 2 demonstrate the actions of the example configuration as both display an OSPFv2 administrative distance value of 200. Note that the value is applied to both OSPFv2 learned routes that match the route-map instance containing the set distance clause. The other OSPFv2 route (route 3), which does not match the relevant instance, continues to have the default OSPFv2 administrative distance of 110.

# OSPFv2 route redistribution

Route redistribution imports and translates different protocol routes into a specified protocol type. On the device, redistribution is supported for static routes, ISIS, OSPF, RIP, and BGP. OSPF redistribution supports the import of static, ISIS, RIP, and BGP routes into OSPF routes.

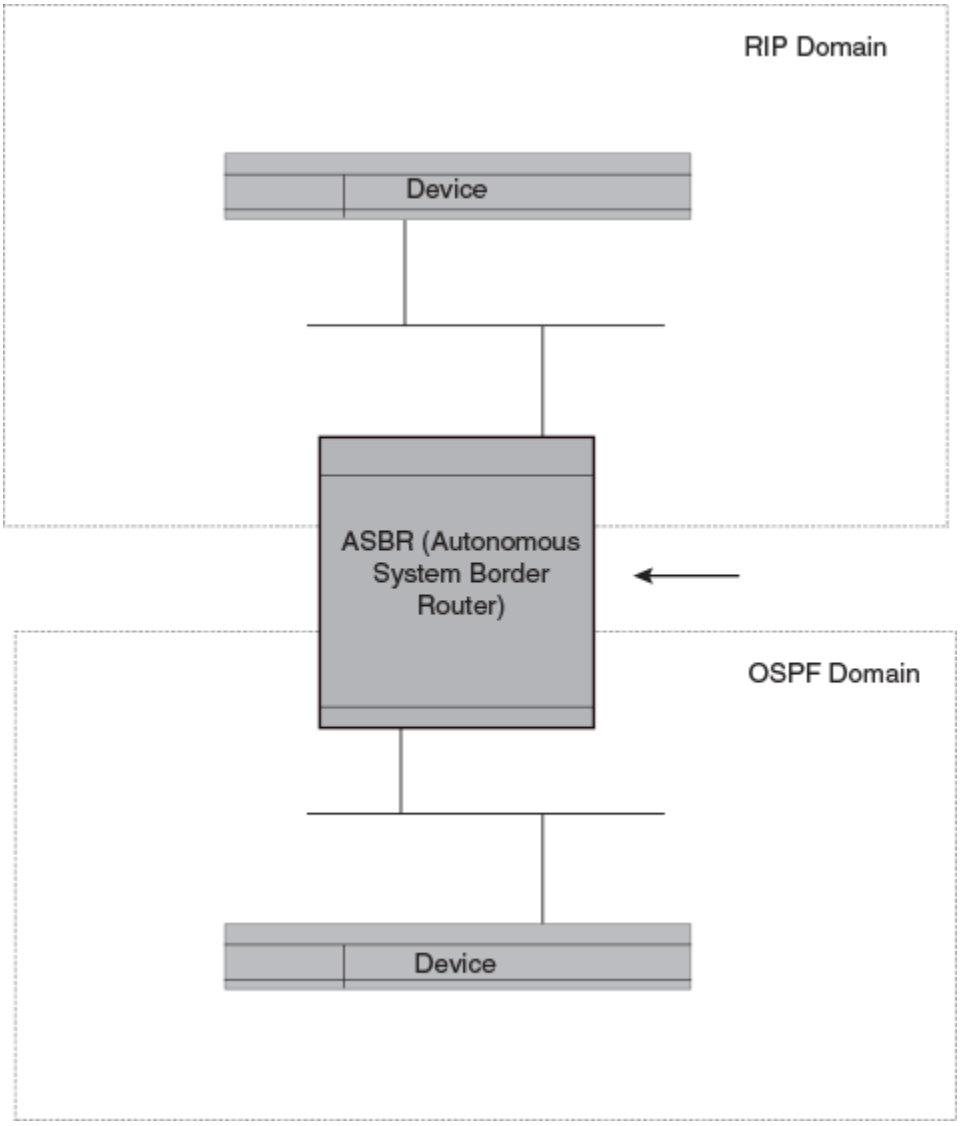
> NOTE
> The device advertises the default route into OSPF even if redistribution is not enabled, and even if the default route is learned through an IBGP neighbor. IBGP routes (including the default route) are not redistributed into OSPF by OSPF redistribution (for example, by the OSPF **redistribute** command).

In the figure below, the device acting as the ASBR (Autonomous System Boundary Router) can be configured between the RIP domain and the OSPF domain to redistribute routes between the two domains.

> NOTE
> The ASBR must be running both RIP and OSPF protocols to support this activity.

> NOTE
> Do not enable redistribution until you have configured the redistribution route map. Otherwise, you might accidentally overload the network with routes you did not intend to redistribute.

**FIGURE 61** Redistributing OSPF and static routes to RIP routes




# Redistributing routes into OSPFv2

OSPFv2 routes can be redistributed, and the routes to be redistributed can be specified.

The redistribution of RIP and static IP routes into OSPFv2 is configured on a device.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **router ospf** command to enter OSPFv2 router configuration mode and enable OSPFv2 on the device.

```
device(config)# router ospf
```

3. Enter the **redistribute** command with the **static** parameter to redistribute static routes.

```
device(config-ospf-router)# redistribute static
```

4. Enter the **redistribute** command with the **rip** parameter to redistribute RIP routes.

```
device(config-ospf-router)# redistribute rip
```

The following example redistributes static and RIP routes into OSPFv2 on a device.

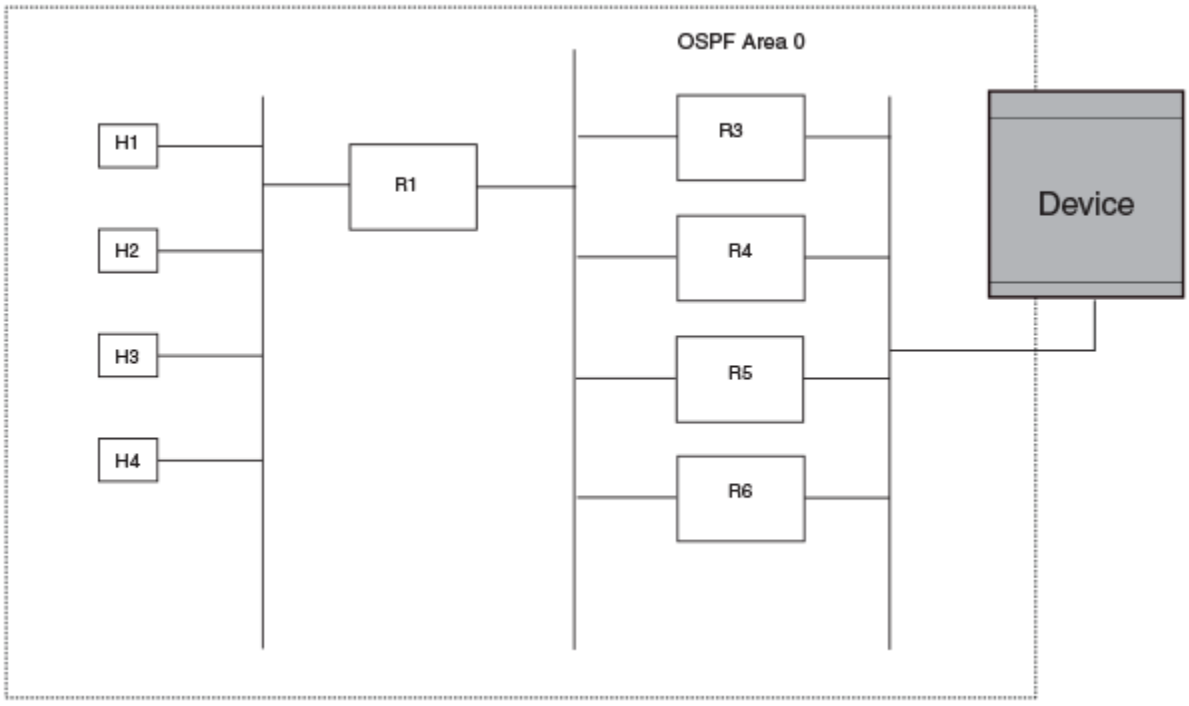
```
device# configure terminal
device(config)# router ospf
device(config-ospf-router)# redistribute static
device(config-ospf-router)# redistribute rip
```

# Load sharing

Extreme devices can load share among up to eight equal-cost IP routes to a destination. By default, IP load sharing is enabled. The default is 4 equal-cost paths but you can specify from 2 to 8 paths.

The device software can use the route information it learns through OSPF to determine the paths and costs.

**FIGURE 62** Example OSPF network with four equal-cost paths




The device has four paths to R1:

- Router ->R3
- Router ->R4
- Router ->R5
- Router ->R6

Normally, the device chooses the path to the R1 with the lower metric. For example, if the metric for R3 is 1400 and the metric for R4 is 600, the device always chooses R4.

However, suppose the metric is the same for all four routers in this example. If the costs are the same, the device now has four equal-cost paths to R1. To allow the device to load share among the equal cost routes, enable IP load sharing. Four equal-cost OSPF paths are supported by default when you enable load sharing.

> NOTE
> The device is not source routing in these examples. The device is concerned only with the paths to the next-hop routers, not the entire paths to the destination hosts.

OSPF load sharing is enabled by default when IP load sharing is enabled.

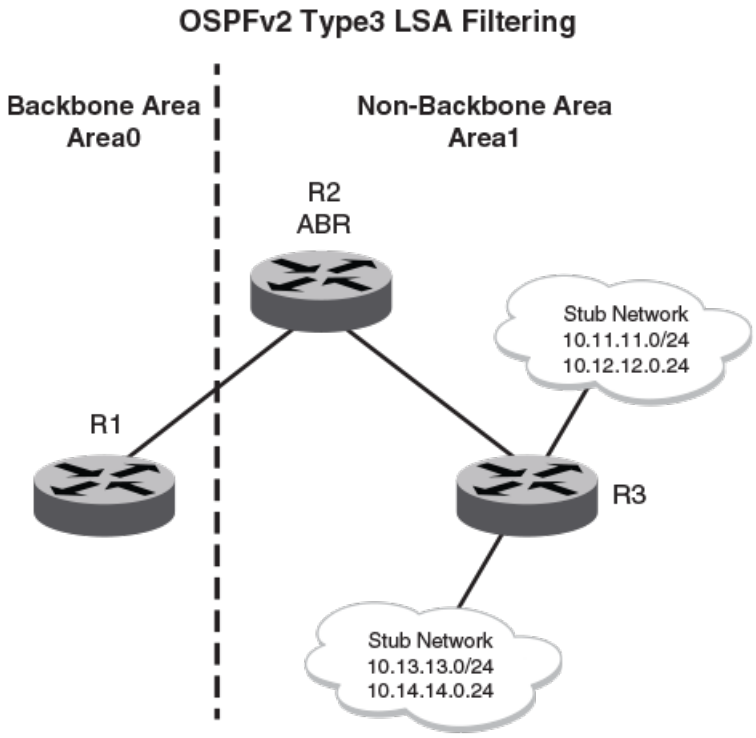
# OSPFv2 type 3 LSA filtering

OSPFv2 type 3 LSA filtering provides an ABR that is running the OSPFv2 protocol with the ability to filter type 3 link-state advertisements (LSAs) that are sent between different OSPFv2 areas. Filtering of routes can be defined using prefix-list filters to either permit or deny certain prefixes. Only specified prefixes can be sent from one area to another area and all other prefixes are prohibited. OSPFv2 type 3 LSA filtering can be applied for the LSAs coming into a specific OSPFv2 area or going out of a specific OSPFv2 area, or into and out of the same OSPFv2 areas concurrently. Any change in the prefix-list used for type 3 filtering may result in the advertisement of new summary LSAs or the withdrawal of previously advertised summary LSAs.

Type 3 LSAs refer to summary links and are sent by ABRs to advertise destinations outside the area. OSPFv2 type 3 LSA filtering gives the administrator improved control of route distribution between OSPFv2 areas.

In certain situations, reachability for some IP network prefixes from outside of an area or to a specific area should be restricted. Such a situation is illustrated in the figure below where a device called R3 is advertising stub networks that belong to the non-backbone area (Area1). An ABR, R2, will generate summary routes for these prefixes. If OSPFv2 type 3 LSA filtering is not configured, all the stub networks are seen as summary LSAs in the backbone area (Area 0) and are flooded to all applicable OSPFv2 devices. These type 3 LSAs can be filtered out using the OSPFv2 type3 LSA Filter.

**FIGURE 63** OSPFv2 type 3 LSA filtering




## Usage and configuration guidelines

- OSPFv2 type 3 LSA filtering is only applicable to ABRs. Configurations are accepted prior to the device becoming an ABR but OSPFv2 type 3 LSA filtering only occurs once the device becomes an ABR.
- When OSPFv2 type 3 LSA filtering is enabled in the "in" direction, all type 3 LSAs originated by the ABR to this area are filtered by the prefix list, based on information from all other areas. Type 3 LSAs, originated as a result of the **area range** command in a different area, are treated like any other individuallly originated type 3 LSA. Any prefix that does not match an entry in the prefix list is implicitly denied.
- When OSPFv2 type 3 LSA filtering is enabled in the "out" direction, all type 3 LSAs advertised by the ABR are filtered by the prefix list, based on information from this area to all other areas. If the **area range** command has been used to configure type 3 LSAs for this area, these type 3 LSAs that correspond to these configurations are treated like any other type 3 LSA. Prefixes that do not match are implicitly denied.

**TABLE 52** Behavior for prefix list configurations

| IP prefix list | OSPF area prefix list | Event | Filtering done |
|---|---|---|---|
| XXX | Not defined | None | No (permit all) |
| Not defined | Defined | None | Yes (deny all) |
| Not defined | Defined | IP prefix list defined | Recalculation |
| Defined (no rules configured) | Defined | None | Implicit deny (deny all) |
| Defined (rules configured) | Defined | IP prefix list deleted | Recalculation and deny all |
| Defined (rules configured) | Defined | IP prefix list rule added or modified or deleted | Recalculation |

TABLE 52 Behavior for prefix list configurations (continued)

| IP prefix list | OSPF area prefix list | Event | Filtering done |
|---|---|---|---|
| Defined (rules configured) | Defined | Area prefix list deleted | Recalculation and permit all |

## Configuring OSPFv2 type 3 LSA filtering

An IP prefix list can be configured and applied to an OSPFv2 area.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **ip prefix-list** command and specify a name.

```
device(config)# ip prefix-list mylist permit 10.1.0.0/16
```

   Configures an IP prefix list named "mylist" that permits routes to network 10.1.0.0/16.

3. Enter the **router ospf** command to enter OSPF router configuration mode and enable OSPFv2 on the device.

```
device(config)# router ospf
```

4. Enter the **area** command to define an OSPFv2 area ID.

```
device(config-ospf-router)# area 1
```

5. Enter the **area prefix-list** command and specify an area address and a prefix-list.

```
device(config-ospf-router)# area 1 prefix-list mylist out
```

   Configures the device to use the IP prefix list "mylist" to determine which routes from Area 1 are advertised to other areas. The device advertises routes that fall within the 10.1.0.0/16 range to other areas because the IP prefix list explicitly permits these routes to be sent to other areas from Area 1.

The following example onfigures an IP prefix list named "mylist" that permits routes to network 10.1.0.0/16. This list is used to determine which routes to send to Area 1.

```
device# configure terminal
device(config)# ip prefix-list mylist permit 10.1.0.0/16
device(config)# router ospf
device(config-ospf-router)# area 1
device(config-ospf-router)# area 1 prefix-list mylist out
```

# Interface types to which the reference bandwidth does not apply

Some interface types are not affected by the reference bandwidth and always have the same cost regardless of the reference bandwidth in use:

- The cost of a loopback interface is always 1.
- The cost of a virtual link is calculated using the Shortest Path First (SPF) algorithm and is not affected by the auto-cost feature.
- The bandwidth for tunnel interfaces is 9 Kbps and is also subject to the auto-cost reference bandwidth setting.

# Changing the reference bandwidth for the cost on OSPFv2 interfaces

Each interface on which OSPFv2 is enabled has a cost associated with it. The device advertises its interfaces and their costs to OSPFv2 neighbors. For example, if an interface has an OSPFv2 cost of ten, the device advertises the interface with a cost of ten to other OSPFv2 routers.

By default, an interface's OSPFv2 cost is based on the port speed of the interface. The cost is calculated by dividing the reference bandwidth by the port speed. The default reference bandwidth is 100 Mbps, which results in the following default costs:

- 10 Mbps port - 10
- All other port speeds - 1

You can change the reference bandwidth. The following formula is used to calculate the cost:

Cost = reference-bandwidth/interface-speed

If the resulting cost is less than 1, the cost is rounded up to 1. The default reference bandwidth results in the following costs:

- 10 Mbps port's cost = 100/10 = 10
- 100 Mbps port's cost = 100/100 = 1
- 1000 Mbps port's cost = 100/1000 = 0.10, which is rounded up to 1
- 10 Gbps port's cost = 100/10000 = 0.01, which is rounded up to 1

The bandwidth for interfaces that consist of more than one physical port is calculated as follows:

- LAG group - The combined bandwidth of all the ports.
- Virtual interface - The combined bandwidth of all the ports in the port-based VLAN that contains the virtual interface.

The default reference bandwidth is 100 Mbps. You can change the reference bandwidth to a value from 1—4294967.

If a change to the reference bandwidth results in a cost change to an interface, the device sends a link-state update to update the costs of interfaces advertised by the device.

NOTE
If you specify the cost for an individual interface, the cost you specify overrides the cost calculated by the software.

# OSPFv2 over VRF

OSPFv2 can run over multiple Virtual Routing and Forwarding (VRF) instances. All OSPFv2 commands are available over default and non-default OSPF instances.

OSPFv2 maintains multiple instances of the routing protocol to exchange route information among various VRF instances. A multi-VRF-capable device maps an input interface to a unique VRF, based on user configuration. These input interfaces can be physical or a virtual interface. By default, all input interfaces are attached to the default VRF instance.

Multi-VRF for OSPF (also known as VRF-Lite for OSPF) provides a reliable mechanism for trusted VPNs to be built over a shared infrastructure. The ability to maintain multiple virtual routing or forwarding tables allows overlapping private IP addresses to be maintained across VPNs.

## Enabling OSPFv2 in a non-default VRF

When OSPFv2 is enabled in a non-default VRF instance, the device enters OSPF router VRF configuration mode. Several commands can then be accessed that allow the configuration of OSPFv2.

A non-default VRF instance has been configured.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **router ospf** command and specify a VRF name to enter OSPF router VRF configuration mode and enable OSPFv2 on a non-default VRF.

```
device(config)# router ospf vrf green
```

The following example enables OSPFv2 in a non-default VRF.

```
device# configure terminal
device(config)# router ospf vrf green
device(config-ospf-router-vrf-green)#
```

# OSPF VRF-Lite for customer edge routers

When a type 3, 5, or 7 LSA is sent from a provider edge (PE) router to a customer edge (CE) router, the DN (down) bit in the LSA options field must be set. This prevents any type 3, 5, or 7 LSA messages sent from the CE router to the PE router from being distributed any farther. The PE router ignores messages with the DN bit set and does not add these routes to the VRF routing table.

When you enable VRF-Lite on the CE router, the DN setting is ignored, allowing the CE router to add these routes to the VRF routing table.

To enable VRF-Lite, enter commands such as the following:

```
device(config)# router ospf vrf 1
device(config-ospf-router-vrf-1)# vrf-lite-capability
```

**Syntax: [no] vrf-lite-capability**

Use the **no** form of the command to disable VRF-Lite. This applies to the VRF instance only. It does not apply to the default VRF.

NOTE
For vpn4 external routes to be installed on CE routers, the domain-tags on PE routers must be different that the domain-tags on CE routers.

NOTE
This command applies to CE routers only. This command does not apply to PE routers.

# Configuring the OSPFv2 Max-Metric Router LSA

By configuring the OSPFv2 max-metric router LSA you can enable OSPFv2 to advertise its locally generated router LSAs with a maximum metric.

NOTE
You can configure OSPFv2 max-metric router LSA in either startup or non-startup mode. When you configure max-metric in non-startup mode, it only applies once and is not persistent across reloads or after the **clear ip ospf all** command is issued.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **router ospf** command to enter OSPF router configuration mode and enable OSPFv2 on the device.

```
device(config)# router ospf
```

3. Enter the **max-metric router-lsa** command with the **on-startup** keyword and specify a value to specify a period of time to advertise a maximum metric after a restart before advertising with a normal metric.

```
device(config-ospf-router)# max-metric router-lsa on-startup 85
```

The following example configures an OSPFv2 device to advertise a maximum metric for 85 seconds after a restart before advertising with a normal metric.

```
device# configure terminal
device(config)# router ospf
device(config-ospf-router)# max-metric router-lsa on-startup 85
```

# Changing default settings

Refer to the relevant Command Reference for other commands you can use to change default OSPF settings. Some commonly configured items include the following:

- Changing reference bandwidth to change interface costs by using the **auto-cost reference-bandwidth** command.
- Defining redistribution filters for the Autonomous System Boundary Router (ASBR) by using the **redistribute** command.

# Disabling and re-enabling OSPFv2 event logging

OSPFv2 event logging can be configured, disabled, and re-enabled.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **router ospf** command to enter OSPF router configuration mode and enable OSPFv2 globally.

```
device(config)# router ospf
```

3. Enter the **no log all** command to disable the logging of all OSPFv2 events.

```
device(config-ospf-router)# no log all
```

The following example re-enables the logging of all OSPFv2 events.

```
device# configure terminal
device(config)# router ospf
device(config-ospf-router)# log all
```

# Understanding the effects of disabling OSPFv2

Consider the following before disabling OSPFv2 on a device:

- If you disable OSPFv2, the device removes all the configuration information for the disabled protocol from the running configuration. Moreover, when you save the configuration to the running configuration file after disabling one of these protocols, all the configuration information for the disabled protocol is removed from the running configuration file.
- If you have disabled the protocol but have not yet saved the configuration to the running configuration file and reloaded the software, you can restore the configuration information by re-entering the **router ospf** command, or by selecting the Web management option to enable the protocol. If you have already saved the configuration to the running configuration file and reloaded the software, the information is gone.
- If you are testing an OSPFv2 configuration and are likely to disable and re-enable the protocol, you might want to make a backup copy of the running configuration file containing the protocol's configuration information. This way, if you remove the configuration information by saving the configuration after disabling the protocol, you can restore the configuration by copying the backup copy of the startup configuration file into the flash memory.

## Disabling OSPFv2

To disable OSPFv2 on a device, use the **no router ospf** command:

1. Enter the **configure terminal** command to access global configuration mode.

   ```
   device# configure terminal
   ```

2. Enter the **no router ospf** command to disable OSPFv2 on the device.

   ```
   device(config)# no router ospf
   ```

The following example disables OSPFv2 on a device.

```
device# configure terminal
device(config)# no router ospf
```

# OSPFv3



# OSPFv3 overview

Open Shortest Path First (OSPF) is a link-state routing protocol. Each OSPF device originates link-state advertisement (LSA) packets to describe its link information. These LSAs are flooded throughout the OSPF area. The flooding algorithm ensures that every device in the area has an identical database. Each device in the area then calculates a Shortest Path Tree (SPT) that shows the shortest distance to every other device in the area, using the topology information in the Link State database..

IPv6 supports OSPF Version 3 (OSPFv3), which functions similarly to OSPFv2, the version that IPv4 supports, except for the following enhancements:

- Support for IPv6 addresses and prefixes.
- Ability to configure several IPv6 addresses on a device interface. (While OSPFv2 runs per IP subnet, OSPFv3 runs per link. In general, you can configure several IPv6 addresses on a router interface, but OSPFv3 forms one adjacency per interface only, using the link local address of the interface as the source for OSPF protocol packets. On virtual links, OSPFv3 uses the global IP address as the source. OSPFv3 imports all or none of the address prefixes configured on a router interface. You cannot select the addresses to import.)
- Ability to run one instance of OSPFv2 and one instance of OSPFv3 concurrently on a link.
- Support for IPv6 link-state advertisements (LSAs).

NOTE
Although OSPFv2 and OSPFv3 function in a similar manner, Extreme has implemented the user interface for each version independently of the other. Therefore, any configuration of OSPFv2 features will not affect the configuration of OSPFv3 features and vice versa.

# Configuring the router ID

When configuring OSPFv3, the router ID for a device must be specified.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **ip router-id** command to specify the router ID.

```
device(config) ip router-id 10.11.12.13
```

The following example configures the router ID for a device.

```
device# configure terminal
device(config)# ip router-id 10.11.12.13
```

# Enabling OSPFv3

When OSPFv3 is enabled on a device, the device enters OSPFv3 router configuration mode. Several commands can then be accessed that allow the configuration of OSPFv3.

Before enabling the device to run OSPFv3, you must perform the following steps:

- Enable the forwarding of IPv6 traffic on the device using the **ipv6 unicast-routing** command.
- Enable IPv6 on each interface on which you plan to enable OSPFv3. You enable IPv6 on an interface by configuring an IPv6 address or explicitly enabling IPv6 on that interface.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **ip router-id** command to specify the router ID.

```
device(config) ip router-id 10.11.12.13
```

3. Enter the **ipv6 router ospf** command to enter OSPFv3 router configuration mode and enable OSPFv3 on the device.

```
device(config)# ipv6 router ospf
```

The following example enables OSPFv3 on a device.

```
device# configure terminal
device(config)# ip router-id 10.11.12.13
device(config)# ipv6 router ospf
device(config-ospf6-router)#
```

# Configuring OSPFv3

A number of steps are required when configuring OSPFv3:

- Configure the router ID.
- Enable OSPFv3 globally.
- Assign OSPFv3 areas.
- Assign OSPFv3 areas to interfaces.

# OSPFv3 areas

After OSPFv3 is enabled, you can assign OSPFv3 areas. You can specify the area id in plain number format , such as "area 1", or in ipv4 address format, such as 10.1.1.1. Each device interface can support one area.

NOTE
You can assign only one area on a device interface.

NOTE
You are required to configure a router ID when running only IPv6 routing protocols.

NOTE
By default, the router ID is the IPv4 address configured on the lowest-numbered loopback interface. If the device does not have a loopback interface, the default router ID is the highest-numbered IPv4 address configured on the device. You can also configure router id using the **ip router-id** command.

## Backbone area

The backbone area (also known as area 0 or area 0.0.0.0) forms the core of OSPF networks. All other areas should be connected to the backbone area either by a direct link or by virtual link configuration. Routers that have interfaces in both backbone area and (at least one) non-backbone area are called Area Border Routers (ABR). Inter area routing happens via ABRs.

The backbone area is the logical and physical structure for the OSPF domain and is attached to all non-zero areas in the OSPF domain.

The backbone area is responsible for distributing routing information between non-backbone areas. The backbone must be contiguous, but it does not need to be physically contiguous; backbone connectivity can be established and maintained through the configuration of virtual links.

## Area range

You can further consolidate routes at an area boundary by defining an area range. The area range allows you to assign an aggregate address to a range of IP and IPv6 addresses.

This aggregate value becomes the address that is advertised instead of all the individual addresses it represents being advertised. Only this aggregate or summary address is advertised into other areas instead of all the individual addresses that fall in the configured range. Area range configuration can considerably reduce the number of Type 3 summary LSAs advertised by a device. You have the option of adding the cost to the summarized route. If you do not specify a value, the cost value is the default range metric calculation for the generated summary LSA cost. You can temporarily pause route summarization from the area by suppressing the type 3 LSA so that the component networks remain hidden from other networks.

You can assign up to 32 ranges in an OSPF area.

## Area types

OSPFv3 areas can be normal, a stub area, a totally stubby area (TSA), or a not-so-stubby area (NSSA).

- Normal: OSPFv3 devices within a normal area can send and receive external link-state advertisements (LSAs).
- Stub: OSPFv3 devices within a stub area cannot send or receive External LSAs. In addition, OSPF devices in a stub area must use a default route to the area's Area Border Router (ABR) to send traffic out of the area.
- TSA: A form of stub area, where Type 3 summary routes are also not propagated in addition to Type 5 external routes.

- NSSA: A form of stub area, where Type 5 external routes by Autonomous System Boundary Routers (ASBRs) outside this area are not propagated, but where it is allowed to have an ASBR in the area, that can advertise external information.
  - ASBRs redistribute (import) external routes into the NSSA as type 7 LSAs. Type 7 External LSAs are a special type of LSA generated only by ASBRs within an NSSA, and are flooded to all the routers within only that NSSA.
  - One of the ABRs of the NSSA area is selected as a NSSA translator, and this router translates the area-specific Type 7 LSAs to Type 5 external LSAs which can be flooded throughout the Autonomous System (except NSSA and stub areas).

When an NSSA contains more than one ABR, OSPFv3 elects one of the ABRs to perform the LSA translation for NSSA. OSPF elects the ABR with the highest router ID. If the elected ABR becomes unavailable, OSPFv3 automatically elects the ABR with the next highest router ID to take over translation of LSAs for the NSSA. The election process for NSSA ABRs is automatic.

# Assigning OSPFv3 areas

Areas can be assigned as OSPFv3 areas.

Enable IPv6 on each interface on which you plan to enable OSPFv3. You enable IPv6 on an interface by configuring an IP address or explicitly enabling IPv6 on that interface.

1. Enter the **configure terminal** command to access global configuration mode.

   ```
   device# configure terminal
   ```

2. Enter the **ip router-id** command to specify the router ID.

   ```
   device(config) ip router-id 10.11.12.13
   ```

3. Enter the **ipv6 router ospf** command to enter OSPFv3 router configuration mode and enable OSPFv3 on the device.

   ```
   device(config)# ipv6 router ospf
   ```

4. Enter the **area** command to define an OSPFv3 area ID.

   ```
   device(config-ospf6-router)# area 0
   ```

5. Enter the **area** command to define a second OSPFv3 area ID.

   ```
   device(config-ospf6-router)# area 10.1.1.1
   ```

The following example assigns an OSPFv3 ID to two areas. One of the areas is assigned by decimal number. The second area is assigned by IP address.

```
device# configure terminal
device(config)# ip router-id 10.11.12.13
device(config)# ipv6 router ospf
device(config-ospf6-router)# area 0
device(config-ospf6-router)# area 10.1.1.1
```

# Assigning OSPFv3 areas to interfaces

Defined OSPFv3 areas can be assigned to device interfaces.

Ensure that OSPFv3 areas are assigned.

NOTE
All device interfaces must be assigned to one of the defined areas on an OSPFv3 device. When an interface is assigned to an area, all corresponding subnets on that interface are automatically included in the assignment.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **interface** command and specify an interface.

```
device(config)# interface ethernet 1/1
```

3. Enter the **ipv6 address** command to add an IPv6 address to the interface.

```
device(config-if-e1000/1/1)# ipv6 address 2001:1:0::1:1/64
```

4. Enter the **ipv6 ospf area** command.

```
device(config-if-e1000/1/1)# ipv6 ospf area 0
```

Area 0 is assigned to the specified interface with the IPv6 address of 2001:1:0:1::1/64.

5. Enter the **exit** command to return to global configuration mode.

```
device(config-if-e1000/1/1)# exit
```

6. Enter the **interface** command and specify an interface.

```
device(config)# interface ethernet 2/1
```

7. Enter the **ipv6 address** command to add an IPv6 address to the interface.

```
device(config-if-e1000/2/1)# ipv6 address 2001:1:0::2:1/64
```

8. Enter the **ipv6 ospf area** command.

```
device(config-if-e1000/2/1)# ipv6 ospf area 1
```

Area 1 is assigned to the specified interface with the IPv6 address of 2001:1:0:2::1/64.

The following example configures and enables OSPFv3 on two specified interfaces, and assigns an interface to two router areas.

```
device# configure terminal
device(config)# interface ethernet 1/1
device(config-if-e1000/1/1)# ipv6 address 2001:1:0::1:1/64
device(config-if-e1000/1/1)# ipv6 ospf area 0
device(config-if-e1000/1/1)# exit
device(config)# interface ethernet 2/1
device(config-if-e1000/2/1)# ipv6 address 2001:1:0::2:1/64
device(config-if-e1000/2/1)# ipv6 ospf area 1
```

# Stub area and totally stubby area

A stub area is an area in which advertisements of external routes are not allowed, reducing the size of the database. A totally stubby area (TSA) is a stub area in which summary link-state advertisement (type 3 LSAs) are not sent. A default summary LSA, with a prefix of 0.0.0.0/0 is originated into the stub area by an ABR, so that devices in the area can forward all traffic for which a specific route is not known, via ABR.

A stub area disables advertisements of external routes. By default, the ABR sends summary LSAs (type 3 LSAs) into stub areas. You can further reduce the number of LSAs sent into a stub area by configuring the device to stop sending type 3 LSAs into the area. You can disable the summary LSAs to create a TSA when you are configuring the stub area or after you have configured the area.

The ABR of a totally stubby area disables origination of summary LSAs into this area, but still accepts summary LSAs from OSPF neighbors and floods them to other neighbors.

When you enter the **area stub** command with the **no-summary** keyword and specify an area to disable the summary LSAs, the change takes effect immediately. If you apply the option to a previously configured area, the device flushes all the summary LSAs it has generated (as an ABR) from the area with the exception of the default summary LSA originated. This default LSA is needed for the internal routers, since external routes are not propagated to them.

NOTE

Stub areas and TSAs apply only when the device is configured as an Area Border Router (ABR) for the area. To completely prevent summary LSAs from being sent to the area, disable the summary LSAs on each OSPF router that is an ABR for the area.

# Configuring a stub area

OSPFv3 areas can be defined as stub areas with modifiable parameters.

1. Enter the **configure terminal** command to access global configuration mode.

   ```
   device# configure terminal
   ```

2. Enter the **ip router-id** command to specify the router ID.

   ```
   device(config) ip router-id 10.4.4.4
   ```

3. Enter the **ipv6 router ospf** command to enter OSPFv3 router configuration mode and enable OSPFv3 on the device.

   ```
   device(config)# ipv6 router ospf
   ```

4. Enter the **area stub** command and specify a metric value.

   ```
   device(config-ospf6-router)# area 4 stub 100
   ```

   Area 4 is defined as a stub area with an additional cost of 100.

The following example sets an additional cost of 100 on a stub area defined as 4.

```
device# configure terminal
device(config)# ip router-id 10.4.4.4
device(config)# ipv6 router ospf
device(config-ospf6-router)# area 4 stub 100
```

# Not-so-stubby area

A not-so-stubby-area (NSSA) is an OSPFv3 area that provides the benefits of stub areas with the extra capability of importing external route information. OSPFv3 does not flood external routes from other areas into an NSSA, but does translate and flood route information from the NSSA into other areas such as the backbone.

NSSAs are especially useful when you want to aggregate type 5 External LSAs (external routes) before forwarding them into an OSPFv3 area. When you configure an NSSA, you can specify an address range for aggregating the external routes that the ABR of the NSSAs exports into other areas.

The OSPFv3 specification (RFC 2328) prohibits the advertising of type 5 LSAs and requires OSPFv3 to flood type 5 LSAs throughout a routing domain.

If the router is an ABR, you can prevent any type 3 and type 4 LSA from being injected into the area by configuring a nssa with the **no-summary** parameter. The only exception is that a default route is injected into the NSSA by the ABR, and strictly as a type 3 LSA. The default type 7 LSA is not originated in this case.

By default, the device's NSSA translator role is set to candidate and the router participates in NSSA translation election, if it is an ABR. You can also configure the NSSA translator role.

In the case where an NSSA ABR is also an ASBR, the default behavior is that it originates type 5 LSAs into normal areas and type 7 LSAs into an NSSA. But you can prevent an NSSA ABR from generating type 7 LSAs into an NSSA by configuring the **no-redistribution** parameter.

# Configuring an NSSA

OSPFv3 areas can be defined as NSSA areas with configurable parameters.

1. Enter the **configure terminal** command to access global configuration mode.

   ```
   device# configure terminal
   ```

2. Enter the **ip router-id** command to specify the router ID.

   ```
   device(config) ip router-id 10.3.3.3
   ```

3. Enter the **ipv6 router ospf** command to enter OSPFv3 router configuration mode and enable OSPFv3 on the device.

   ```
   device(config)# ipv6 router ospf
   ```

4. Enter the **area nssa** command with the **default-information-originate** keyword and specify a cost.

   ```
   device(config-ospf6-router)# area 3 nssa default-information-originate metric 33
   ```

   Area 3 is defined as an NSSA with the default route option and an additional cost of 33.

The following example sets an additional cost of 33 on an NSSA defined as 3.

```
device# configure terminal
device(config)# ip router-id 10.3.3.3
device(config)# ipv6 router ospf
device(config-ospf6-router)# area 3 nssa default-information-originate metric 33
```

# LSA types for OSPFv3

Communication among OSPFv3 areas is provided by means of link-state advertisements (LSAs). OSPFv3 supports a number of types of LSAs:

- Router LSAs (Type 1)
- Network LSAs (Type 2)
- Interarea-prefix LSAs for ABRs (Type 3)
- Interarea-router LSAs for ASBRs (Type 4)
- Autonomous system External LSAs (Type 5)
- NSSA External LSAs (Type 7)
- Link LSAs (Type 8)
- Intra-area-prefix LSAs (Type 9)

For more information about these LSAs, refer to RFC 5340.

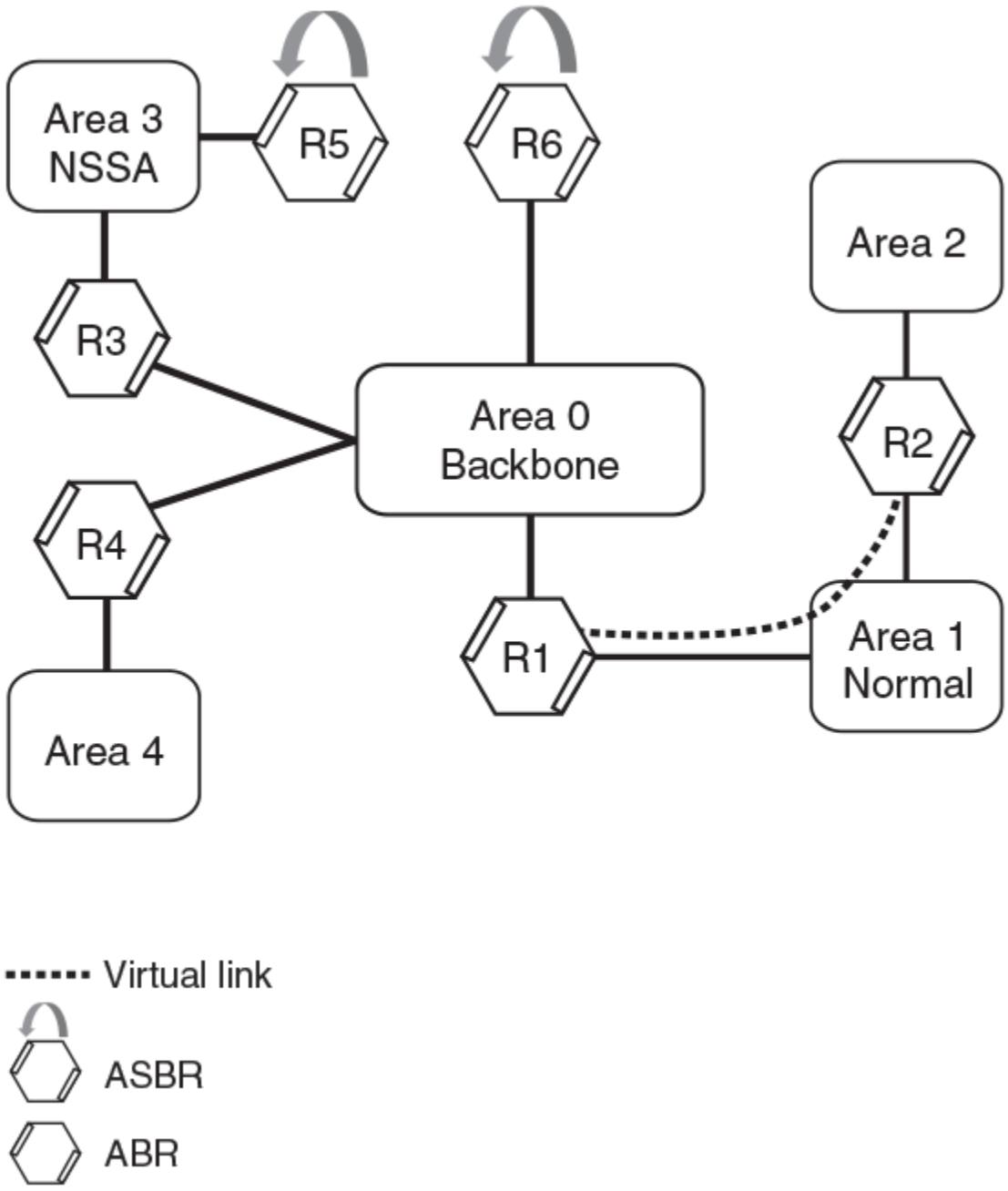
# Virtual links

All ABRs must have either a direct or indirect link to an OSPFv3 backbone area (0 or 0.0.0.0). If an ABR does not have a physical link to a backbone area, you can configure a virtual link from the ABR to another router within the same area that has a physical connection to the backbone area.

The path for a virtual link is through an area shared by the neighbor ABR (router with a physical backbone connection) and the ABR requiring a logical connection to the backbone.

In the following figure, a virtual link has been created between ABR1 and ABR2. ABR1 has a direct link to the backbone area, while ABR2 has an indirect link to the backbone area through Area 1.

**FIGURE 64** OSPFv3 virtual link




Two parameters must be defined for all virtual links—transit area ID and neighbor router:

- The transit area ID represents the shared area of the two ABRs and serves as the connection point between the two routers. This number should match the area ID value.
- The neighbor router is the router ID of the device that is physically connected to the backbone when assigned from the router interface requiring a logical connection. The neighbor router is the router ID (IPv4 address) of the router requiring a logical connection to the backbone when assigned from the router interface with the physical connection.

When you establish an area virtual link, you must configure it on both ends of the virtual link. For example, imagine that ABR1 in Area 1 and Area 2 is cut off from the backbone area (Area 0). To provide backbone access to ABR1, you can add a virtual link between ABR1 and ABR2 in Area 1 using Area 1 as a transit area. To configure the virtual link, you define the link on the router that is at each end of the link. No configuration for the virtual link is required on the routers in the transit area.

Virtual links cannot be configured in stub areas and NSSAs.

## Virtual link source address assignment

When devices at both ends of a virtual link communicate with one another, a global IPv6 address is automatically selected for each end device and this address is advertised into the transit area as an intra-area-prefix LSA.

The automatically selected global IPv6 address for that router is the first global address of any loopback interface in that transit area. If no global IPv6 address is available on a loopback interface in the area, the first global IPv6 address of the lowest-numbered interface in the UP state (belonging to the transit area) is assigned. If no global IPv6 address is configured on any of the OSPFv3 interfaces in the transit area, the virtual links in the transit area do not operate. The automatically selected IPv6 global address is updated whenever the previously selected IPv6 address of the interface changes, is removed, or if the interface goes down.

> NOTE
> The existing selected virtual link address does not change because the global IPv6 address is now available on a loopback interface or a lower-numbered interface in the transit area. To force the global IPv6 address for the virtual link to be the global IPv6 address of a newly configured loopback, or a lower-numbered interface in the area, you must either disable the existing selected interface or remove the currently selected global IPv6 address from the interface.

## Configuring virtual links

If an Area Border Router (ABR) does not have a physical link to a backbone area, a virtual link can be configured between that ABR and another device within the same area that has a physical link to a backbone area.

A virtual link is configured, and a virtual link endpoint on two devices, ABR1 and ABR2, is defined.

1. On ABR1, enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **ip router-id** command to specify the router ID.

```
device(config) ip router-id 10.1.1.1
```

3. Enter the **ipv6 router ospf** command to enter OSPFv3 router configuration mode and enable OSPFv3 on the device.

```
device(config)# ipv6 router ospf
```

4. Enter the **area** command to assign an OSPFv3 area ID.

```
device(config-ospf6-router)# area 0
```

5. Enter the **area** command to assign an OSPFv3 area ID.

```
device(config-ospf6-router)# area 1
```

6. Enter the **area virtual-link** command and the ID of the OSPFv3 device at the remote end of the virtual link to configure the virtual link endpoint.

```
device(config-ospf6-router)# area 1 virtual-link 10.2.2.2
```

7. On ABR2, enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

8. Enter the **ip router-id** command to specify the router ID.

```
device(config) ip router-id 10.2.2.2
```

9. Enter the **ipv6 router ospf** command to enter OSPFv3 router configuration mode and enable OSPFv3 on the device.

```
device(config)# ipv6 router ospf
```

10. Enter the **area** command to assign an OSPFv3 area ID.

```
device(config-ospf6-router)# area 1
```

11. Enter the **area** command to assign an OSPFv3 area ID.

```
device(config-ospf6-router)# area 2
```

12. Enter the **area virtual-link** command and the ID of the OSPFv3 device at the remote end of the virtual link to configure the virtual link endpoint.

```
device(config-ospf6-router)# area 1 virtual-link 10.1.1.1
```

The following example configures a virtual link between two devices.

```
ABR1:
device1# configure terminal
device1(config)# ip router-id 10.1.1.1
device1(config)# ipv6 router ospf
device1(config-ospf6-router)# area 0
device1(config-ospf6-router)# area 1
device1(config-ospf6-router)# area 1 virtual-link 10.2.2.2

ABR2:
device2# configure terminal
device2(config)# ip router-id 10.2.2.2
device2(config)# ipv6 router ospf
device2(config-ospf6-router)# area 1
device2(config-ospf6-router)# area 2
device2(config-ospf6-router)# area 1 virtual-link 10.1.1.1
```

# OSPFv3 route redistribution

Routes from various sources can be redistributed into OSPFv3. These routes can be redistributed in a number of ways.

You can configure the device to redistribute routes from the following sources into OSPFv3:

- IPv6 static routes
- Directly connected IPv6 networks
- BGP4+
- IPv6 IS-IS
- RIPng

You can redistribute routes in the following ways:

- By route types. For example, the device redistributes all IPv6 static routes.
- By using a route map to filter which routes to redistribute. For example, the device redistributes specified IPv6 static routes only.

NOTE
You must configure the route map before you configure a redistribution filter that uses the route map.

NOTE
For an external route that is redistributed into OSPFv3 through a route map, the metric value of the route remains the same unless the metric is set by the **set metric** command inside the route map or the **default-metric** command. For a route redistributed without using a route map, the metric is set by the metric parameter if set or the **default-metric** command if the metric parameter is not set.

NOTE
The CLI VRF-lite-capability present in OSPFv2 is not present in OSPFv3. Therefore, when a BGP route is redistributed from an MPLS domain into OSPFv3 and the DN (down) bit is set, the routes must be installed in the OSPFv3 routing table. Such routes could get propagated back into the MLPS cloud if there are OSPFv3 back-door links configured.

# Redistributing routes into OSPFv3

OSPFv3 routes can be redistributed, and the routes to be redistributed can be specified.

The redistribution of both static routes and BGP routes into OSPFv3 is configured on device1. The redistribution of connected routes into OSPFv3 is configured on device2, and the connected routes to be redistributed are specified.

1. On device1, enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **ipv6 router ospf** command to enter OSPFv3 router configuration mode and enable OSPFv3 on the device.

```
device(config)# ipv6 router ospf
```

3. Enter the **redistribute** command with the **static** parameter to redistribute static routes.

```
device(config-ospf6-router)# redistribute static
```

4. Enter the **redistribute** command with the **bgp** parameter to redistribute static routes.

```
device(config-ospf6-router)# redistribute bgp
```

5. On device2, enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

6. Enter the **ipv6 router ospf** command to enter OSPFv3 router configuration mode and enable OSPFv3 on the device.

```
device(config)# ipv6 router ospf
```

7. Enter the **redistribute** command with the **connected** and **route-map** parameters to redistribute connected routes and specify a route map.

```
device(config-ospf6-router)# redistribute connected route-map rmap1
```

The following example redistributes static and BGP routes routes into OSPFv3 on a device.

```
device# configure terminal
device(config)# ipv6 router ospf
device(config-ospf6-router)# redistribute static
device(config-ospf6-router)# redistribute bgp
```

The following example redistributes connected routes into OSPFv3 on a device and specifies a route map.

```
device# configure terminal
device(config)# ipv6 router ospf
device(config-ospf6-router)# redistribute connected route-map rmap1
```

# Default route origination

When the device is an OSPFv3 Autonomous System Boundary Router (ASBR), you can configure it to automatically generate a default external route into an OSPFv3 routing domain.

By default, a device does not advertise the default route into the OSPFv3 domain. If you want the device to advertise the OSPFv3 default route, you must explicitly enable default route origination. When you enable OSPFv3 default route origination, the device advertises a type 5 default route that is flooded throughout the autonomous system, with the exception of stub areas.

The device advertises the default route into OSPFv3 even if OSPFv3 route redistribution is not enabled, and even if the default route is learned through an IBGP neighbor. The device does not, however, originate the default route if the active default route is learned from an OSPFv3 router in the same domain.

> NOTE
> The device does not advertise the OSPFv3 default route, regardless of other configuration parameters, unless you explicitly enable default route origination.

If default route origination is enabled and you disable it, the default route originated by the device is flushed. Default routes generated by other OSPFv3 devices are not affected. If you re-enable the default route origination, the change takes effect immediately and you do not need to reload the software.

## Configuring default external routes

OSPFv3 default routes can be created and advertised.

1. Enter the **configure terminal** command to access global configuration mode.

   ```
   device# configure terminal
   ```

2. Enter the **ipv6 router ospf** command to enter OSPFv3 router configuration mode and enable OSPFv3 globally.

   ```
   device(config)# ipv6 router ospf
   ```

3. Enter the **default-information-originate** command with the **always**, **metric**, and **metric-type** parameters.

   ```
   device(config-ospf6-router)# default-information-originate always metric 2 metric-type type1
   ```

   A default type 1 external route with a metric of 2 is created and advertised.

The following example creates and advertises a default route with a metric of 2 and a type 1 external route.

```
device# configure terminal
device(config)# ipv6 router ospf
device(config-ospf6-router)# default-information-originate always metric 2 metric-type type1
```

# Disabling and re-enabling OSPFv3 event logging

OSPFv3 event logging, such as neighbor state changes and database overflow conditions, can be disabled and re-enabled.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **ipv6 router ospf** command to enter OSPFv3 router configuration mode and enable OSPFv3 globally.

```
device(config)# ipv6 router ospf
```

3. Enter the **no log-status-change** command to disable the logging of OSPFv3 events.

```
device(config-ospf6-router)# no log-status-change
```

The following example re-enables the logging of OSPFv3 events.

```
device# configure terminal
device(config)# ipv6 router ospf
device(config-ospf6-router)# log-status-change
```

# Filtering OSPFv3 routes

You can filter the routes to be placed in the OSPFv3 route table by configuring distribution lists.

The functionality of OSPFv3 distribution lists is similar to that of OSPFv2 distribution lists. However, unlike OSPFv2 distribution lists, which filter routes based on criteria specified in an Access Control List (ACL), OSPFv3 distribution lists can filter routes using information specified in an IPv6 prefix list or a route map.

## Configuring an OSPFv3 distribution list using an IPv6 prefix list as input

An IPv6 prefix list can be used to filter OSPFv3 routes.

1. Enter the **show ipv6 ospf route** command to verify the OSPFv3 routes.

```
device> show ipv6 ospf route
  Current Route count: 5
    Intra: 3 Inter: 0 External: 2 (Type1 0/Type2 2)
    Equal-cost multi-path: 0
    Destination                  Options   Area            Cost Type2 Cost
    Next Hop Router              Outgoing Interface
*IA 2001:db8:1::/64              --------- 10.0.0.1          0 0
    ::                           ve 10
*E2 2001:db8:2::/64              --------- 0.0.0.0           10 0
    fe80::2e0:52ff:fe00:10       ve 10
*IA 2001:db8:3::/64              V6E---R-- 0.0.0.0           11 0
    fe80::2e0:52ff:fe00:10       ve 10
*IA 2001:db8:4::/64              --------- 0.0.0.0           10 0
    ::                           ve 11
*E2 2001:db8:5::/64              --------- 0.0.0.0           10 0
    fe80::2e0:52ff:fe00:10       ve 10
```

2. Enter the **enable** command to access privileged EXEC mode.

```
device> enable
```

3. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

4. Enter the **ip router-id** command to specify the router ID.

```
device(config)# ip router-id 10.11.12.13
```

5. Enter the **ipv6 prefix-list** command, using the **deny** keyword and specify a name.

```
device(config)# ipv6 prefix-list filterOspfRoutes seq 5 deny 2001:db8:2::/642
```

An IPv6 prefix list called "filterOspfRoutes" that denies route 2001:db8:2::/64 is configured.

6. Enter the **ipv6 prefix-list** command using the **deny** keyword and specify a name. Use the **ge** keyword to specify a prefix length greater than or equal to the *ipv6-prefix/prefix-length* arguments. Use the **le** keyword to specify a prefix length less than or equal to the *ipv6-prefix/prefix-length* arguments.

```
device(config)# ipv6 prefix-list filterOspfRoutes seq 7 permit ::/0 ge 1 le 128
```

7. Enter the **ipv6 router ospf** command to enter OSPFv3 router configuration mode and enable OSPFv3 on the device.

```
device(config)# ipv6 router ospf
```

8. Enter the **distribute-list prefix-list** command, using the **in** keyword and specifying a name to configure a distribution list that applies the filterOspfRoutes prefix list globally.

```
device(config-ospf6-router)# distribute-list prefix-list filterOspfRoutes in
```

9. Enter the **exit** command until you return to user EXEC mode.

```
device(config-ospf6-router)# exit
```

10. Enter the **show ipv6 ospf route** command to verify that route 2001:db8:2::/64 is now omitted from the route table.

```
device> show ipv6 ospf route
  Current Route count: 4
    Intra: 3 Inter: 0 External: 1 (Type1 0/Type2 1)
    Equal-cost multi-path: 0
    Destination                   Options   Area             Cost Type2 Cost
    Next Hop Router               Outgoing Interface
*IA 2001:db8:1::/64               --------- 10.0.0.1            0 0
    ::                            ve 10
*IA 2001:db8:3::/64               V6E---R-- 0.0.0.0             11 0
    fe80::2e0:52ff:fe00:10        ve 10
*IA 2001:db8:4::/64               --------- 0.0.0.0             10 0
    ::                            ve 11
*E2 2001:db8:5::/64               --------- 0.0.0.0             10 0
    fe80::2e0:52ff:fe00:10        ve 10
```

The following example configures an IPv6 prefix list that is used to filter OSPFv3 routes. A distribution list is then configured and route 2001:db8:2::/64 is omitted from the OSPFv3 route table.

```
device> show ipv6 ospf route
device> enable
device# configure terminal
device(config)# ip router-id 10.11.12.13
device(config)# ipv6 prefix-list filterOspfRoutes seq 5 deny 2001:db8:2::/642
device(config)# ipv6 prefix-list filterOspfRoutes seq 7 permit ::/0 ge 1 le 128
device(config)# ipv6 router ospf
device(config-ospf6-router)# distribute-list prefix-list filterOspfRoutes in
device(config-ospf6-router)# exit
device> show ipv6 ospf route
```

# Configuring an OSPFv3 distribution list using a route map as input

A route map that matches internal routes can be used to filter OSPFv3 routes.

1. Enter the **show ipv6 ospf route** command to verify the OSPFv3 routes.

```
device# show ipv6 ospf route
  Current Route count: 5
    Intra: 3 Inter: 0 External: 2 (Type1 0/Type2 2)
    Equal-cost multi-path: 0
    Destination                   Options   Area            Cost Type2 Cost
    Next Hop Router               Outgoing Interface
*IA 2001:db8:1::/64               --------- 10.0.0.1          0 0
    ::                            ve 10
*E2 2001:db8:2::/64               --------- 0.0.0.0           10 0
    fe80::2e0:52ff:fe00:10        ve 10
*IA 2001:db8:3::/64               V6E---R-- 0.0.0.0           11 0
    fe80::2e0:52ff:fe00:10        ve 10
*IA 2001:db8:4::/64               --------- 0.0.0.0           10 0
    ::                            ve 11
*E2 2001:db8:5::/64               --------- 0.0.0.0           10 0
    fe80::2e0:52ff:fe00:10        ve 10
```

2. Enter the **enable** command to access privileged EXEC mode.

```
device> enable
```

3. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

4. Enter the **ip router-id** command to specify the router ID.

```
device(config)# ip router-id 10.11.12.13
```

5. Enter the **route-map** command, using the **permit** keyword and specify a name.

```
device(config)# route-map allowInternalRoutes permit 10
```

A route map called “allowInternalRoutes” that permits a matching pattern is configured.

6. Enter the **match route-type** command, using the **internal** keyword to match internal route types in the route map instance.

```
device(config-routemap allowInternalRoutes)# match route-type internal
```

7. Enter the **exit** command to return to global configuration mode.

```
device(config-routemap allowInternalRoutes)# exit
```

8. Enter the **ipv6 router ospf** command to enter OSPFv3 router configuration mode and enable OSPFv3 on the device.

```
device(config)# ipv6 router ospf
```

9. Enter the **distribute-list route-map** command, using the **in** keyword and specifying a name to create a distribution list a distribution list using the configured route map “allowinternalroutes”.

```
device(config-ospf6-router)# distribute-list route-map allowinternalroutes in
```

10. Enter the **exit** command until you return to user EXEC mode.

```
device(config-ospf6-router)# exit
```

11. Enter the **show ipv6 ospf route** command to verify that the external routes are omitted from the OSPFv3 route table.

```
device> show ipv6 ospf route
  Current Route count: 3
    Intra: 3 Inter: 0 External: 0 (Type1 0/Type2 0)
    Equal-cost multi-path: 0
    Destination                    Options   Area            Cost Type2 Cost
    Next Hop Router                Outgoing Interface
*IA 2001:db8:3001::/64            --------- 10.0.0.1            0 0
    ::                             ve 10
*IA 2001:db8:3015::/64             V6E---R-- 0.0.0.0           11 0
    fe80::2e0:52ff:fe00:10         ve 10
*IA 2001:db8:3020::/64             --------- 0.0.0.0           10 0
    ::                             ve 11
```

The following example configures a route map that is used to filter OSPFv3 routes. A distribution list is then configured and external routes omitted from the OSPFv3 route table.

```
device> show ipv6 ospf route
device> enable
device# configure terminal
device(config)# ip router-id 10.11.12.13
device(config)# route-map allowInternalRoutes permit 10
device(config-routemap allowInternalRoutes)# match route-type internal
device(config-routemap allowInternalRoutes)# exit
device(config)# ipv6 router ospf
device(config-ospf6-router)# distribute-list route-map allowinternalroutes in
device(config-ospf6-router)# exit
device> show ipv6 ospf route
```

# SPF timers

The device uses an SPF delay timer and an SPF hold-time timer to calculate the shortest path for OSPFv3 routes. The values for both timers can be changed.

The device uses the following timers when calculating the shortest path for OSPFv3 routes:

- SPF delay: When the device receives a topology change, it waits before starting a Shortest Path First (SPF) calculation. By default, the device waits 5 seconds. You can configure the SPF delay to a value from 0 through 65535 seconds. If you set the SPF delay to 0 seconds, the device immediately begins the SPF calculation after receiving a topology change.
- SPF hold time: The device waits a specific amount of time between consecutive SPF calculations. By default, it waits 10 seconds. You can configure the SPF hold time to a value from 0 through 65535 seconds. If you set the SPF hold time to 0 seconds, the device does not wait between consecutive SPF calculations.

You can set the SPF delay and hold time to lower values to cause the device to change to alternate paths more quickly if a route fails. Note that lower values for these parameters require more CPU processing time.

You can change one or both of the timers.

NOTE

If you want to change only one of the timers, for example, the SPF delay timer, you must specify the new value for this timer as well as the current value of the SPF hold timer, which you want to retain. The device does not accept only one timer value.

## Modifying SPF timers

The Shortest Path First (SPF) delay and hold time can be modified.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **ipv6 router ospf** command to enter OSPFv3 router configuration mode and enable OSPFv3 globally.

```
device(config)# ipv6 router ospf
```

3. Enter the **timers** command with the **spf** parameter.

```
device(config-ospf6-router)# timers spf 1 5
```

   The SPF delay is changed to 1 second and the SPF hold time is changed to 5 seconds.

The following example changes the SPF delay and hold time.

```
device# configure terminal
device(config)# ipv6 router ospf
device(config-ospf6-router)# timers spf 1 5
```

# OSPFv3 administrative distance

Devices can learn about networks from various protocols and select a route based on the source of the route information. This decision can be influenced if the default administrative distance for OSPFv3 routes is changed. Consequently, the routes to a network may differ depending on the protocol from which the routes were learned.

You can influence the device's decision by changing the default administrative distance for OSPFv3 routes. You can configure a unique administrative distance for each type of OSPFv3 route. For example, you can configure the Extreme device to prefer a static route over an OSPFv3 inter-area route and to prefer OSPFv3 intra-area routes over static routes. The distance you specify influences the choice of routes when the device has multiple routes to the same network from different protocols. The device prefers the route with the lower administrative distance.

You can specify unique default administrative distances for the following OSPFv3 route types:

- Intra-area routes
- Inter-area routes
- External routes

NOTE
The choice of routes within OSPFv3 is not influenced. For example, an OSPFv3 intra-area route is always preferred over an OSPFv3 inter-area route, even if the intra-area route's distance is greater than the inter-area route's distance.

## Configuring administrative distance based on route type

The default administrative distances for intra-area routes, inter-area routes, and external routes can be altered.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **ipv6 router ospf** command to enter OSPFv3 router configuration mode and enable OSPFv3 globally.

```
device(config)# ipv6 router ospf
```

3. Enter the **distance** command with the **intra-area** parameter.

```
device(config-ospf6-router)# distance intra-area 80
```

   The administrative distance for intra-area routes is changed from the default to 80.

4. Enter the **distance** command with the **inter-area** parameter.

```
device(config-ospf6-router)# distance inter-area 90
```

   The administrative distance for inter-area routes is changed from the default to 90.

5. Enter the **distance** command with the **external** parameter.

```
device(config-ospf6-router)# distance external 100
```

   The administrative distance for external routes is changed from the default to 100.

The following example changes the default administrative distances for intra-area routes, inter-area routes, and external routes.

```
device# configure terminal
device(config)# ipv6 router ospf
device(config-ospf6-router)# distance intra-area 80
device(config-ospf6-router)# distance inter-area 90
device(config-ospf6-router)# distance external 100
```

# Changing the reference bandwidth for the cost on OSPFv3 interfaces

The reference bandwidth for OSPFv3 can be altered, resulting in various costs.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **ipv6 router ospf** command to enter OSPFv3 router configuration mode and enable OSPFv3 globally.

```
device(config)# ipv6 router ospf
```

3. Enter the **auto-cost reference-bandwidth** command to change the reference bandwidth.

```
device(config-ospf6-router)# auto-cost reference-bandwidth 500
```

The following example changes the auto-cost reference bandwidth to 500.

```
device# configure terminal
device(config)# ipv6 router ospf
device(config-ospf6-router)# auto-cost reference-bandwidth 500
```

The reference bandwidth specified in this example results in the following costs:

- 10-Mbps port cost = 500/10 = 50
- 100-Mbps port cost = 500/100 = 5
- 1000-Mbps port cost = 500/1000 = 0.5, which is rounded up to 1
- 155-Mbps port cost = 500/155 = 3.23, which is rounded up to 4
- 622-Mbps port cost = 500/622 = 0.80, which is rounded up to 1
- 2488-Mbps port cost = 500/2488 = 0.20, which is rounded up to 1

The costs for 10-Mbps, 100-Mbps, and 155-Mbps ports change as a result of the changed reference bandwidth. Costs for higher-speed interfaces remain the same.

# OSPFv3 LSA refreshes

To prevent a refresh from being performed each time an individual LSA's refresh timer expires, OSPFv3 LSA refreshes are delayed for a specified time interval. This pacing interval can be altered.

The device paces OSPFv3 LSA refreshes by delaying the refreshes for a specified time interval instead of performing a refresh each time an individual LSA's refresh timer expires. The accumulated LSAs constitute a group, which the device refreshes and sends out together in one or more packets.

The pacing interval, which is the interval at which the device refreshes an accumulated group of LSAs, is configurable in a range from 10 through 1800 seconds (30 minutes). The default is 240 seconds (4 minutes). Thus, every four minutes, the device refreshes the group of accumulated LSAs and sends the group together in the same packets.

The pacing interval is inversely proportional to the number of LSAs the device is refreshing and aging. For example, if you have approximately 10,000 LSAs, decreasing the pacing interval enhances performance. If you have a very small database (40 to 100 LSAs), increasing the pacing interval to 10 to 20 minutes may enhance performance only slightly.

## Configuring the OSPFv3 LSA pacing interval

The interval between OSPFv3 LSA refreshes can be modified.

1. Enter the **configure terminal** command to access global configuration mode.

   ```
   device# configure terminal
   ```

2. Enter the **ipv6 router ospf** command to enter OSPFv3 router configuration mode and enable OSPFv3 globally.

   ```
   device(config)# ipv6 router ospf
   ```

3. Enter the **timers** command with the **lsa-group-pacing** parameter.

   ```
   device(config-ospf6-router)# timers lsa-group-pacing 120
   ```

   The OSPFv3 LSA pacing interval is changed to 120 seconds (two minutes).

The following example restores the pacing interval to the default value of 240 seconds (4 minutes).

```
device# configure terminal
device(config)# ipv6 router ospf
device(config-ospf6-router)# no timers lsa-group-pacing
```

# External route summarization

An ASBR can be configured to advertise one external route as an aggregate for all redistributed routes that are covered by a specified IPv6 summary address range.

When you configure a summary address range, the range takes effect immediately. All the imported routes are summarized according to the configured summary address range. Imported routes that have already been advertised and that fall within the range are flushed out of the autonomous system and a single route corresponding to the range is advertised.

If a route that falls within a configured summary address range is imported by the device, no action is taken if the device has already advertised the aggregate route; otherwise, the device advertises the aggregate route. If an imported route that falls within a configured summary address range is removed by the device, no action is taken if there are other imported routes that fall within the same summary address range; otherwise, the aggregate route is flushed.

You can configure up to 32 summary address ranges.

The device sets the forwarding address of the aggregate route to 0 and sets the tag to 0. If you delete a summary address range, the advertised aggregate route is flushed and all imported routes that fall within the range are advertised individually. If an external link-state database (LSDB) overflow condition occurs, all aggregate routes and other external routes are flushed out of the autonomous system. When the device exits the external LSDB overflow condition, all the imported routes are summarized according to the configured address ranges.

NOTE
If you use redistribution filters in addition to summary address ranges, the device applies the redistribution filters to routes first, and then applies them to the summary address ranges.

NOTE
If you disable redistribution, all the aggregate routes are flushed, along with other imported routes.

NOTE
Only imported, type 5 external LSA routes are affected. A single type 5 LSA is generated and flooded throughout the autonomous system for multiple external routes.

# OSPFv3 over VRF

OSPFv3 can run over multiple Virtual Routing and Forwarding (VRF) instances. OSPFv3 maintains multiple instances of the routing protocol to exchange route information among various VRF instances. A multi-VRF-capable router maps an input interface to a unique VRF, based on user configuration. These input interfaces can be physical or a virtual interface. By default, all input interfaces are attached to the default VRF instance. All OSPFv3 commands are available over default and nondefault VRF instances.

Multi-VRF for OSPF (also known as VRF-Lite for OSPF) provides a reliable mechanism for trusted VPNs to be built over a shared infrastructure. The ability to maintain multiple virtual routing or forwarding tables allows overlapping private IP addresses to be maintained across VPNs.

# Enabling OSPFv3 in a non-default VRF

When OSPFv3 is enabled in a non-default VRF instance, the device enters OSPFv3 router VRF configuration mode. Several commands can then be accessed that allow the configuration of OSPFv3.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **vrf** command and specify a name to enter Virtual Routing and Forwarding (VRF) configuration mode and create a non-default VRF instance.

```
device(config)# vrf green
```

3. Enter the **rd** command, assigning an administrative number and arbitrary number the route, to distinguish a route for VRF green.

```
device(config-vrf-green)# rd 100:200
```

4. Enter the **ip router-id** command to specify the router ID.

```
device(config-vrf-green)# ip router-id 10.11.12.14
```

5. Enter the **address-family ipv6** command to enter IPv6 address-family configuration mode.

```
device(config-vrf-green)# address-family ipv6
```

6. Enter the **exit** command until you return to global configuration mode.

```
device(config-vrf-green-ipv6)# exit
```

7. Enter the **ipv6 router ospf** command and specify a VRF name to enter OSPFv3 router VRF configuration mode and enable OSPFv3 on a non-default VRF.

```
device(config)# ipv6 router ospf vrf green
```

The following example enables OSPFv3 in a non-default VRF.

```
device# configure terminal
device(config)# vrf green
device(config-vrf-green)# rd 100:200
device(config-vrf-green)# ip router-id 10.11.12.14
device(config-vrf-green)# address-family ipv6
device(config-vrf-green-ipv6)#
device(config-vrf-green-ipv6)# exit
device(config)# ipv6 router ospf vrf green
device(config-ospf6-router-vrf-green)#
```

# Assigning OSPFv3 areas in a non-default VRF

Areas can be assigned as OSPFv3 areas in a non-default VRF.

Enable IPv6 on each interface on which you plan to enable OSPFv3. You enable IPv6 on an interface by configuring an IP address or explicitly enabling IPv6 on that interface.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **vrf** command and specify a name to enter Virtual Routing and Forwarding (VRF) configuration mode and create a non-default VRF instance.

```
device(config)# vrf red
```

3. Enter the **rd** command, assigning an administrative number and arbitrary number the route, to distinguish a route for VRF green.

```
device(config-vrf-red)# rd 100:200
```

4. Enter the **ip router-id** command to specify the router ID.

```
device(config-vrf-red)# ip router-id 10.11.12.14
```

5. Enter the **address-family ipv6** command to enter IPv6 address-family configuration mode.

```
device(config-vrf-red)# address-family ipv6
```

6. Enter the **exit** command until you return to global configuration mode.

```
device(config-vrf-red-ipv6)# exit
```

7. Enter the **ipv6 router ospf** command and specify a VRF name to enter OSPFv3 configuration mode and enable OSPFv3 in a non-default VRF.

```
device(config)# ipv6 router ospf vrf red
```

8. Enter the **area** command to define an OSPFv3 area ID.

```
device(config-ospf6-router-vrf-red)# area 0
```

9. Enter the **area** command to define a second OSPFv3 area ID.

```
device(config-ospf6-router-vrf-red)# area 10.1.1.1
```

The following example assigns an OSPFv3 ID to two areas in a non-default VRF instance. One of the areas is assigned by decimal number. The second area is assigned by IP address.

```
device# configure terminal
device(config)# vrf red
device(config-vrf-red)# rd 100:200
device(config-vrf-red)# ip router-id 10.11.12.13
device(config-vrf-red)# address-family ipv6
device(config-vrf-red-ipv6)#
device(config-vrf-red-ipv6)# exit
device(config)# ipv6 router ospf vrf red
device(config-ospf6-router-vrf-red)# area 0
device(config-ospf6-router-vrf-red)# area 10.1.1.1
```

# Setting all OSPFv3 interfaces to the passive state

All OSPFv3 interfaces can be set as passive, causing them to drop all OSPFv3 control packets.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **ipv6 router ospf** command to enter OSPFv3 router configuration mode and enable OSPFv3 globally.

```
device(config)# ipv6 router ospf
```

3. Enter the **default-passive-interface** command to mark all interfaces passive by default.

```
device(config-ospf6-router)# default-passive-interface
```

The following example sets all OSPFv3 interfaces as passive, causing them to drop all the OSPFv3 control packets.

```
device# configure terminal
device(config)# ipv6 router ospf
device(config-ospf6-router)# default-passive-interface
```

# OSPFv3 graceful restart helper

The OSPFv3 graceful restart (GR) helper provides a device with the capability to participate in a graceful restart in helper mode so that it assists a neighboring routing device that is performing a graceful restart.

When OSPFv3 GR helper is enabled on a device, the device enters helper mode upon receipt of a grace-LSA where the neighbor state is full. By default, the helper capability is enabled when you start OSPFv3, even if graceful restart is not supported.

## Disabling OSPFv3 graceful restart helper

The OSPFv3 graceful restart (GR) helper is enabled by default, and can be disabled on a routing device.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **ipv6 router ospf** command to enter OSPFv3 router configuration mode and enable OSPFv3 globally.

```
device(config)# ipv6 router ospf
```

3. Enter the **no graceful-restart helper** command with the **strict-lsa-checking** to disable the GR helper with strict link-state advertisement (LSA) checking.

```
device(config-ospf6-router)# no graceful-restart helper strict-lsa-checking
```

The following example disables the GR helper with strict link-state advertisement (LSA) checking.

```
device# configure terminal
device(config)# ipv6 router ospf
device(config-ospf6-router)# no graceful-restart helper strict-lsa-checking
```

## Re-enabling OSPFv3 graceful restart helper

If the OSPFv3 graceful restart (GR) helper has been disabled on a routing device, it can be re-enabled. GR helper mode can also be enabled with strict link-state advertisement (LSA) checking.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **ipv6 router ospf** command to enter OSPFv3 router configuration mode and enable OSPFv3 globally.

```
device(config)# ipv6 router ospf
```

3. Enter the **graceful-restart helper** command and specify the **strict-lsa-checking** parameter to re-enable the GR helper with strict LSA checking.

```
device(config-ospf6-router)# graceful-restart helper strict-lsa-checking
```

The following example re-enables the GR helper with strict LSA checking.

```
device# configure terminal
device(config)# ipv6 router ospf
device(config-ospf6-router)# graceful-restart helper strict-lsa-checking
```

# OSPFv3 non-stop routing

OSPFv3 can continue operation without interruption during hitless failover when the NSR feature is enabled.

During graceful restart (GR), the restarting neighbors must help build routing information during a failover. However, the GR helper may not be supported by all devices in a network. Non-stop routing (NSR) eliminates this dependency.

NSR does not require support from neighboring devices to perform hitless failover, and OSPF can continue operation without interruption.

> NOTE
> NSR does not support IPv6-over-IPv4 tunnels and virtual links, so traffic loss is expected while performing hitless failover.

## Enabling OSPFv3 NSR

OSPFv3 non-stop routing (NSR) can be re-enabled if it has been disabled. The following task re-enables NSR for OSPFv3.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **ipv6 router ospf** command to enter OSPFv3 router configuration mode and enable OSPFv3 globally.

```
device(config)# ipv6 router ospf
```

3. Enter the **graceful restart** command to re-enable GR on the device.

```
device(config-ospf6-router)# nonstop-routing
```

The following example re-enables NSR for OSPFv3.

```
device# configure terminal
device(config)# ipv6 router ospf
device(config-ospf6-router)# nonstop-routing
```

# OSPFv3 max-metric router LSA

OSPFv3 can be configured to advertise its locally generated router LSAs with a maximum metric to direct transit traffic away from the device, while still routing for directly connected networks.

By advertising the maximum metric, the device does not attract transit traffic. A device which does not handle transit traffic, and only forwards packets destined for its directly connected links, is known as a stub router. In OSPFv3 networks, a device could be placed in a stub router role by advertising large metrics for its connected links, so that the cost of a path through the device becomes larger than that of an alternative path.

You can configure OSPFv3 max-metric router LSA in either startup or non-startup mode. Configuring max-metric on startup may be helpful on ASBRs where protocols such as BGP converge after OSPF converges. Configuring max-metric on non-startup may be helpful in database overflow scenarios.

NOTE
The **on-startup** configuration does not apply to NSR restarts.

## Configuring the OSPFv3 max-metric router LSA

By configuring the OSPFv3 max-metric router LSA feature you can enable OSPFv3 to advertise its locally generated router LSAs with a maximum metric.

NOTE
You can configure OSPFv3 max-metric router LSA in either startup or non-startup mode. Configuring max-metric with non-startup mode, it is only applied once and is not persistent across reloads, or after the **clear ip ospf all** command is issued.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **ip router-id** command to specify the router ID.

```
device(config)# ip router-id 10.11.12.13
```

3. Enter the **ipv6 router ospf** command to enter OSPFv3 router configuration mode and enable OSPFv3 on the device.

```
device(config)# ipv6 router ospf
```

4. Enter the **max-metric router-lsa** command with the **external-lsa** keyword and specify a value to configure the maximum metric value .

```
device(config-ospf6-router)# max-metric router-lsa external-lsa 1500
```

5. Enter the **max-metric router-lsa** command with the **on-startup** keyword and specify a value to specify a period of time to advertise a maximum metric after a restart before advertising with a normal metric.

```
device(config-ospf6-router)# max-metric router-lsa on-startup 85
```

The following example configures an OSPFv3 device to advertise a maximum metric and sets the maximum metric value for external LSAs to 1500, and configures the device to advertise a maximum metric for 85 seconds after a restart before advertising with a normal metric.

```
device# configure terminal
device(config)# ip router-id 10.11.12.13
device(config)# ipv6 router ospf
device(config-ospf6-router)# max-metric router-lsa external-lsa 1500
device(config-ospf6-router)# max-metric router-lsa on-startup 85
```

# IPsec for OSPFv3

IP Security (IPsec) secures OSPFv3 communications by authenticating and encrypting each IP packet of a communication session.

IPsec provides security features such as data integrity, replay protection, and message confidentiality. You can use IPsec to secure specific OSPFv3 areas and interfaces and protect OSPFv3 virtual links.

The Encapsulating Security Payload (ESP) protocol authenticates routing information between peers. ESP can provide message confidentiality, connectionless data integrity, and optional replay protection. ESP has both a header and a trailer. The authentication data of ESP cannot protect the outer IP header, only the payload that is being encrypted.

IPsec is available for OSPFv3 traffic only and only for packets that are "for-us". A for-us packet is addressed to one of the IPv6 addresses on the device or to an IPv6 multicast address. Packets that are only forwarded by the line card do not receive IPsec scrutiny.

The devices support the following components of IPsec for IPv6-addressed packets:

- Authentication through ESP in transport mode
- Hashed Message Authentication Code-Secure Hash Algorithm 1 (HMAC-SHA-1) as the authentication algorithm
- Security parameter index (SPI)
- Manual configuration of keys
- Configurable rollover timer

IPsec can be enabled on the following logical entities:

- Interface
- Area
- Virtual link

IPsec is based on security associations (SAs). With respect to traffic classes, this implementation of IPsec uses a single security association between the source and destination to support all traffic classes and does not differentiate between the different classes of traffic that the DSCP bits define.

IPsec on a virtual link is a global configuration. Interface and area IPsec configurations are more granular.

Among the entities that can have IPsec protection, the interfaces and areas can overlap. The interface IPsec configuration takes precedence over the area IPsec configuration when an area and an interface within that area use IPsec. Therefore, if you configure IPsec for an interface and an area configuration also exists that includes this interface, the interface's IPsec configuration is used by that interface. However, if you disable IPsec on an interface, IPsec is disabled on the interface even if the interface has its own specific authentication.

For IPsec, the system generates two types of databases. The Security Association Database (SAD) contains a security association for each interface or one global database for a virtual link. Even if IPsec is configured for an area, each interface that uses the area's IPsec still has its own security association in the SAD. Each SA in the SAD is a generated entry that is based on your specifications of an authentication protocol (for example, ESP), destination address, and a security parameter index (SPI). The SPI number is user-specified according to the network plan. Consideration for the SPI values to specify must apply to the whole network.

The system-generated security policy databases (SPDs) contain the security policies against which the system checks the for-us packets. For each for-us packet that has an ESP header, the applicable security policy in the security policy database (SPD) is checked to see if this packet complies with the policy. The IPsec task drops the non-compliant packets. Compliant packets continue on to the OSPFv3 task.

# IPsec for OSPFv3 configuration

IPsec authentication can be enabled on both default and nondefault VRFs. IPsec authentication is disabled by default.

The following IPsec parameters are configurable:

- ESP protocol
- Authentication
- Hashed Message Authentication Code-Secure Hash Algorithm 1 (HMAC-SHA-1) authentication algorithm
- Security parameter index (SPI)
- A 40-character key using hexadecimal characters
- An option for not encrypting the keyword when it appears in **show** command output
- Key rollover timer
- Specifying the key add remove timer

# IPsec for OSPFv3 considerations

IPsec generates security associations and security policies based on certain user-specified parameters. Refer to the *NetIron Command Reference* for more information on user-specified parameters.

- The system creates a security association for each interface or virtual link based on the values specified by the user.
- The system creates a security policy database for each interface or virtual link based on the values specified by the user.
- You can configure the same SPI and key on multiple interfaces and areas, but they still have unique IPsec configurations because the SA and policies are added to each separate security policy database (SPD) that is associated with a particular interface. If you configure an SA with the same SPI in multiple places, the rest of the parameters associated with the SA—such as key, cryptographic algorithm, security protocol, and so on—must match. If the system detects a mismatch, it displays an error message.
- IPsec authentication for OSPFv3 requires the use of multiple SPDs, one for each interface. A virtual link has a separate, global SPD. The authentication configuration on a virtual link must be different from the authentication configuration for an area or interface, as required by RFC4 552. The interface number is used to generate a non-zero security policy database identifier (SPDID), but for the global SPD for a virtual link, the system-generated SPDID is always zero. As a hypothetical example, the SPD for interface eth 1/1 might have the system-generated SPDID of 1, and so on.
- If you change an existing key, you must also specify a different SPI value. For example, in an interface context where you intend to change a key, you must enter a different SPI value—which occurs before the key parameter on the command line—before you enter the new key.
- The old key is active for twice the current configured key rollover interval for the inbound direction. In the outbound direction, the old key remains active for a duration equal to the key rollover interval. If the key rollover interval is set to 0, the new key immediately takes effect for both directions.

# Configuring IPsec on an OSPFv3 area

IPsec can be configured to secure communications on an OSPFv3 area.

Currently certain keyword parameters must be entered though only one keyword choice is possible for that parameter. For example, the only authentication algorithm is HMAC-SHA1-96, but you must nevertheless enter the **sha1** keyword for this algorithm. Also, although ESP is currently the only authentication protocol, you must enter the **esp** keyword.

NOTE
When IPsec is configured for an area, the security policy is applied to all the interfaces in the area.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **ip router-id** command to specify the router ID.

```
device(config)# ip router-id 10.11.12.13
```

3. Enter the **ipv6 router ospf** command to enter OSPFv3 configuration mode and enable OSPFv3 on the device.

```
device(config)# ipv6 router ospf
```

4. Enter **area authentication ipsec spi** *spi* **esp sha1**, specifying an area, and enter a 40-character hexadecimal key.

```
device(config-ospf6-router)# area 0 authentication ipsec spi 600 esp sha1
abcef12345678901234fedcba098765432109876
```

IPsec is configured in OSPv3 area 0 with a security parameter index (SPI) value of 600, and Hashed Message Authentication Code (HMAC) Secure Hash Algorithm 1 (SHA-1) authentication is enabled.

The following example enables HMAC SHA-1 authentication for the OSPFv3 area, setting an SPI value of 600.

```
device# configure terminal
device(config)# ip router-id 10.11.12.13
device(config)# ipv6 router ospf
device(config-ospf6-router)# area 0 authentication ipsec spi 600 esp sha1
abcef12345678901234fedcba098765432109876
```

# Configuring IPsec on an OSPFv3 interface

IPsec can be configured to secure communications on an OSPFv3 interface.

For IPsec to work, the IPsec configuration must be the same on all the routers to which an interface connects.

Currently certain keyword parameters must be entered though only one keyword choice is possible for that parameter. For example, the only authentication algorithm is HMAC-SHA1-96, but you must nevertheless enter the **sha1** keyword for this algorithm. Also, although ESP is currently the only authentication protocol, you must enter the **esp** keyword.

NOTE
Ensure that OSPFv3 areas are assigned. All device interfaces must be assigned to one of the defined areas on an OSPFv3 router. When an interface is assigned to an area, all corresponding subnets on that interface are automatically included in the assignment.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **interface** command and specify an interface.

```
device(config)# interface ethernet 1/1
```

3. Enter the **ipv6 ospf area** command to assign a specified area to the interface.

```
device(config-if-e1000/1/1)# ipv6 ospf area 0
```

4. Enter **ipv6 ospf authentication ipsec spi** *value* **esp sha1** and specify a 40-character hexadecimal key.

```
device(config-if-e1000/1/1)# ipv6 ospf authentication ipsec spi 512 esp sha1
abcef12345678901234fedcba098765432109876
```

   IPsec is configured on the specified interface with a security parameter index (SPI) value of 512, and the Encapsulating Security Payload (ESP) protocol is selected. Secure Hash Algorithm 1 (SHA-1) authentication is enabled.

The following example enables ESP and SHA-1 on a specified OSPFv3 Ethernet interface.

```
device# configure terminal
device(config)# interface ethernet 1/1
device(config-if-e1000/1/1)# ipv6 ospf area 0
device(config-if-e1000/1/1)# ipv6 ospf authentication ipsec spi 512 esp sha1
abcef12345678901234fedcba098765432109876
```

# Configuring IPsec on OSPFv3 virtual links

IP Security (IPsec) can be configured for virtual links.

An OSPFv3 virtual link must be configured.

Currently certain keyword parameters must be entered though only one keyword choice is possible for that parameter. For example, the only authentication algorithm is HMAC-SHA1-96, but you must nevertheless enter the **sha1** keyword for this algorithm. Also, although ESP is currently the only authentication protocol, you must enter the **esp** keyword.

The virtual link IPsec security associations (SAs) and policies are added to all interfaces of the transit area for the outbound direction. For the inbound direction, IPsec SAs and policies for virtual links are added to the global database.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **ip router-id** command to specify the router ID.

```
device(config)# ip router-id 10.1.1.1
```

3. Enter the **ipv6 router ospf** command to enter OSPFv3 configuration mode and enable OSPFv3 on the device.

```
device(config)# ipv6 router ospf
```

4. Enter **area virtual-link authentication ipsec spi** *value* **esp sha1 no-encrypt** *key*, specifying an area address and the ID of the OSPFv3 device at the remote end of the virtual link..

```
device(config-ospf6-router)# area 1 virtual-link 10.1.1.1 authentication ipsec spi 512 esp sha1 no-
encrypt 113456789022345678901234567890123456789O
```

   IPsec is configured on the specified virtual link in OSPF area 1.The device ID associated with the virtual link neighbor is 10.1.1.1, the SPI value is 512, and the Encapsulating Security Payload (ESP) protocol is selected. Secure Hash Algorithm 1 (SHA-1) authentication is enabled. The 40-character key is not encrypted in **show** command displays.

The following example configures IPsec on an OSPFv3 area.

```
device# configure terminal
device(config)# ip router-id 10.1.1.1
device(config)# ipv6 router ospf
device(config-ospf6-router)# area 1 virtual-link 10.1.1.1 authentication ipsec spi 512 esp sha1 no-encrypt
1134567890223456789012345678901234567890
```

# Specifying the key rollover and key add-remove timers

The key rollover timer can be configured so that rekeying takes place on all the nodes at the same time and the security parameters are consistent across all the nodes. The timing of the authentication key add-remove interval can also be altered.

1. Enter the **configure terminal** command to access global configuration mode.

   ```
   device# configure terminal
   ```

2. Enter the **ip router-id** command to specify the router ID.

   ```
   device(config)# ip router-id 10.11.12.13
   ```

3. Enter the **ipv6 router ospf** command to enter OSPFv3 router configuration mode and enable OSPFv3 on the device.

   ```
   device(config)# ipv6 router ospf
   ```

4. Enter the **key-add-remove-interval** command and specify the desired interval to set the timing of the authentication key add-remove interval.

   ```
   device(config-ospf6-router)# key-add-remove-interval 240

   device(config-ipv6-router-ospf-vrf-default-vrf)# key-add-remove-interval 240
   ```

5. Enter the **key-rollover-interval** command and specify the desired interval to set the timing of the configuration changeover.

   ```
   device(config-ospf6-router)# key-rollover-interval 240
   ```

The following example sets the key add-remove interval to 240 seconds (4 minutes) and sets the timing of the configuration changeover to 240 seconds (4 minutes).

```
device# configure terminal
device(config)# ip router-id 10.11.12.13
device(config)# ipv6 router ospf
device(config-ospf6-router)# key-add-remove-interval 240
device(config-ospf6-router)# key-rollover-interval 240
```

# Clearing IPsec statistics

Statistics related to IP security (IPsec) can be cleared using the **clear ipsec statistics** command.

Use the **show ipsec statistics** command to display the IPsec statistics. After using the **clear ipsec statistics** command to clear the IPsec statistics, re-enter the **show ipsec statistics** command to verify the IPsec statistics have been cleared. **The clear ipsec statistics** command resets the IPsec packet statistics and IPsec error statistics counters to zero.

1. Enter the **exit** command as necessary to access user EXEC mode

   ```
   device(config)# exit
   ```

2. Enter the **show ipsec statistics** command to display statistics related to IPsec.

```
device# show ipsec statistics
                    IPSecurity Statistics
secEspCurrentInboundSAs 1          ipsecEspTotalInboundSAs:  2
secEspCurrentOutboundSA 1          ipsecEspTotalOutboundSAs: 2
                  IPSecurity Packet Statistics
secEspTotalInPkts:      20         ipsecEspTotalInPktsDrop:  0
secEspTotalOutPkts:     84
                  IPSecurity Error Statistics
secAuthenticationErrors 0
secReplayErrors:        0          ipsecPolicyErrors:        13
secOtherReceiveErrors:  0          ipsecSendErrors:          0
secUnknownSpiErrors:    0
```

3. Enter the **clear ipsec statistics** command to clear statistics related to IPsec from the configuration.

```
device# clear ipsec statistics
```

4. Enter the **show ipsec statistics** command to verify that statistics related to IPsec have been cleared from the configuration.

```
device# show ipsec statistics
                     IPSecurity Statistics
ipsecEspCurrentInboundSAs 0          ipsecEspTotalInboundSAs:  0
ipsecEspCurrentOutboundSA 0          ipsecEspTotalOutboundSAs: 0
                   IPSecurity Packet Statistics
ipsecEspTotalInPkts:      0          ipsecEspTotalInPktsDrop:  0
ipsecEspTotalOutPkts:     0
                   IPSecurity Error Statistics
ipsecAuthenticationErrors 0
ipsecReplayErrors:        0          ipsecPolicyErrors:        0
ipsecOtherReceiveErrors:  0          ipsecSendErrors:          0
ipsecUnknownSpiErrors:    0
device#
```

The counters holding IPsec packet statistics and IPsec error statistics are reset to 0.

The following example clears IPsec statistics and verifies that the IPsec statistics have been cleared.

```
device(config-ospf6-router)# exit
device(config)# exit
device# show ipsec statistics
device# clear ipsec statistics
device# show ipsec statistics
```

# Displaying OSPFv3 results

The **show ipv6 ospf** command and its variations can be used to display information about OSPFv3 configurations.

Use one or more of the following commands to verify OSPFv3 information. Using the **show ipv6 ospf** command is optional, and the variations of the command can be entered in any order.

1. Enter the **exit** command as necessary to access user EXEC mode.

```
device# exit
```

2. Enter the **show ipv6 ospf** command to display general OSPFv3 information.

```
device> show ipv6 ospf

OSPFv3 Process number 0 with Router ID 0x10010101(10.1.1.1)
    Running 0 days 0 hours 1 minutes 53 seconds
    Number of AS scoped LSAs is 3
    Sum of AS scoped LSAs Checksum is fabdd4de
 External LSA Limit is 250000
 Database Overflow Interval is 10
 Database Overflow State is NOT OVERFLOWED
 Route calculation executed 0 times
 Pending outgoing LSA count 0
 Authentication key rollover interval 30 seconds
    Authentication key add/remove interval 0 seconds
 Number of areas in this router is 3
 Router is operating as ABR
 Router is operating as ASBR, Redistribute: CONNECTED
 High Priority Message Queue Full count: 0
 BFD is disabled
```

3. The following example of the **show ipv6 ospf area** command shows detailed output for assigned OSPFv3 Area 0.

```
device> show ipv6 ospf area 0

Area 0:
  Interface attached to this area: loopback 2 ethe 3/2 tunnel 2
        Number of Area scoped LSAs is 6
        Statistics of Area 0:
            SPF algorithm executed 16 times
            SPF last updated: 335256 sec ago
            Current SPF node count: 3
                Router: 2 Network: 1
                Maximum of Hop count to nodes: 2
```

4. The following example of the **show ipv6 ospf interface brief** command shows limited OSPFv3 interface information.

```
device> show ipv6 ospf interface brief

Interface     Area     Status Type Cost  State    Nbrs(F/C)
eth 1/1       1        up      BCST  1     BDR      0/1
eth 2/1       1        up      BCST  1     DR       0/0
loopback 1    1        up      BCST  1     Loopback 0/0
```

5. The following example of the **show ipv6 ospf neighbor** command shows OSPFv3 neighbor information for the device.

```
device> show ipv6 ospf neighbor

RouterID     Pri State    DR              BDR         Interface [State]
10.1.1.1     1 Full       10.223.223.223  10.1.1.1    ethe 3/2  [DR]
```

6. The following example of the **show ipv6 ospf virtual-neighbor** command shows information about an OSPFv3 virtual neighbor.

```
device> show ipv6 ospf virtual-neighbor

Index Router ID        Address                       State     Interface
1     10.14.14.14      2001:db8:44:44::4             Full      eth 1/8
                Option: 00-00-00    QCount: 0    Timer: 408
2     10.14.14.14      2001:db8:44:44::4             Full      tunnel 256
                Option: 00-00-00    QCount: 0    Timer: 43
```

7. The following example of the **show ipv6 ospf database** command shows information about different OSPFv3 LSAs.

```
device> show ipv6 ospf database

LSA Key - Rtr:Router Net:Network Inap:InterPrefix Inar:InterRouter
          Extn:ASExternal Grp:GroupMembership Typ7:Type7 Link:Link
          Iap:IntraPrefix Grc:Grace
Area ID     Type LSID     Adv Rtr      Seq(Hex)  Age  Cksum Len   Sync
0           Iap  0        10.1.1.1     80000001  3    343d  52    Yes
0           Iap  0        10.2.2.2     80000001  8    c61d  58    No
```

8. The following example of the **show ipv6 ospf routes** command shows output for OSPFv3 routes.

```
device> show ipv6 ospf routes

Current Route count: 4
    Intra: 4 Inter: 0 External: 0 (Type1 0/Type2 0)
    Equal-cost multi-path: 0
    OSPF Type: IA- Intra, OA - Inter, E1 - External Type1, E2 - External Type2
   Destination               Cost     E2Cost    Tag         Flags    Dis
IA 2001:db8:200:1::1/128     0        0         0           00000003 110
   Next_Hop_Router           Outgoing_Interface Adv_Router
   ::                        loopback 1         10.1.2.1
   Destination               Cost     E2Cost    Tag         Flags    Dis
IA 2001:db8:300:1::1/128     0        0         0           00000003 110
   Next_Hop_Router           Outgoing_Interface Adv_Router
   ::                        loopback 2         10.1.2.1
   Destination               Cost     E2Cost    Tag         Flags    Dis
IA 2001:db8:400:1::1/128     0        0         0           00000003 110
   Next_Hop_Router           Outgoing_Interface Adv_Router
   ::                        loopback 1         10.1.2.1
   Destination               Cost      E2Cost   Tag         Flags    Dis
IA 2001:db8:500:1::1/128     1         0        0           00000003 110
   Next_Hop_Router           Outgoing_Interface Adv_Router
   ::                        loopback 2         10.1.2.1
```

9. The following example of the **show ipv6 ospf database** command with the **tree** shows information about the SPF trees.

```
device> show ipv6 ospf spf tree

 SPF tree for Area 0
+- 10.223.223.223 cost 0
+- 10.223.223.223:88 cost 1
+- 10.1.1.1:0 cost 1
```

10. The following example of the **show ipv6 ospf database** command with the **table** shows information about the SPF table.

```
device> show ipv6 ospf spf table

SPF table for Area 0
 Destination         Bits Options  Cost Nexthop                   Interface
R 10.1.1.1           ---- V6E---R-   1  fe80::2e0:52ff:fe91:bb37  ethe 3/2
N 10.223.223.223[88] ---- V6E---R-   1  ::                        ethe 3/2
```

11. The following example of the **show ipv6 ospf redistribute route** command shows information about routes that the device has redistributed into OSPFv3.

```
device> show ipv6 ospf redistribute route

Id     Prefix                 Protocol  Metric Type  Metric
1      2001:db8::/32          Static    Type-2       1
2      2001:db8:1234::/48     Static    Type-2       0
```

12. The following example of the **show ipv6 ospf routes** command shows information about a specified OSPFv3 route.

```
device> show ipv6 ospf routes 2000::
Destination                 Cost      E2Cost    Tag       Flags    Dis
IA 2000::/64                1         0         0         00000003 110
   Next_Hop_Router           Outgoing_Interface Adv_Router
   ::                           eth 1/1           10.1.1.1
```

# RIP



# RIP overview

Routing Information Protocol (RIP) is an IP route exchange protocol that uses a distance vector (a number representing distance) to measure the cost of a given route. The cost is a distance vector because the cost often is equivalent to the number of router hops between the Extreme device and the destination network.

A device can receive multiple paths to a destination. The software evaluates the paths, selects the best path, and saves the path in the IP route table as the route to the destination. Typically, the best path is the path with the fewest hops. A hop is another router through which packets must travel to reach the destination. If a RIP update is received from another router that contains a path with fewer hops than the path stored in the Extreme device route table, the older route is replaced with the newer one. The new path is then included in the updates sent to other RIP routers, including Extreme devices.

RIP routers, including Extreme devices, also can modify a route cost, generally by adding to it, to bias the selection of a route for a given destination. In this case, the actual number of router hops may be the same, but the route has an administratively higher cost and is thus less likely to be used than other, lower-cost routes.

A RIP route can have a maximum cost of 15. Any destination with a higher cost is considered unreachable. Although limiting to larger networks, the low maximum hop count prevents endless loops in the network.

Extreme devices support the following RIP versions:

- Version 1 (v1)
- Version 2 (v2, the default)
- V1 compatible with v2

# RIP parameters and defaults

You can configure global RIP parameters for the protocol and interface RIP parameters on those interfaces that send and receive RIP information.

## RIP global parameters

**TABLE 53** RIP global parameters

| Parameter | Description | Default |
|---|---|---|
| RIP state | The global state of the protocol. | Disabled |

**TABLE 53** RIP global parameters (continued)

| Parameter | Description | Default |
|---|---|---|
| | **NOTE**<br>You also must enable the protocol on individual interfaces. Globally enabling the protocol does not allow interfaces to send and receive RIP information. | |
| Administrative distance | The administrative distance is a numeric value assigned to each type of route on the device.<br>When the device is selecting from among multiple routes (sometimes of different origins) to the same destination, the device compares the administrative distances of the routes and selects the route with the lowest administrative distance. | 120 |
| Redistribution | RIP can redistribute routes from other routing protocols such as OSPF and BGP4 into RIP. A redistributed route is one that a router learns through another protocol, and then distributes into RIP. | Disabled |
| Redistribution metric | RIP assigns a RIP metric (cost) to each external route redistributed from another routing protocol into RIP. | 1 |
| Update Interval | How often the router sends route updates to its RIP neighbors. | 30 seconds |
| Learning default routes | The device can learn default routes from its RIP neighbors.<br>**NOTE**<br>You also can enable or disable this parameter on an individual interface basis. | Disabled |
| Advertising and learning with specific neighbors | The device learns and advertises RIP routes with all its neighbors by default. You can prevent the device from advertising routes to specific neighbors or learning routes from specific neighbors. | Learning and advertising permitted for all neighbors |

# RIP interface parameters

**TABLE 54** RIP interface parameters

| Parameter | Description | Default |
|---|---|---|
| RIP state and version | The state of the protocol and the version that is supported on the interface. The version can be one of the following:<br>• Version 1 only<br>• Version 2 only<br>• Version 1, but also compatible with version 2 | Disabled |

**TABLE 54** RIP interface parameters (continued)

| Parameter | Description | Default |
|---|---|---|
| | **NOTE**<br>You also must enable RIP globally. | |
| Metric | A numeric cost the device adds to RIP routes learned on the interface. This parameter applies only to RIP routes. | 1 |
| Learning default routes | Locally overrides the global setting. | Disabled |
| Loop prevention | The method a device uses to prevent routing loops caused by advertising a route on the same interface as the one on which the device learned the route.<br>• Split horizon - The device does not advertise a route on the same interface as the one on which the device learned the route.<br>• Poison reverse - The device assigns a cost of 16 ("infinite" or "unreachable") to a route before advertising it on the same interface as the one on which the device learned the route.<br>**NOTE**<br>Enabling poison reverse disables split horizon on the interface. | Split horizon |
| Advertising and learning specific routes | You can control the routes that a device learns or advertises. | The device learns and advertises all RIP routes on all interfaces. |

# Configuring RIP parameters

## Enabling RIP

RIP is disabled by default. To enable RIP, you must enable it globally and also on individual interfaces on which you want to advertise RIP. Globally enabling the protocol does not enable it on individual interfaces. When you enable RIP on a port, you also must specify the version (version 1 only, version 2 only, or version 1 compatible with version 2).

To enable RIP globally, enter the **router rip** command.

```
device(config)# router rip
```

**Syntax: [no] router rip**

After globally enabling the protocol, you must enable it on individual interfaces. You can enable the protocol on physical interfaces as well as virtual routing interfaces. To enable RIP on an interface, enter commands such as the following.

```
device(config)# interface ethernet 1/1
device(config-if-e1000-1/1)# ip rip v1-only
```

**Syntax: [no] ip rip {v1-only | v1-compatible-v2 | v2-only}**

# Configuring route costs

By default, the device port increases the cost of a RIP route that is learned on the port. The device increases the cost by adding one to the route metric before storing the route.

You can change the amount that an individual port adds to the metric of RIP routes learned on the port.

To increase the metric for learned routes, enter the **ip rip metric-offset** command.

```
device(config-if-e1000-1/1)# ip rip metric-offset 5 in
```

In the above example, the **ip rip metric-offset** command configures the port to add 5 to the cost of each route it learns.

**Syntax: [no] ip rip metric-offset** *num* {**in** | **out**}

The *num* variable specifies a range from 1 through 16.

> NOTE
> RIP considers a route with a metric of 16 to be unreachable. You can prevent the device from using a specific port for routes learned though that port by setting its metric to 16.

The **in** keyword applies to routes the port learns from RIP neighbors.

The **out** keyword applies to routes the port advertises to its RIP neighbors.

# Changing the administrative distance

By default, the Extreme device assigns the default RIP administrative distance (120) to RIP routes. When comparing routes based on administrative distance, the Extreme device selects the route with the lower distance. You can change the administrative distance for RIP routes.

To change the administrative distance for RIP routes, enter the **distance** command followed by a value from 1 through 255. The following example changes the administrative distance to 140 for all RIP routes.

```
device# configure terminal
device(config)# router rip
device(config-rip-router)# distance 140
```

# Configuring redistribution

You can configure the Extreme device to redistribute routes learned through Open Shortest Path First (OSPF) or Border Gateway Protocol version 4 (BGP4), connected into RIP, or static routes. When you redistribute a route from one of these other protocols into RIP, the Extreme device can use RIP to advertise the route to its RIP neighbors.

To configure redistribution, perform the following tasks.

1. Configure redistribution filters (optional). You can configure filters to permit or deny redistribution for a route based on its origin (OSPF, BGP4, and so on), the destination network address, and the route's metric. You also can configure a filter to set the metric based on these criteria.
2. Change the default redistribution metric (optional). The Extreme device assigns a RIP metric of 1 to each redistributed route by default. You can change the default metric to a value up to 15.
3. Enable redistribution.

## Configuring redistribution filters

RIP redistribution filters apply to all interfaces. Use route maps to define how you want to deny or permit redistribution.

> NOTE
> The default redistribution action is permit, even after you configure and apply redistribution filters to the virtual routing interface. If you want to tightly control redistribution, apply a filter to deny all routes as the last filter (the filter with the highest ID), and then apply filters to allow specific routes.

A route map is a named set of match conditions and parameter settings that the Extreme device can use to modify route attributes and to control redistribution of the routes into other protocols. A route map consists of a sequence of up to 50 instances. The Extreme device evaluates a route according to a route map's instances in ascending numerical order. The route is first compared against instance 1, then against instance 2, and so on. If a match is found, the Extreme device stops evaluating the route against the route map instances.

Route maps can contain match statements and set statements. Each route map contains a permit or deny action for routes that match the match statements:

- If the route map contains a permit action, a route that matches a match statement is permitted; otherwise, the route is denied.
- If the route map contains a deny action, a route that matches a match statement is denied.
- If a route does not match any match statements in the route map, the route is denied. This is the default action. To change the default action, configure the last match statement in the last instance of the route map to "permit any any".
- If there is no match statement, the route is considered to be a match.
- For route maps that contain address filters, AS-path filters, or community filters, if the action specified by a filter conflicts with the action specified by the route map, the route map's action takes precedence over the individual filter's action.

If the route map contains set statements, routes that are permitted by the route map's match statements are modified according to the set statements.

In RIP, the match statements are based on prefix lists and access control lists. Set statements are based on tag values and metric values.

To configure redistribution filters, enter the following command.

```
device(config-rip-router)# redistribute bgp route-map longroute
```

**Syntax: [no] redistribute** {**connected** | **bgp** | **ospf** | **static** [**metric** *value* | **route-map** *name*]}

The **connected** keyword applies redistribution to connected types.

The **bgp** keyword applies redistribution to BGP4 routes.

The **ospf** keyword applies redistribution to OSPF routes.

The **static** keyword applies redistribution to IP static routes.

The **metric** *value* parameter sets the RIP metric value from 1 through 15 that will be applied to the routes imported into RIP.

The **route-map** *name* parameter indicates the route map's name.

## Matching based on RIP protocol type

The **match** option has been added to the **route-map** command that allows statically configured routes or the routes learned from the IGP protocol RIP.

To configure the route map to match to RIP, enter the **match protocol rip** command.

```
device(config-routemap test)# match protocol rip
```

**Syntax: [no] match protocol rip**

### *Changing the default redistribution metric*

When the Extreme device redistributes a route into RIP, the software assigns a RIP metric (cost) to the route. By default, the software assigns a metric of 1 to each route that is redistributed into RIP. You can increase the metric that the Extreme device assigns, up to 15.

To change the RIP metric the Extreme device assigns to redistributed routes, enter a command such as the following.

```
device(config-rip-router)# default-metric 10
```

This command assigns a RIP metric of 10 to each route that is redistributed into RIP.

**Syntax: [no] default-metric** *1-15*

## Configuring route learning and advertising parameters

By default, a device learns routes from all its RIP neighbors and advertises RIP routes to those neighbors.

You can configure the following learning and advertising parameters:

- Update interval - The update interval specifies how often the device sends RIP route advertisements to its neighbors You can change the interval to a value from 3 through 65535 seconds. The default is 30 seconds.
- Learning and advertising of RIP default routes - The Extreme device can learn and advertise RIP default routes. You can disable learning and advertising of default routes on a global or individual interface basis.
- Learning of standard RIP routes - By default, the Extreme device can learn RIP routes from all its RIP neighbors. You can configure RIP neighbor filters to explicitly permit or deny learning from specific neighbors.

### *Enabling learning of RIP default routes*

By default, the Extreme device does not learn default RIP routes. You can enable learning of RIP default routes on a global or interface basis.

To enable learning of default RIP routes on a global basis, enter the following command.

```
device(config-rip-router)# learn-default
```

**Syntax: [no] learn-default**

To enable learning of default RIP routes on an interface, enter the ip rip learn-default command.

```
device(config)# interface ethernet 1/1
device(config-if-e10000-1/1)# ip rip learn-default
```

**Syntax: [no] ip rip learn-default**

### *Configuring a RIP neighbor filter*

By default, a device learns RIP routes from all its RIP neighbors. Neighbor filters allow you to specify the neighbor routers from which the Extreme device can receive RIP routes. Neighbor filters apply globally to all ports.

To configure a RIP neighbor filters, enter the **neighbor** command.

```
device(config-rip-router)# neighbor 1 deny any
```

This command configures the Extreme device so that the device does not learn any RIP routes from any RIP neighbors.

**Syntax: [no] neighbor** *filter-num* {**permit** | **deny**} {*source-ip-address* | **any**}

The following commands configure the Extreme device to learn routes from all neighbors except 10.70.12.104. Once you define a RIP neighbor filter, the default action changes from learning all routes from all neighbors to denying all routes from all neighbors except the

ones you explicitly permit. Thus, to deny learning from a specific neighbor but allow all other neighbors, you must add a filter that allows learning from all neighbors. Make sure you add the filter to permit all neighbors as the last filter (the one with the highest filter number). Otherwise, the software can match on the permit all filter before a filter that denies a specific neighbor, and learn routes from that neighbor.

```
device(config-rip-router)# neighbor 2 deny 10.70.12.104
device(config-rip-router)# neighbor 64 permit any
```

# Changing the route loop prevention method

RIP uses the following methods to prevent routing loops:

- Split horizon - The device does not advertise a route on the same interface as the one on which the Extreme device learned the route. This is the default.
- Poison reverse - The device assigns a cost of 16 ("infinite" or "unreachable") to a route before advertising it on the same interface as the one on which the Extreme device learned the route.

These loop prevention methods are configurable on a global basis as well as on an individual interface basis. One of the methods is always in effect on an interface enabled for RIP. Thus, if you disable one method, the other method is enabled.

> NOTE
> These methods are in addition to RIP's maximum valid route cost of 15.

To disable poison reverse and enable split horizon on a global basis, enter the following command.

```
device(config-rip-router)# no poison-reverse
```

**Syntax: [no] poison-reverse**

To disable poison reverse and enable split horizon on an interface, enter commands such as the following.

```
device(config)#interface ethernet 1/1
device(config-if-e10000-1/1)# no ip rip poison-reverse
```

**Syntax: [no] ip rip poison-reverse**

To disable split horizon and enable poison reverse on an interface, enter commands such as the following.

```
device(config)#interface ethernet 1/1
device(config-if-e10000-1/1)# ip rip poison-reverse
```

You can configure the Extreme device to avoid routing loops by advertising local RIP routes with a cost of 16 ("infinite" or "unreachable") when these routes go down.

```
device(config-rip-router)# poison-local-routes
```

**Syntax: [no] poison-local-routes**

# Suppressing RIP route advertisement on a VRRP or VRRPE backup interface

> NOTE
> This section applies only if you configure the device for Virtual Router Redundancy Protocol (VRRP) or VRRP Extended (VRRPE).

Normally, a VRRP or VRRPE Backup includes route information for the virtual IP address (the backed up interface) in RIP advertisements. As a result, other routers receive multiple paths for the backed up interface and might sometimes unsuccessfully use the path to the Backup rather than the path to the Master.

You can prevent the backups from advertising route information for the backed up interface by enabling suppression of the advertisements.

To suppress RIP advertisements for the backed up interface, enter the following commands.

```
device(config)# router rip
device(config-rip-router)# use-vrrp-path
```

**Syntax: [no] use-vrrp-path**

The syntax is the same for VRRP and VRRP-E.

# Configuring RIP route filters using prefix-lists and route maps

You can configure prefix lists to permit or deny specific routes, then apply them globally or to individual interfaces and specify whether the lists apply to learned routes (in) or advertised routes (out).

You can configure route maps to permit or deny specific routes, then apply a route map to an interface, and specify whether the map applies to learned routes (in) or advertised routes (out).

> NOTE
> A route is defined by the destination's IP address and network mask.

> NOTE
> By default, routes that do not match a prefix list are learned or advertised. To prevent a route from being learned or advertised, you must configure a prefix list to deny the route.

To configure a prefix list, enter commands such as the following.

```
device(config)# ip prefix-list list1  permit 10.53.4.1 255.255.255.0
device(config)# ip prefix-list list2  permit 10.53.5.1 255.255.255.0
device(config)# ip prefix-list list3  permit 10.53.6.1 255.255.255.0
device(config)# ip prefix-list list4 deny 10.53.7.1 255.255.255.0
```

The prefix lists permit routes to three networks, and deny the route to one network.

Because the default action is permit, all other routes (routes not explicitly permitted or denied by the filters) can be learned or advertised.

**Syntax: [no] ip prefix-list** *name* {**permit** | **deny**} {*source-ip-address* | **any** *source-mask* | **any**}

To apply a prefix list at the global level of RIP, enter commands such as the following.

```
device(config-rip-router)# prefix-list list1 in
```

**Syntax: no prefix-list** *name* {**in** | **out**}

To apply prefix lists to a RIP interface, enter commands such as the following.

```
device(config-if-e1000-1/2)# ip rip prefix-list list2 in
device(config-if-e1000-1/2)# ip rip prefix-list list3 out
```

**Syntax: no ip rip prefix-list** *name* {**in** | **out**}

**In** is for Inbound filtering. It applies the prefix list to routes the Extreme device learns from its neighbor on the interface.

**Out** is for Outbound filtering. It applies the prefix list to routes the Extreme device advertises to its neighbor on the interface.

The commands apply RIP list2 route filters to all routes learned from the RIP neighbor on the port and applies the lists to all routes advertised on the port.

To configure a route-map, enter commands such as the following.

```
device(config)#access-list 21 deny 160.1.0.0 0.0.255.255
device(config)#access-list 21 permit any
device(config)# route-map routemap1 permit 21
device(config-routemap routemap1)# match ip address 21
device(config)# route-map routemap2 permit 22
```

The route-map permit routes to two networks, and denies the route to one network.

**Syntax: [no] route-map** *map-name* {**permit** | **deny**} *num*

To apply a route map to a RIP interface, enter commands such as the following.

```
device(config-if-e1000-1/2)# ip rip route-map map1 in
```

**Syntax: [no] ip rip route-map** *name* {**in** | **out**}

The **route-map** can be a prefix list or an ACL. Setting this command can change the metric.

**In** applies the route map to routes the Extreme device learns from its neighbor on the interface.

**Out** applies the route map to routes the Extreme device advertises to its neighbor on the interface.

The commands apply route map map1 as route filters to routes learned from the RIP neighbor on the port.

# Setting RIP timers

When RIP is enabled, you can set protocol timers. The **timers** command specifies how often RIP update messages are sent, and it specifies the frequency of several other actions.

> NOTE
> The update timer can be modified separately with the **update-time** command.

**TABLE 55** RIP timers

| Timer | Definition | Default | Settings |
|---|---|---|---|
| Update | interval between RIP routing updates | 30 seconds | 3 through 65535 |
| Timeout | interval after which a route is considered unreachable | 180 seconds | 9 through 65535 seconds |
| Hold down | interval during which information about other paths is ignored | 180 seconds | 0 through 65535 seconds |
| Garbage collection | interval after which a route is removed from the RIP routing table | 120 seconds | 0 through 65535 seconds |

To set the timers, enter the **timer** command at the RIP router configuration level and include a value for each timer as shown in the following example.

```
device# configure terminal
device(config)# router rip
device(config-rip-router)# 60 180 180 120
```

To reset the timers to their defaults, enter the **no timer** command with the current value of all timer parameters.

**Syntax: [no] timers** *update-timer timeout-timer hold-down-timer garbage-collection-timer*

# Displaying RIP Information

To display RIP filters, enter the following command at any CLI level.

```
device# show ip rip
RIP Summary
  Default port 520
      Administrative distance is 120
      Updates every 30 seconds, expire after 180
      Holddown lasts 180 seconds, garbage collect after 120
      Last broadcast 2, Next Update 26
      Need trigger update  0, Next trigger broadcast 3
      Minimum update  interval 25, Max update Offset 5
      Split horizon is on; poison reverse is off
      Import metric 1
      Prefix List, Inbound : Not set
      Prefix List, Outbound : Not set
      Route-map, Inbound : Not set
      Route-map, Outbound : Not set
      Redistribute:

  No Neighbors are configured in RIP Neighbor Filter Table
```

**Syntax: show ip rip**

**TABLE 56** CLI display of neighbor filter information

| Field. | Defiinition |
|---|---|
| **RIP Summary** area | Shows the current configuration of RIP on the device. |
| Static metric | Shows the static metric configuration. ".not defined" means the route map has not been distributed. |
| OSPF metric | Shows what OSPF route map has been applied. |
| **Neighbor Filter Table** area | |
| Index | The filter number. You assign this number when you configure the filter. |
| Action | The action the Extreme device takes for RIP route packets to or from the specified neighbor:<br>• deny - If the filter is applied to an interface's outbound filter group, the filter prevents the Extreme device from advertising RIP routes to the specified neighbor on that interface. If the filter is applied to an interface's inbound filter group, the filter prevents the Extreme device from receiving RIP updates from the specified neighbor.<br>• permit - If the filter is applied to an interface's outbound filter group, the filter allows the Extreme device to advertise RIP routes to the specified neighbor on that interface. If the filter is applied to an interface's inbound filter group, the filter allows the Extreme device to receive RIP updates from the specified neighbor. |
| Neighbor IP Address | The IP address of the RIP neighbor. |

To display RIP filters for a specific interface, enter the following command.

```
device# show ip rip interface ethernet 1/20
Interface eth 1/20
 Rip Mode : Version 2 Running: TRUE
 Route summarization disabled
 Split horizon is on; poison reverse is off
 Default routes not accepted
 Metric-offset, Inbound 1
 Metric-offset, Outbound 0
 Prefix List, Inbound : Not set
```

```
      Prefix List, Outbound : Not set
 Route-map, Inbound : Not set
Interface ve 10
 RIP Mode : Compatible Running: TRUE
 Route summarization disabled
 Split horizon is off; poison reverse is on
 Default routes not accepted
 Metric-offset, Inbound 1
 Metric-offset, Outbound 0
 Prefix List, Inbound : Not set
 Prefix List, Outbound : Not set
 Route-map, Inbound : Not set
 Route-map, Outbound : Not set
Interface ve 20
 RIP Mode : Version1 Running: TRUE
 Route summarization enabled
 Split horizon is off; poison reverse is on
 Default routes not accepted
 Metric-offset, Inbound 1
 Metric-offset, Outbound 0
 Prefix List, Inbound : Not set
      Prefix List, Outbound : Not set
 Route-map, Inbound : Not set
 Route-map, Outbound : Not set
```

**Syntax: show ip rip interface** *ifName*

To display RIP route information, enter the following command.

```
device# show ip rip route
RIP Routing Table - 35 entries:
10.0.0.0/8, from 10.1.0.2, ve 10 (2)
         RIP, metric 4, tag 0, timers: aging 17 holddown -163
10.0.0.0/8, from 10.1.0.2, ve 10 (6)
         RIP, metric 16, tag 0, timers: holddown 19 garbage 19
10.1.1.0/24, from 10.0.0.0, eth 1/20 (34)
         MCAST, metric 1, tag 0, timers: none
10.1.0.0/24, from 10.0.0.0, ve 10 (1)
         MCAST, metric 1, tag 0, timers: none
```

**Syntax: show ip rip route**

To display current running configuration for interface 1/20, enter the following command.

```
device# show running-config interface ethernet 1/20
interface ethernet 1/20
 enable
 ip ospf area 0
 ip ospf priority 0
 ip rip v2-only
 ip address 10.1.1.2/24
 ipv6 address 2000::1/32
 ipv6 enable
!
```

To display current running configuration for ve 10, enter the following command.

```
device# show running-config interface ve 10
interface ve 10
 ip ospf area 2
 ip rip v1-compatible-v2
 ip rip poison-reverse
 ip address 10.1.0.1/24
 ipv6 address 2001:db8:1::14/64
!
```

To display current running configuration for ve 20, enter the following command.

```
device# show running-config interface ve 20
interface ve 20
```

```
 ip ospf area 1
 ip rip v1-only
 ip rip poison-reverse
 ip address 10.2.0.1/24
!
```

# Displaying CPU utilization statistics

You can display CPU utilization statistics for RIP and other IP protocols. To display CPU utilization statistics for RIP, enter the **show cpu-utilization** command at any level of the CLI.

```
device#show cpu-utilization

07:46:48 GMT+00 Mon Nov 18 2013

... Usage average for all tasks in the last 1 seconds  ...
============================================================
Name                    us/sec          %

idle                    978720          100
con                     18              0
mon                     110             0
flash                   116             0
dbg                     27              0
boot                    89              0
main                    0               0
itc                     0               0
tmr                     2201            0
ip_rx                   6413            0
sfm_mgr                 2309            0
scp                     49              0
lpagent                 0               0
console                 227             0
vlan                    163             0
mac_mgr                 84              0
mrp                     219             0
vsrp                    0               0
erp                     214             0
mxrp                    69              0
snms                    0               0
rtm                     796             0
rtm6                    686             0
ip_tx                   2858            0
rip                     0               0
l2vpn                   0               0
mpls                    0               0

(Output truncated)
```

**Syntax: show cpu-utilization**

The command lists the usage statistics for the previous five-second, one-minute, five-minute, and fifteen-minute intervals.

# RIPng



# RIPng Overview

Routing Information Protocol (RIP) is an IP route exchange protocol that uses a distance vector (a number representing a distance) to measure the cost of a given route. RIP uses a hop count as its cost or metric.

IPv6 RIP, known as Routing Information Protocol Next Generation or RIPng, functions similarly to IPv4 RIP version 2. RIPng supports IPv6 addresses and prefixes.

RIPng maintains a Routing Information Database (RIB), which is a local route table. The local RIB contains the lowest-cost IPv6 routes learned from other RIP routers. RIPng attempts to add routes from its local RIB into the main IPv6 route table.

# Configuring RIPng

To configure RIPng, you must enable RIPng globally on the Extreme device and on individual device interfaces. The following configuration tasks are optional:

- Change the default settings of RIPng timers
- Configure how the Extreme device learns and advertises routes
- Configure which routes are redistributed into RIPng from other sources
- Configure how the Extreme device distributes routes through RIPng
- Configure poison reverse parameters

## Enabling RIPng

Before configuring the device to run RIPng, you must do the following:

- Enable the forwarding of IPv6 traffic on the device using the **ipv6 unicast-routing** command.
- Enable IPv6 on each interface over which you plan to enable RIPng. You enable IPv6 on an interface by configuring an IPv6 address or explicitly enabling IPv6 on that interface.

By default, RIPng is disabled. To enable RIPng, you must enable it globally on the Extreme device and also on individual device interfaces.

NOTE
Enabling RIPng globally on the Extreme device does not enable it on individual device interfaces.

To enable RIPng globally, enter the following command.

```
device(config-rip-router)#ipv6 router rip
device(config-ripng-router)#
```

After you enter this command, the device enters the RIPng configuration level, where you can access several commands that allow you to configure RIPng.

**Syntax: [no] ipv6 router rip**

To disable RIPng globally, use the **no** form of this command.

After enabling RIPng globally, you must enable it on individual Extreme device interfaces. You can enable it on physical as well as virtual routing interfaces. For example, to enable RIPng on Ethernet interface 3/1, enter the following commands.

```
device(config)# interface ethernet 3/1
device(config-if-e100-3/1)# ipv6 rip enable
```

**Syntax: [no] ipv6 rip enable**

To disable RIPng on an individual device interface, use the **no** form of this command.

### *Enabling RIPng for a VRF instance*

To enable RIPng for a specific VRF instance, enter the following commands:

```
device(config-rip-router)#ipv6 router rip vrf red
device(config-ripng-router-vrf-red)#
```

**Syntax: [no] ipv6 router rip vrf***vrf-name*

*vrf-name* is the specified VRF name for the RIPng. If the VRF name is not specified, RIPng is configured using the default VRF.

To disable RIPng for a specific VRF instance, use the **no** form of this command.

## Configuring RIPng timers

**TABLE 57** RIPng timers

| Timer | Description | Default |
|---|---|---|
| Update | Amount of time (in seconds) between RIPng routing updates. | 30 seconds. |
| Timeout | Amount of time (in seconds) after which a route is considered unreachable. | 180 seconds. |
| Hold-down | Amount of time (in seconds) during which information about other paths is ignored. | 180 seconds. |
| Garbage-collection | Amount of time (in seconds) after which a route is removed from the routing table. | 120 seconds. |

You can adjust these timers for RIPng. Before doing so, keep the following caveats in mind:

- If you adjust these RIPng timers, Extreme strongly recommends setting the same timer values for all routers and access servers in the network.
- Setting the update timer to a shorter interval can cause the devices to spend excessive time updating the IPv6 route table.
- Extreme recommends setting the timeout timer value to at least three times the value of the update timer.
- Extreme recommends a shorter hold-down timer interval, because a longer interval can cause delays in RIPng convergence.

The following example sets updates to be advertised every 45 seconds. If a route is not heard from in 135 seconds, the route is declared unusable. Further information is suppressed for an additional 10 seconds. Assuming no updates, the route is flushed from the routing table 20 seconds after the end of the hold-down period.

```
device(config)# ipv6 router rip
device(config-ripng-router)# timers 45 135 10 20
```

**Syntax: [no] timers***update-timer timeout-timer hold-down-timer garbage-collection-timer*

Possible values for the timers are as follows:

- Update timer: 3 through 65535 seconds.
- Timeout timer: 9 through 65535 seconds.
- Hold-down timer: 9 through 65535 seconds.
- Garbage-collection timer: 9 through 65535 seconds.

NOTE
You must enter a value for each timer, even if you want to retain the current setting of a particular timer.

To return to the default values of the RIPng timers, use the **no** form of this command.

# Configuring route learning and advertising parameters

You can configure the following learning and advertising parameters:

- Learning and advertising of RIPng default routes.
- Advertising of IPv6 address summaries.
- Metric of routes learned and advertised on a device interface.

## *Configuring default route learning and advertising*

By default, the device does not learn IPv6 default routes (::/0). You can originate default routes into RIPng, which causes individual Extreme device interfaces to include the default routes in their updates. When configuring the origination of the default routes, you can also do the following:

- Suppress all other routes from the updates.
- Include all other routes in the updates.

For example, to originate default routes in RIPng and suppress all other routes in updates sent from Ethernet interface 3/1, enter the following commands.

```
device(config)# interface ethernet 3/1
device(config-if-e100-3/1)# ipv6 rip default-information only
```

To originate IPv6 default routes and include all other routes in updates sent from Ethernet interface 3/1, enter the following commands.

```
device(config)# interface ethernet 3/1
device(config-if-e100-3/1)# ipv6 rip default-information originate
```

**Syntax: [no] ipv6 rip default-information** { **only** | **originate** }

The **only** keyword originates the default routes and suppresses all other routes from the updates.

The **originate** keyword originates the default routes and includes all other routes in the updates.

To remove the explicit default routes from RIPng and suppress advertisement of these routes, use the **no** form of this command.

### *Advertising IPv6 address summaries*

You can configure RIPng to advertise a summary of IPv6 addresses from a device interface and to specify an IPv6 prefix that summarizes the routes.

If a route's prefix length matches the value specified in the **ipv6 rip summary-address** command, RIPng advertises the prefix specified in the **ipv6 rip summary-address** command instead of the original route.

For example, to advertise the summarized prefix 2001:db8::/36 instead of the IPv6 address 2001:db8:0:adff:8935:e838:78:e0ff with a prefix length of 64 bits from Ethernet interface 3/1, enter the following commands.

```
device(config)# interface ethernet 3/1
device(config-if-e100-3/1)# ipv6 address 2001:db8:0:adff:8935:e838:78:
e0ff /64
device(config-if-e100-3/1)# ipv6 rip summary-address 2001:db8::/36
```

**Syntax: [no] ipv6 rip summary-address** *ipv6-prefix*/*prefix-length*

You must specify the *ipv6-prefix* parameter in hexadecimal using 16-bit values between colons as documented in RFC 2373.

You must specify the *prefix-length* parameter as a decimal value. A slash mark (/) must follow the ipv6-prefix parameter and precede the prefix-length parameter.

To stop the advertising of the summarized IPv6 prefix, use the **no** form of this command.

### *Changing the metric of routes learned and advertised on an interface*

A device interface increases the metric of an incoming RIPng route it learns by an offset (the default is one). The device then places the route in the route table. When the device sends an update, it advertises the route with the metric plus the default offset of zero in an outgoing update message.

You can change the metric offset an individual interface adds to a route learned by the interface or advertised by the interface. For example, to change the metric offset for incoming routes learned by Ethernet interface 3/1 to one and the metric offset for outgoing routes advertised by the interface to three, enter the following commands.

```
device(config)# interface ethernet 3/1
device(config-if-e100-3/1)# ipv6 rip metric-offset 1
device(config-if-e100-3/1)# ipv6 rip metric-offset out 3
```

In this example, if Ethernet interface 3/1 learns about an incoming route, it will increase the incoming metric by two (the default offset of 1 and the additional offset of 1 as specified in this example). If Ethernet interface 3/1 advertises an outgoing route, it will increase the metric by 3 as specified in this example.

**Syntax: [no] ipv6 rip metric-offset** [**out**] *1-16*

To return the metric offset to its default value, use the **no** form of this command.

## Redistributing routes into RIPng

You can configure the Extreme device to redistribute routes from the following sources into RIPng:

- IPv6 static routes
- Directly connected IPv6 networks
- BGP4+
- IPv6 IS-IS
- OSPFv3

When you redistribute a route from BGP4+, IPv6 IS-IS, or OSPFv3 into RIPng, the device can use RIPng to advertise the route to its RIPng neighbors.

When configuring the Extreme device to redistribute routes, such as BGP4+ routes, you can optionally specify a metric for the redistributed routes. If you do not explicitly configure a metric, the default metric value of one is used.

For example, to redistribute OSPFv3 routes into RIPng, enter the following command.

```
device(config)# ipv6 router rip
device(config-ripng-router)# redistribute ospf
```

**Syntax: [no] redistribute**{ **bgp** | **connected** | **isis** | **ospf** | **static** [ **metric** *number* ] }

For the metric, specify a numerical value that is consistent with RIPng.

# Controlling distribution of routes through RIPng

You can create a prefix list and then apply it to RIPng routing updates that are received or sent on a device interface. Performing this task allows you to control the distribution of routes through RIPng.

For example, to permit the inclusion of routes with the prefix 2001:db8::/32 in RIPng routing updates sent from Ethernet interface 3/1, enter the following commands.

```
device(config)# ipv6 prefix-list routesfor2001 permit 2001:db8::/32
device(config)# ipv6 router rip
device(config-ripng-router)# distribute-list prefix-list routesfor2001 out
```

To deny prefix lengths greater than 64 bits in routes that have the prefix 2001:db8::/64 and allow all other routes received on tunnel interface 3/1, enter the following commands.

```
device(config)# ipv6 prefix-list 2001routes deny 2001:db8::/64 le 128
device(config)# ipv6 prefix-list 2001routes permit ::/0 ge 0 le 128
device(config)# ipv6 router rip
device(config-ripng-router)# distribute-list prefix-list 2001routes in
```

**Syntax: [no] distribute-list prefix-list** *name* { **in** | **out** }

The name parameter indicates the name of the prefix list generated using the **ipv6 prefix-list** command.

The **in** keyword indicates that the prefix list is applied to incoming routing updates on the specified interface.

The **out** keyword indicates that the prefix list is applied to outgoing routing updates on the specified interface.

For the *interface* parameter, you can specify the ethernet, loopback, ve, or tunnel keywords. If you specify an Ethernet interface, also specify the port number associated with the interface. If you specify a VE or tunnel interface, also specify the VE or tunnel number.

To remove the distribution list, use the **no** form of this command.

# Configuring poison reverse parameters

By default, poison reverse is disabled on a RIPng Extreme device. If poison reverse is enabled, RIPng advertises routes it learns from a particular interface over that same interface with a metric of 16, which means that the route is unreachable.

If poison reverse is enabled on the RIPng Extremedevice, it takes precedence over split horizon (if it is also enabled).

To enable poison reverse on the RIPng Extreme device, enter the following commands.

```
device(config)# ipv6 router rip
device(config-ripng-router)# poison-reverse
```

**Syntax:[no] poison-reverse**

To disable poison-reverse, use the **no** form of this command.

By default, if a RIPng interface goes down, the Extreme device does not send a triggered update for the interface's IPv6 networks.

To better handle this situation, you can configure a RIPng Extreme device to send a triggered update containing the local routes of the disabled interface with an unreachable metric of 16 to the other RIPng routers in the routing domain. You can enable the sending of a triggered update by entering the following commands.

```
device(config)# ipv6 router rip
device(config-ripng-router)# poison-local-routes
```

**Syntax: [no] poison-local-routes**

To disable the sending of a triggered update, use the **no** form of this command.

# Clearing RIPng routes from IPv6 route table

To clear all RIPng routes from the RIPng route table and the IPv6 main route table and reset the routes, enter the following command at the Privileged EXEC level or any of the configuration levels of the CLI.

```
device# clear ipv6 rip route
```

**Syntax: clear ipv6 rip route**

## Clearing RIPng for a VRF instance

To clear all RIPng routes for a specific VRF, enter the following command:

```
device# clear ipv6 rip vrf red route
```

**Syntax: clear ipv6 rip vrf** *vrf-name* **route**

# Displaying RIPng information

You can display the following RIPng information:

- RIPng configuration
- RIPng routing table

## Displaying RIPng configuration

To display RIPng configuration information, enter the **show ipv6 rip** command at any CLI level.

```
device# show ipv6 rip
IPv6 rip enabled, port 521
      Administrative distance is 120
      Updates every 30 seconds, expire after 180
      Holddown lasts 180 seconds, garbage collect after 120
      Split horizon is on; poison reverse is off
      Default routes are not generated
      Periodic updates 5022, trigger updates 10
      Distribute List, Inbound : Not set
      Distribute List, Outbound : Not set
      Redistribute: CONNECTED
```

**Syntax: show ipv6 rip**

**TABLE 58** show ipv6 rip output descriptions

| Field | Description |
|---|---|
| IPv6 RIP status/port | The status of RIPng on the device. Possible status is "enabled" or "disabled."<br>The UDP port number over which RIPng is enabled. |
| Administrative distance | The setting of the administrative distance for RIPng. |
| Updates/expiration | The settings of the RIPng update and timeout timers. |
| Holddown/garbage collection | The settings of the RIPng hold-down and garbage-collection timers. |
| Split horizon/poison reverse | The status of the RIPng split horizon and poison reverse features. Possible status is "on" or "off." |
| Default routes | The status of RIPng default routes. |
| Periodic updates/trigger updates | The number of periodic updates and triggered updates sent by the RIPng Extreme device. |
| Distribution lists | The inbound and outbound distribution lists applied to RIPng. |
| Redistribution | The types of IPv6 routes redistributed into RIPng. The types can include the following:<br>• STATIC - IPv6 static routes are redistributed into RIPng.<br>• CONNECTED - Directly connected IPv6 networks are redistributed into RIPng.<br>• BGP - BGP4+ routes are redistributed into RIPng.<br>• IS-IS – IPv6IS-IS routes are redistributed into RIPng.<br>• OSPF - OSPFv3 routes are redistributed into RIPng. |

## *Displaying RIPng configuration for a VRF instance*

To display the RIPng configuration information for a VRF instance, enter the following command:

```
device# show ipv6 rip vrf red
IPv6 rip enabled, port 521
      Administrative distance is 120
      Updates every 30 seconds, expire after 180
      Holddown lasts 180 seconds, garbage collect after 120
      Split horizon is on; poison reverse is off
      Default originate routes are not generated
      Periodic updates 1137, trigger updates 6
      Distribute List, Inbound : Not set
      Distribute List, Outbound : Not set
      Redistribute: BGP
```

**Syntax: ipv6 rip vrf** *vrf-name*

# Displaying RIPng routing table

To display the RIPng routing table, enter the following command at any CLI level.

```
device# show ipv6 rip route
IPv6 RIP Routing Table - 4 entries:
2001:db8::/64, from ::, null (0)
         CONNECTED, metric 1, tag 0, timers: none
2001:db8:46a::/64, from ::, null (1)
         CONNECTED, metric 1, tag 0, timers: none
2001:db8::1/128, from ::, null (2)
          CONNECTED, metric 1, tag 0, timers: none
2001:db8:2::/64, from ::, null (3)
CONNECTED, metric 1, tag 0, timers: none
```

**Syntax: show ipv6 rip route** [ *ipv6-prefix*/*prefix-length* | *ipv6-address* ]

The *ipv6-prefix/prefix-length* parameters restrict the display to the entries for the specified IPv6 prefix. You must specify the *ipv6-prefix* parameter in hexadecimal using 16-bit values between colons as documented in RFC 2373. You must specify the *prefix-length* parameter as a decimal value. A slash mark (/) must follow the *ipv6-prefix* parameter and precede the *prefix-length* parameter.

The *ipv6-address* parameter restricts the display to the entries for the specified IPv6 address. You must specify this parameter in hexadecimal using 16-bit values between colons as documented in RFC 2373.

**TABLE 59** show ipv6 rip route output descriptions

| Field | Description |
|---|---|
| IPv6 RIP Routing Table entries | The total number of entries in the RIPng routing table. |
| *ipv6-prefix /prefix-length*<br>*ipv6-address* | The IPv6 prefix and prefix length.<br>The IPv6 address. |
| Next-hop router | The next-hop router for this Extreme device. If :: appears, the route is originated locally. |
| Interface | The interface name. If "null" appears, the interface is originated locally. |
| Source of route | The source of the route information. The source can be one of the following:<br>• RIP - routes learned by RIPng.<br>• CONNECTED - IPv6 routes redistributed from directly connected networks.<br>• STATIC - IPv6 static routes are redistributed into RIPng.<br>• BGP - BGP4+ routes are redistributed into RIPng.<br>• OSPF - OSPFv3 routes are redistributed into RIPng. |
| Metric *number* | The cost of the route. The *number* parameter indicates the number of hops to the destination. |
| Tag *number* | The tag value of the route. |
| Timers | Indicates if the hold-down timer or the garbage-collection timer is set. |

## *Displaying RIPng routing table for a VRF instance*

To display the RIPng route information for a specified VRF, enter the following command at any CLI level.

```
device# show ipv6 rip vrf red route
IPv6 RIP Routing Table - 4 entries:
2001:db8::/64, from ::, null (0)
         CONNECTED, metric 1, tag 0, timers: none
2001:db8:46a::/64, from ::, null (1)
         CONNECTED, metric 1, tag 0, timers: none
2001:db8::1/128, from ::, null (2)
          CONNECTED, metric 1, tag 0, timers: none
2001:db8:2::/64, from ::, null (3)
CONNECTED, metric 1, tag 0, timers: none
```

**Syntax: ipv6 rip** [ **vrf** *vrf-name* ] **route** [ *ipv6-prefix/prefix-length* | *ipv6-address* ]

# VRRPv2



# VRRPv2 overview

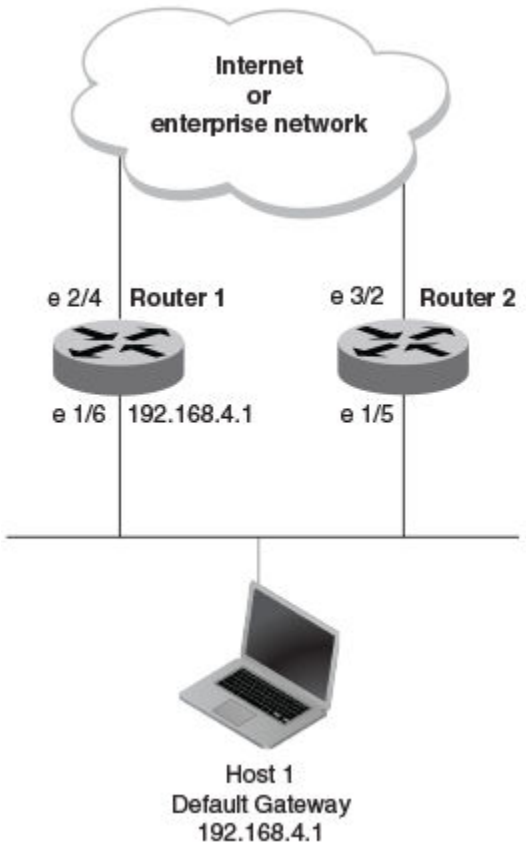
Virtual Router Redundancy Protocol (VRRP) is an election protocol that provides redundancy to routers within a Local Area Network (LAN).

VRRP was designed to eliminate a single point of failure in a static default-route environment by dynamically assigning virtual IP routers to participating hosts. A virtual router is a collection of physical routers whose interfaces must belong to the same IP subnet. A virtual router ID (VRID) is assigned to each virtual router, but there is no restriction against reusing a VRID with a different address mapping on different LANs.

> NOTE
> VRRP extended (VRRP-E) is an extended version of the VRRP protocol. Extreme Networks developed VRRP-E as a proprietary protocol to address some limitations in standards-based VRRP.

Before examining more details about how VRRP works, it is useful to see why VRRP was developed to solve the issue of a single point of failure.

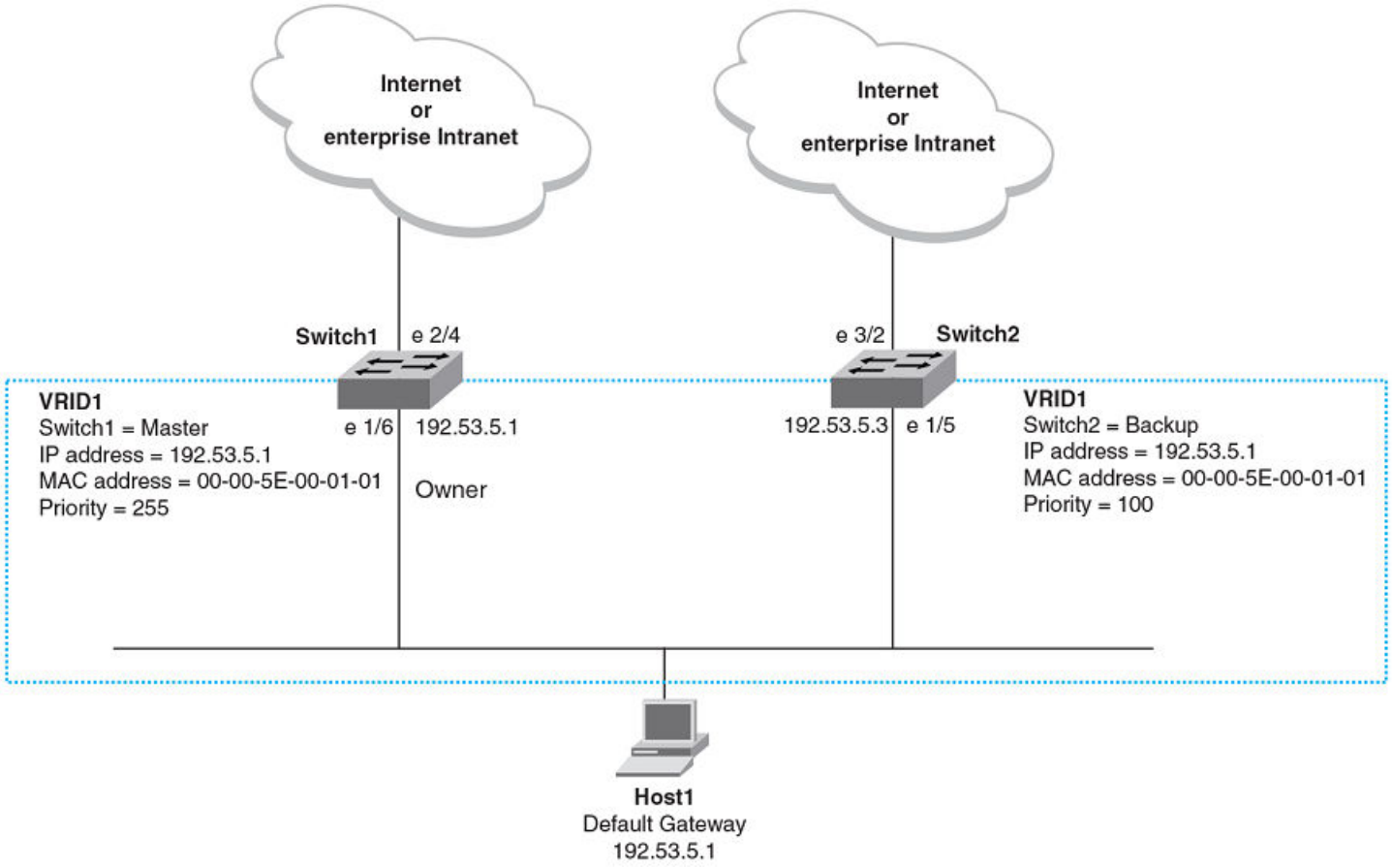
**FIGURE 65** Single point of failure with Device 1 being the Host1 default gateway




To connect to the Internet or an internal intranet Host 1, in the figure, uses the IP address of 192.168.4.1 on Router 1 as its default gateway. If this interface goes down, Host 1 is cut off from the rest of the network. Router 1 is a single point of failure for Host 1 to access other networks. In small networks, the administrative burden of configuring Router 2 as the new default gateway is not an issue, but in larger networks reconfiguring default gateways is impractical. Configuring a VRRP virtual router on Router 1 and Router 2 provides a redundant path for the hosts. VRRP allows you to provide alternate router paths for a host without changing the IP address or MAC address by which the host knows its gateway.

To illustrate how VRRP works, the following figure shows the same network, but a VRRP virtual router is configured on the two physical routers, Router 1 and Router 2. This virtual router provides redundant network access for Host 1. If Router 1 were to fail, Router 2 would provide the default gateway out of the subnet.

**FIGURE 66** Devices configured as VRRP virtual routers for redundant network access for Host 1




The blue rectangle in the figure represents a VRRP virtual router. When you configure a virtual router, one of the configuration parameters is a group number (also known as a virtual router ID or VRID), which can be a number from 1 through 255. The virtual router is identified with a group, and within the VRRP group, there is one physical device that forwards packets for the virtual router and this is called a master VRRP device. The VRRP master device may be a Layer 3 switch or a router.

In VRRP, one of the physical IP addresses is configured as the IP address of the virtual router, the virtual IP address. The device on which the virtual IP address is assigned becomes the VRRP owner, and this device responds to packets addressed to any of the IP addresses in the virtual router group. The owner device becomes the master VRRP device by default and is assigned the highest priority. Backup devices are configured as members of the virtual router group, and, if the master device goes offline, one of the backup devices assumes the role of the master device.

NOTE
VRRP operation is independent of BGP4, OSPF, and RIP. Their operation is unaffected when VRRP is enabled on the same interface as BGP4, OSPF, and RIP.

## VRRP terminology

Before implementing VRRP in your network, you must understand some key terms and definitions.

The following VRRP-related terms are in logical order, not alphabetic order:

| | |
|---|---|
| *Virtual router* | A collection of physical routers that can use VRRP to provide redundancy to routers within a LAN. |
| *Virtual router ID* | A group of physical routers that are assigned to the same virtual router ID (VRID). |
| *Virtual router address* | The virtual router IP address must belong to the same subnet as a real IP address configured on the VRRP interface, and it can be the same as a real IP address configured on the VRRP interface. The virtual router whose virtual IP address is the same as a real IP address is the IP address owner and the default master. |
| *Owner* | The owner is the physical router whose real interface IP address is the IP address that you assign to the virtual router. The owner responds to packets addressed to any of the IP addresses in the corresponding virtual router. The owner, by default, is the master and has the highest priority (255). |
| *Master* | The physical router that responds to packets addressed to any of the IP addresses in the corresponding virtual router. For VRRP, if the physical router whose real interface IP address is the IP address of the virtual router, then this physical router is always the master. |
| *Backup* | Routers that belong to a virtual router, but are not the master. If the master becomes unavailable, the backup router with the highest priority (a configurable value) becomes the new master. By default, routers are given a priority of 100. |

# VRRP hold timer

The hold timer delays the preemption of a master VRRP device by a high-priority backup device.

A hold timer is used when a VRRP-enabled device that was previously a master device failed, but is now back up. This restored device now has a higher priority than the current VRRP master device, and VRRP normally triggers an immediate switchover. In this situation, it is possible that not all software components on the backup device have converged yet. The hold timer can enforce a waiting period before the higher-priority backup device assumes the role of master VRRP device again. The timer must be set to a number greater than 0 seconds for this functionality to take effect.

Hold timer functionality is supported in both version 2 and version 3 of VRRP and VRRP-E.

# VRRP interval timers

Various timers for the intervals between hello messages sent between devices running VRRP can be configured.

## *Hello intervals*

Hello messages are sent from the master VRRP device to the backup devices. The purpose of the hello messages is to determine that the master device is still online. If the backup devices stop receiving hello messages for a period of time, as defined by the dead (or master-down-interval) interval, the backup devices assume that the master device is offline. When the master device is offline, the backup device with the highest priority assumes the role of the master device.

NOTE

The hello intervals must be set to the same value on both owner and backup devices for the same VRID.

## *Dead interval*

The dead interval is defined as the period of time for which backup devices wait for a hello message from the master device before assuming that the master device is offline. An immediate switchover to the backup device with the highest priority is triggered after the dead interval expires and there is no hello message from the master device. If a value for the dead interval is not configured, the default value is calculated as three times the hello interval plus the skew time. Skew time is defined as (256 – priority)/256.

NOTE

The dead interval must be set to the same value on both owner and backup devices for the same VRID.

### *Backup hello message state and interval*

By default, backup devices do not send hello messages to advertise themselves to the master device. Hello messages from backup devices can be activated, and the messages are sent at 60-second intervals, by default. The interval between the backup hello messages can be modified.

## VRRP authentication

The VRRP authentication type is not a parameter specific to the virtual router. VRRP uses the authentication type associated with the interfaces on which the virtual router is defined.

If your interfaces do not use authentication, neither does VRRP. For example, if you configure your device interfaces to use an MD5 password to authenticate traffic, VRRP uses the same MD5 password, and VRRP packets that do not contain the password are dropped.

In summary, if the interfaces on which you configure the virtual router use authentication, the VRRP or VRRP Extended (VRRP-E) packets on those interfaces must use the same authentication. The following VRRP and VRRP-E authentication types are supported:

- No authentication—The interfaces do not use authentication. This authentication type is the default for VRRP and VRRP-E.
- Simple—The interfaces use a simple text string as a password in packets that they send. If the interfaces use simple password authentication, the virtual router configured on the interfaces must use the same authentication type and the same password.
- MD5—This method of authentication ensures that the packet is authentic and cannot be modified in transit. Syslog and SNMP traps are generated when a packet is dropped due to MD5 authentication failure. MD5 authentication is supported only in VRRP-E, and the device configuration is unique on a per-interface basis. The MD5 authentication configuration on an interface takes effect for all VRRP-E virtual routers configured on a particular interface.

  NOTE
  Authentication is not supported for VRRPv3.

## VRRP master device abdication to backup device

To allow temporary control of a VRRP virtual router ID (VRID) to pass to a backup device, you can force the master device to abdicate to a backup device by setting a lower priority.

Changing the priority of a VRRP master device allows a temporary abdication of the master device status to allow a backup device with a higher priority to assume the master device role. By default, a VRRP owner device has a priority of 255, and the lower priority must be set to a lower priority than at least one of the backup devices associated with the VRID.

When you change the priority of a VRRP owner, the change takes effect only for the current power cycle. The change is not saved to the startup configuration file when you save the configuration, and it is not retained across a reload or reboot. Following a reload or reboot, the VRRP owner again has priority 255.

NOTE
This feature supports IPv4 VRRP only. IPv6 VRRP, VRRP-E, and IPv6 VRRP-E are not supported.

## ARP and VRRP control packets

Control packets for ARP and VRRP are handled differently by VRRP and VRRP-E.

### *Source MAC addresses in VRRP control packets*

- VRRP—The virtual MAC address is the source.
- VRRP-E—The physical MAC address is the source.

### *VRRP control packets*

- VRRP—Control packets are IP type 112 (reserved for VRRP), and they are sent to the VRRP multicast address 224.0.0.18.
- VRRP-E—Control packets are UDP packets destined to port 8888, and they are sent to the all-router multicast address 224.0.0.2.

### *Gratuitous ARP*

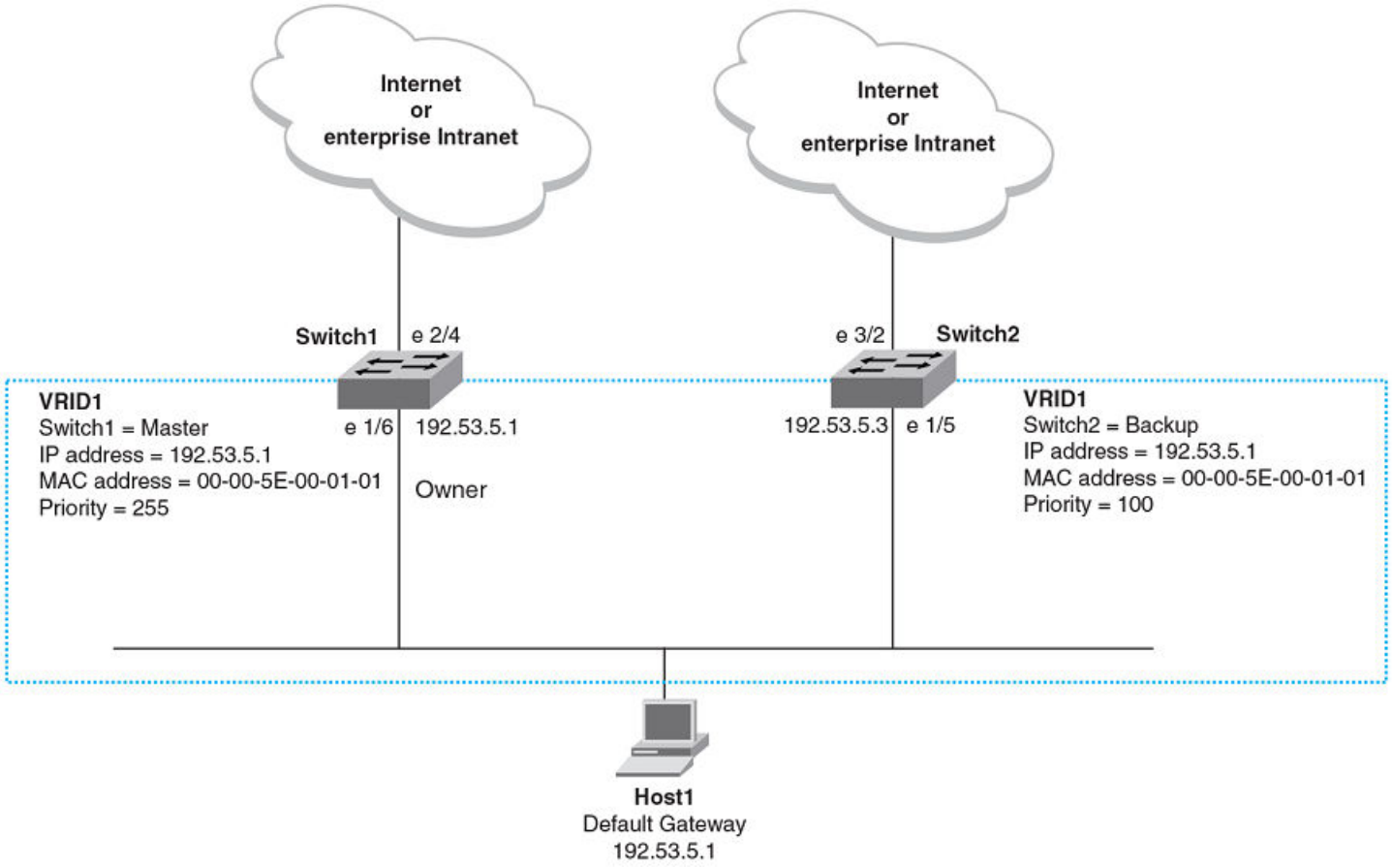
When a VRRP device (either master or backup) sends an ARP request or a reply packet, the MAC address of the sender is the MAC address of the router interface. One exception is if the owner sends an ARP request or a reply packet, in which case the MAC address of the sender is the virtual MAC address. Only the master answers an ARP request for the virtual router IP address. Any backup router that receives this request forwards the request to the master.

- VRRP—A control message is sent only once when the VRRP device assumes the role of the master.
- VRRP-E—A control message is sent every 2 seconds by the VRRP-E master device because VRRP-E control packets do not use the virtual MAC address.

# Enabling an owner VRRP device

This task is performed on the device that is designated as the owner VRRP device because the IP address of one of its physical interfaces is assigned as the IP address of the virtual router. For example, Router 1 is the owner VRRP device in the figure that follows. For each VRRP session, there are master and backup routers, and the owner router is elected, by default, as the master router.

**FIGURE 67** Basic VRRP topology




1. On the device designated as the owner VRRP device, from privileged EXEC mode, enter global configuration mode by issuing the **configure terminal** command.

```
device# configure terminal
```

2. Globally enable VRRP.

```
device(config)# router vrrp
```

3. Configure the Ethernet interface link for Router 1.

```
device(config)# interface ethernet 1/6
```

4. Configure the IP address of the interface.

```
device(config-if-e1000-1/6)# ip address 10.53.5.1/24
```

5. Assign Router 1 to the virtual router ID (VRID) 1.

```
device(config-if-e1000-1/6)# ip vrrp vrid 1
```

NOTE
You can assign a VRID number in the range of 1 through 255.

6. Designate this router as the VRRP owner device.

```
device(config-if-e1000-1/6-vrid-1)# owner
```

7. Configure the IP address of the VRID.

```
device(config-if-e1000-1/6-vrid-1)# ip-address 10.53.5.1
```

8. Enable the VRRP session.

```
device(config-if-e1000-1/6-vrid-1)# activate
```

The following example configures a VRRP owner device.

```
device# configure terminal
device(config)# router vrrp
device(config)# interface ethernet 1/6
device(config-if-e1000-1/6)# ip address 10.53.5.1/24
device(config-if-e1000-1/6)# ip vrrp vrid 1
device(config-if-e1000-1/6-vrid-1)# owner
device(config-if-e1000-1/6-vrid-1)# ip-address 10.53.5.1
device(config-if-e1000-1/6-vrid-1)# activate
VRRP router 1 for this interface is activating
```

# Enabling a backup VRRP device

This task is performed on any device that is designated as a backup VRRP device. For each VRRP virtual routing instance, there is one master device and all other devices are backups. For example, Router 2 in Figure 67 on page 599 is assigned as a backup device. Repeat this task for all devices that are to be designated as backup devices.

1. On the device designated as a backup VRRP device, from privileged EXEC mode, enter global configuration mode by issuing the **configure terminal** command.

```
device# configure terminal
```

2. Globally enable VRRP.

```
device(config)# router vrrp
```

3. Configure the Ethernet interface link.

```
device(config)# interface ethernet 1/5
```

4. Configure the IP address of the interface for Router 2. All devices configured for the same virtual router ID (VRID) must be on the same subnet.

```
device(config-if-e1000-1/5)# ip address 10.53.5.3/24
```

5. Assign Router 2 to VRID 1, the same VRID as Router 1.

```
device(config-if-e1000-1/5)# ip vrrp vrid 1
```

NOTE

You can assign a VRID number in the range of 1 through 255.

6. Designate this router as a backup VRRP device.

```
device(config-if-e1000-1/5-vrid-1)# backup priority 110
```

   While configuring a backup device, you can set a priority that is used when a master VRRP device goes offline. The backup device with the highest priority will assume the role of master device.

7. Configure the number of seconds between hello messages.

```
device(config-if-e1000-1/5-vrid-1)# hello-interval 10
```

8. By default, backup VRRP devices do not send hello messages to advertise themselves to the master. Use the following command to enable a backup router to send hello messages to the master VRRP device.

```
device(config-if-e1000-1/5-vrid-1)# advertise backup
```

9. Configure the IP address of the VRID.

```
device(config-if-e1000-1/5-vrid-1)# ip-address 10.53.5.1
```

   The VRID IP address is the same virtual IP address you used for Router 1.

10. Enable the VRRP session.

```
device(config-if-e1000-1/5-vrid-1)# activate
VRRP router 1 for this interface is activating
```

The following example configures a VRRP backup device.

```
device# configure terminal
device(config)# router vrrp
device(config)# interface ethernet 1/5
device(config-if-e1000-1/5)# ip address 10.53.5.3/24
device(config-if-e1000-1/5)# ip vrrp vrid 1
device(config-if-e1000-1/5-vrid-1)# backup priority 110
device(config-if-e1000-1/5-vrid-1)# hello-interval 10
device(config-if-e1000-1/5-vrid-1)# advertise backup
device(config-if-e1000-1/5-vrid-1)# ip-address 10.53.5.1
device(config-if-e1000-1/5-vrid-1)# activate
VRRP router 1 for this interface is activating
```

# Configuring simple text authentication on VRRP interfaces

A simple text password can be used for interface authentication in a network. VRRP uses the authentication type associated with the interfaces on which you define the virtual router ID (VRID).

A VRRP session must be configured and running.

If you configure your device interfaces to use a simple password to authenticate traffic, VRRP interfaces can be configured with the same simple password, and VRRP packets that do not contain the password are dropped. If your interfaces do not use authentication, neither does VRRP. Repeat this task on all interfaces on all devices that support the VRID.

NOTE

This task supports VRRPv2 and VRRP-Ev2 only. VRRPv3 and VRRP-Ev3 are not supported.

1. From privileged EXEC mode, enter global configuration mode by issuing the **configure terminal** command.

```
device# configure terminal
```

2. Globally enable VRRP.

```
device(config)# router vrrp
```

3. Configure an Ethernet interface.

```
device(config)# interface ethernet 1/6
```

4. Enter the simple text password configuration using the **ip vrrp auth-type** command with a text password.

```
device(config-if-e10000-1/6)# ip vrrp auth-type simple-text-auth yourpwd
```

5. Verify the password on the interface using the **show ip vrrp** command with either the VRID or Ethernet options.

```
device(config-if-e10000-1/6-vrid-1)# show ip vrrp

Total number of VRRP routers defined: 1
Interface ethernet 1/6
auth-type simple text authentication
VRID 1
state backup
administrative-status enabled
mode owner
priority 99
current priority 99
hello-interval 1 sec
ip-address 10.53.5.1
backup routers 10.53.5.2
```

In this example, the authentication type is simple text authentication. A **show running-config** command with appropriate parameters will actually display the password. The output verifies the type of authentication.

# Configuring MD5 authentication on VRRP interfaces

Interfaces can be configured with an MD5 encrypted password for authentication, and VRRP can use the same authentication type associated with the interfaces on which you define the virtual router ID (VRID).

If you configure your device interfaces to use an MD5 encrypted password to authenticate traffic, VRRP interfaces can be configured with the same MD5 password, and VRRP packets that do not contain the password are dropped. If your interfaces do not use authentication, neither does VRRP. Repeat this task on all interfaces on all devices that support the VRID.

1. From privileged EXEC mode, enter global configuration mode by issuing the **configure terminal** command.

```
device# configure terminal
```

2. Globally enable VRRP.

```
device(config)# router vrrp
```

3. Specify an interface associated with the VRRP VRID.

```
device(config)# interface ethernet 1/6
```

4. Enter the MD5 password configuration using the **ip vrrp auth-type** command with a text password. The password will be encrypted when saved in the configuration file. When an MD5 authentication password is configured on an interface, a syslog message is displayed.

```
device(config-if-e10000-1/6)# ip vrrp auth-type md5-auth gy42mb
Aug 10 18:17:39 VRRP: Configuration VRRP_CONFIG_MD5_AUTHENTICATION request received
Aug 10 18:17:39 VRRP: Port 1/6, VRID 2 - send advertisement
Ver:3 Type:1 Vrid:2 Pri:240 #IP:1 AuthType:2 Adv:1 Chksum:0x0000
HMAC-MD5 CODE:[00000000000000000000400010]
IpAddr: 10.53.5.1
```

5. Verify the password on the interface using the **show ip vrrp** command.

```
device(config-if-e10000-1/6-vrid-1)# show ip vrrp

Total number of VRRP routers defined: 1
Interface ethernet 1/6
auth-type MD5 authentication
VRID 1
state backup
administrative-status enabled
mode owner
priority 99
current priority 99
hello-interval 1 sec
ip-address 10.53.5.1
backup routers 10.53.5.2
```

   In this example, the auth-type is MD5 authentication where the entered password is encrypted. A **show run** command with appropriate parameters will actually display the encrypted password, and you can use the **enable password-display** command to actually display the encrypted password. The output verifies the type of authentication.

The following example enables MD5 authentication on Ethernet interface 1/6 and verifies the authentication type.

```
device# configure terminal
device(config)# router vrrp
device(config)# interface ethernet 1/6
device(config-if-e10000-1/6)# ip vrrp auth-type MD5 yourpwd
device(config-if-e10000-1/6-vrid-1)# show ip vrrp

Total number of VRRP routers defined: 1
Interface ethernet 1/6
auth-type MD5 authentication
VRID 1
state backup
administrative-status enabled
mode owner
priority 99
current priority 99
hello-interval 1 sec
ip-address 10.53.5.1
backup routers 10.53.5.2
```

# Abdicating VRRP master device status

Changing the priority of a VRRP master device allows a temporary abdication of the master device status to allow a backup device with a higher priority to assume the master device role.

A VRRP session must be configured and running.

When you change the priority of a VRRP owner, the change takes effect only for the current power cycle. The change is not saved to the startup configuration file when you save the configuration, and it is not retained across a reload or reboot. Following a reload or reboot, the VRRP owner again has priority 255.

NOTE
This task is supports IPv4 VRRP only. IPv6 VRRP, VRRP-E, and IPv6 VRRP-E are not supported.

1. On the master device and from privileged EXEC mode, enter global configuration mode by issuing the **configure terminal** command.

```
device# configure terminal
```

2. Globally enable VRRP.

```
device(config)# router vrrp
```

3. Configure an Ethernet interface.

```
device(config)# interface ethernet 1/6
```

4. Enter the virtual router ID (VRID) for which the device is the VRRP owner.

```
device(config-if-e1000-1/6)# ip vrrp vrid 1
```

NOTE
You can assign a VRID number in the range of 1 through 255.

5. Enter a priority for this device that is lower than the priority of at least one backup device associated with the VRID.

```
device(config-if-e1000-1/6-vrid-1)# owner priority 99
```

6. Verify the abdication of the master device using the **show ip vrrp** command.

```
device(config-if-e10000-1/6-vrid-1)# show ip vrrp

Total number of VRRP routers defined: 1
Interface ethernet 1/6
auth-type no authentication
VRID 1
state backup
administrative-status enabled
mode owner
priority 99
current priority 99
hello-interval 1 sec
ip-address 10.53.5.1
backup routers 10.53.5.2
```

In this example, the mode shows this device as the owner of the virtual router (mode owner), but the VRRP priority for the device is only 99 and the state is now backup instead of master. The administrative status is still enabled. The output verifies that this device is now a backup device.

# Tracked ports and track priority with VRRP and VRRP-E

Port tracking allows interfaces not configured for VRRP or VRRP-E to be monitored for link-state changes that can result in dynamic changes to the VRRP device priority.

A tracked port allows you to monitor the state of the interfaces on the other end of a route path. A tracked interface also allows the virtual router to lower its priority if the exit path interface goes down, allowing another virtual router in the same VRRP (or VRRP-E) group to take over. When a tracked interface returns to an up state, the configured track priority is added to the current virtual router priority value. The following conditions and limitations exist for tracked ports:

- Track priorities must be lower than VRRP or VRRP-E priorities.
- The dynamic change of router priority can trigger a master device switchover if preemption is enabled. However, if the router is an owner, the master device switchover will not occur.
- The maximum number of interfaces that can be tracked for a virtual router is 16.
- Port tracking is allowed for physical interfaces and port channels.

## Tracking ports and setting the VRRP priority

Configuring port tracking on an exit path interface and setting a priority on a VRRP device enables VRRP to monitor the interface. For VRRP, if the interface goes down, the device priority is set to the priority value and another backup device with a higher priority assumes the role of master. For VRRP-E, if the interface goes down, the device priority is lowered by the priority value and another backup device with a higher priority assumes the role of master.

Configure this task on the device on which the tracked interface exists.

NOTE
Only IPv4/IPv6 IPsec tunnels can be configured with a specific **track-port** command priority. Other interfaces use the track priority associated with the **owner** or **backup** commands.

1. Enter the **configure terminal** command to enter global configuration mode.

```
device# configure terminal
```

2. Enter the **router vrrp** command to configure VRRP globally.

```
device(config)# router vrrp
```

3. Configure the Ethernet interface.

```
device(config)# interface ethernet 1/6
```

4. Enter the IP address for the interface to be used for the virtual router ID (VRID).

```
device(config-if-e10000-1/6)# ip address 10.53.5.3/24
```

5. Enter the following command to enter the appropriate VRRP virtual router ID (VRID) mode.

```
device(config-if-e10000-1/6)# ip vrrp vrid 1
```

6. Enter the **track-port** command to set the track port and priority:

```
device(config-if-e10000-1/6-vrid-1)# track-port tunnel 1 priority 40
```

   The priority value is used when a tracked port goes down and the new priority is set to this value. Ensure that the priority value is lower than the priority set for any existing master or backup device to force a renegotiation for the master device.

The following example shows how to configure tunnel 1 on virtual router 1 to be tracked; if the interface fails, the VRRP priority of the device becomes 40, forcing a negotiation for a new master device.

```
device# configure terminal
device(config)# router vrrp
device(config)# interface ethernet 1/6
device(config-if-e10000-1/6)# ip address 10.53.5.1/24
device(config-if-e10000-1/6)# ip vrrp vrid 1
device(config-if-e10000-1/6-vrid-1)# track-port tunnel 1 priority 40
```

# VRRP backup preemption

Preemption of a backup VRRP device acting as a master device is allowed when another backup device has a higher priority.

By default, preemption is enabled for VRRP. In VRRP, preemption allows a backup device with the highest priority to become the master device when the master (also the owner) device goes offline. If another backup device is added with a higher priority, it will assume the role of the master VRRP device. In some larger networks there may be a number of backup devices with varying levels of priority, and preemption can cause network flapping. To prevent the flapping, disable preemption.

NOTE
If preemption is disabled for VRRP, the owner device is not affected because the owner device always preempts the active master. When the owner device is online, the owner device assumes the role of the master device regardless of the setting for the preempt parameter.

In VRRP-E, preemption is disabled by default. In situations where a new backup device is to be added with a higher priority, preemption can be enabled. There are no owner devices in VRRP-E to automatically preempt a master device.

## Disabling VRRP backup preemption

VRRP backup preemption can be disabled to avoid route flapping when a backup VRRP device that is acting as the master device could be preempted by another backup device with a higher priority value.

A VRRP or VRRP-E session must be globally enabled using the **router vrrp** or **router vrrp-extended** command in global configuration mode.

Preemption is enabled by default for VRRP and VRRP-E, but if several devices come back online with higher priorities than the original backup device, route flapping can occur as these devices preempt each other. The following steps can be used when you want to avoid a backup device acting as the master from being preempted by another backup device with a higher priority value.

1. Enter interface configuration mode.

```
device(config)# interface ethernet 1/5
```

2. Enter the IP address for the interface to be used for the virtual router ID (VRID).

```
device(config-if-e10000-1/5)# ip address 10.53.5.3/24
```

3. Enter the following command to enter the appropriate VRRP VRID mode.

```
device(config-if-e10000-1/5)# ip vrrp vrid 1
```

4. Enter the **non-preempt-mode** command to disable backup preemption.

```
device(config-if-e10000-1/5-vrid-1)# non-preempt-mode
```

   Even if a backup device has a higher priority than the current backup acting as a master device, the backup device will not assume the role of the VRRP master device.

The following example disables preemption on a backup VRRP device.

```
device(config)# router vrrp
device(config)# interface ethernet 1/5
device(config-if-e10000-1/5)# ip address 10.53.5.3/24
device(config-if-e10000-1/5)# ip vrrp vrid 1
device(config-if-e10000-1/5-vrid-1)# non-preempt-mode
```

# Virtual router MAC address

When you configure a virtual routing ID (VRID), the software automatically uses the MAC address as the MAC address of the virtual router. The first five octets of the address are the standard MAC prefix for VRRP packets. The last octet is the VRID.

When the virtual router becomes the master router, it broadcasts a gratuitous ARP (GARP) request containing the virtual router's MAC address for each IP address associated with the virtual router. Hosts use the MAC address of the virtual router in routed traffic they send to their default IP gateway.

You can manually configure a unique virtual MAC address for each IPv4 and IPv6 VRRP instance per VRID. If there is no manually configured virtual MAC address for a VRRP instance, the system automatically assigns one.

The ability to configure a unique virtual MAC address is subject to the following limitations:

- This feature does not support configurable VRRP virtual MAC addresses over Multi-Chassis Trunking (MCT).
- This feature has no impact on short-path forwarding for VRRP-E.

NOTE
A virtual MAC address can be dynamically updated while a VRRP or VRRP-E session is enabled. When the VRRP or VRRP-E virtual MAC address is modified on the master device, expect a traffic drop until the host device receives the GARP or Router Advertisement (RA) containing the updated virtual MAC address from the master VRRP device.

## Configuring unique virtual MAC addresses per VRID

In addition to system-configured standards-based virtual MAC addresses, you can manually configure a unique virtual MAC address for each IPv4 and IPv6 VRRP instance per virtual routing ID (VRID). If there is no manually configured virtual MAC address for a VRRP instance, the system automatically assigns one.

For the MLX Series platform, you can configure a maximum of 2000 virtual MAC addresses. For CES 2000 Series and CER 2000 Series platforms, you can configure a maximum of 255 virtual MAC addresses.

NOTE
System-assigned virtual MAC addresses and manually configured virtual MAC addresses can exist at the same time on the device under the same VRID, however the configured value takes precedence. When the configured value is deleted, the assigned value again applies.

NOTE
A virtual MAC address can be dynamically updated while a VRRP or VRRP-E session is enabled. When the VRRP or VRRP-E virtual MAC address is modified on the master device, expect a traffic drop until the host device receives the GARP request or Router Advertisement (RA) containing the updated virtual MAC address from the master VRRP device.

To configure a unique VRRP or VRRP-E virtual MAC address for a VRID, complete the following steps.

1. On the device designated as a VRRP-E device, from privileged EXEC mode, enter configuration mode by issuing the **configure terminal** command.

```
device# configure terminal
```

2. Globally enable the VRRP-E protocol.

```
device(config)# router vrrp-extended
```

3. Configure the ethernet interface link.

```
device(config-vrrpe-router)# interface ethernet 1/5
```

4. Configure the IP address of the interface. All devices configured for the same VRID must be on the same subnet.

```
device(conf-if-e1000-1/5)# ip address 10.53.5.3/24
```

5. Assign the device to VRID 1.

```
device(conf-if-e1000-1/5)# ip vrrp-extended vrid 1
```

NOTE
You can assign a VRID number in the range of 1 through 255.

6. To configure an IPv4 virtual MAC address for VRID 1 (for example), enter the following command at the configure VRID level of the CLI:

```
device(config-if-e1000-1/5-vrid-1)# virtual-mac aaaa.bbbb.cccc
```

NOTE
System-assigned virtual MAC addresses and manually configured virtual MAC addresses can exist at the same time on the device under the same VRID, however the configured value takes precedence. When the configured value is deleted, the assigned value again applies.

7. To display IPv4 VRRP-E virtual MAC address configuration information about VRID 1 (for example), enter the following command:

```
device# show ip vrrp-extended vrid 1

Interface 1/5
----------------
auth-type md5-authentication
VRID 1 (index 1)
interface 1/5
state master
administrative-status disabled
mode non-owner(backup)
virtual mac aaaa.bbbb.cccc (configured)
priority 100
current priority 100
track-priority 5
hello-interval 1 sec
backup hello-interval 60 sec
slow-start timer (configured) 30 sec
advertise backup disabled
dead-interval 0 ms
preempt-mode true
virtual ip address 10.20.1.100
short-path-forwarding disabled
```

The following example configures an IPv4 virtual MAC address for VRID 1 on a VRRP-E device. This example shows all the steps required to configure and activate a VRRP-E device.

```
device# configure terminal
device(config)# router vrrp-extended
device(config)# interface ethernet 1/5
device(config-if-e1000-1/5)# ip address 10.53.5.3/24
device(config-if-e1000-1/5)# ip vrrp-extended vrid 1
device(config-if-e1000-1/5-vrid-1)# backup priority 50 track-priority 10
device(config-if-e1000-1/5-vrid-1)# ip-address 10.53.5.1
device(config-if-e1000-1/5-vrid-1)# virtual-mac aaaa.bbbb.cccc
device(config-if-e1000-1/5-vrid-1)# activate
VRRP-E router 1 for this interface is activating
```

# Suppressing RIP route advertisements on VRRP backup devices

RIP route advertisement suppression can be enabled on VRRP backup devices to prevent other VRRP devices from learning multiple paths for a backed-up interface.

A VRRP or VRRP-E session with master and backup devices must be configured and running.

Normally, a VRRP or VRRP-E backup includes route information for the virtual IP address (the backed-up interface) in RIP advertisements. As a result, other devices receive multiple paths for the backed-up interface and might sometimes unsuccessfully use the path to the backup device rather than the path to the master device.

You can prevent the backups from advertising route information for the backed-up interface by enabling suppression of the advertisements.

NOTE
The command syntax is the same for VRRP and VRRP-E.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enable RIP.

```
device(config)# router rip
```

3. Suppress RIP route advertisements.

```
device(config-rip-router)# use-vrrp-path
```

The following example suppresses RIP advertisements for the backed-up interface.

```
device# configure terminal
device(config)# router rip
device(config-rip-router)# use-vrrp-path
```

# VRRP-Ev2 overview

VRRP Extended (VRRP-E) is an extended version of VRRP. VRRP-E is designed to avoid the limitations in the standards-based VRRP.

VRRP-E is implemented the following differences from RFC 3768 which describes VRRPv2 to provide extended functionality and ease of configuration:

- VRRP-E does not include the concept of an owner device, and a master VRRP-E is determined by the priority configured on the device.
- While the VRRP-E virtual router IP address must belong in the same subnet as a real IP address assigned to a physical interface of the device on which VRRP-E is configured, it must not be the same as any of the actual IP addresses on any interface.
- Configuring VRRP-E uses the same task steps for all devices; there are no differences between master and backup device configuration. The device configured with the highest priority assumes the master role.

  NOTE
  VRRP-E is supported on the devices described in this guide. In a mixed-device environment, consult your documentation for the other devices to determine if VRRP-E is supported.
  VRRP-E does not interoperate with VRRP sessions.

# Enabling a VRRP-E device

This task is performed on any device that is designated as a VRRP extended (VRRP-E) device. For each VRRP-E virtual routing instance, there is one master device and all other devices are backups; but, unlike VRRP, every device is configured as a backup and the device with the highest priority becomes the master VRRP-E device. Repeat this task for all devices that are to be designated as VRRP-E devices.

NOTE
Only VRRP or VRRP-E can be enabled in your network.

1. On the device designated as a VRRP-E device, from privileged EXEC mode, enter global configuration mode by issuing the **configure terminal** command.

```
device# configure terminal
```

2. Globally enable VRRP-E.

```
device(config)# router vrrp-extended
```

3. Configure the Ethernet interface link.

```
device(config-vrrpe-router)# interface ethernet 1/5
```

4. Configure the IP address of the interface. All devices configured for the same virtual router ID (VRID) must be on the same subnet.

```
device(config-if-e1000-1/5)# ip address 10.53.5.3/24
```

5. Assign the device to VRID 1.

```
device(config-if-e1000-1/5)# ip vrrp-extended vrid 1
```

NOTE
You can assign a VRID number in the range of 1 through 255.

6. Designate this router as a backup VRRP device.

```
device(config-if-e1000-1/5-vrid-1)# backup priority 50 track priority 10
```

While configuring a backup device, you can set a priority that is used when a master VRRP device goes offline. The backup device with the highest priority will assume the role of master device.

7. Configure the IP address of the VRID.

```
device(config-if-e1000-1/5-vrid-1)# ip-address 10.53.5.254
```

The IP address associated with the VRID must not be configured on any of the devices used for VRRP-E.

8. Enable the VRRP-E session.

```
device(config-if-e1000-1/5-vrid-1)# activate
VRRP-E router 1 for this interface is activating
```

The following example configures a VRRP-E device.

```
device# configure terminal
device(config)# router vrrp-extended
device(config-vrrpe-router)# interface ethernet 1/5
device(config-if-e1000-1/5)# ip address 10.53.5.3/24
device(config-if-e1000-1/5)# ip vrrp-extended vrid 1
device(config-if-e1000-1/5-vrid-1)# backup priority 50 track-priority 10
device(config-if-e1000-1/5-vrid-1)# ip-address 10.53.5.1
device(config-if-e1000-1/5-vrid-1)# activate
VRRP-E router 1 for this interface is activating
```

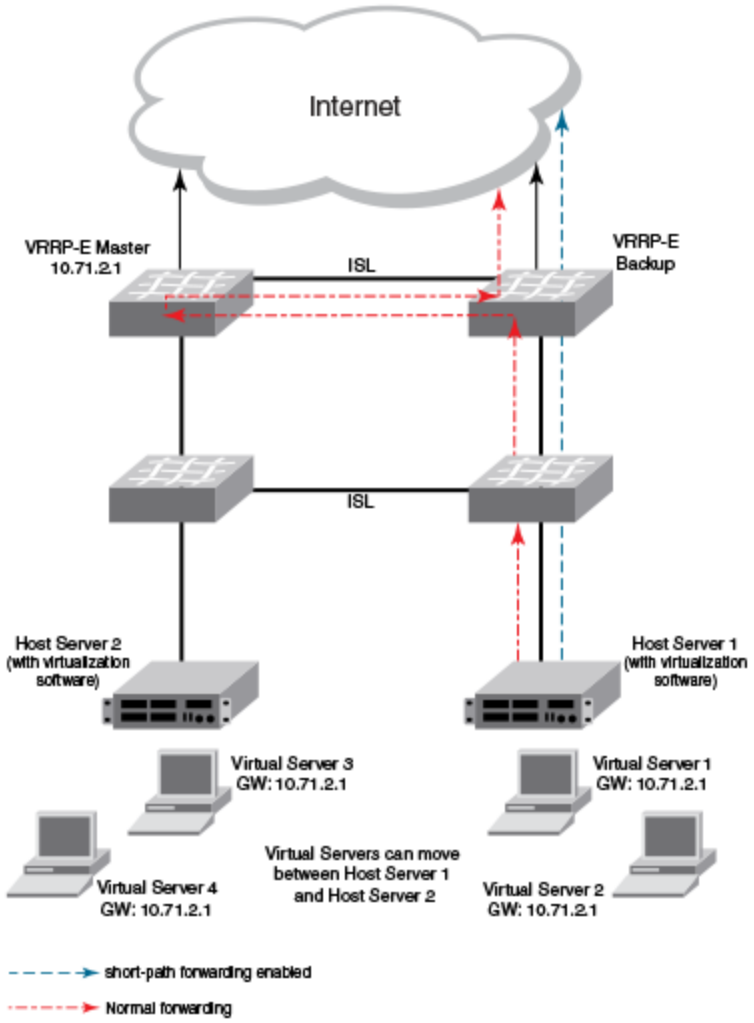
# VRRP-E load-balancing using short-path forwarding

The VRRP-E Extension for Server Virtualization feature allows devices to bypass the VRRP-E master router and directly forward packets to their destination through interfaces on the VRRP-E backup router. This is called *short-path forwarding*. A backup router participates in a VRRP-E session only when short-path forwarding is enabled.

## Packet routing with short-path forwarding to balance traffic load

When short-path forwarding is enabled, traffic load-balancing is performed because both master and backup devices can be used to forward packets.

**FIGURE 68** Short-path forwarding




If you enable short-path forwarding in both master and backup VRRP-E devices, packets sent by Host Server 1 (in the figure) and destined for the Internet cloud through the device on which a VRRP-E backup interface exists can be routed directly to the VRRP-E backup device (blue dotted line) instead of being switched to the master router and then back (red dotted-dash line).

In the figure, load-balancing is achieved using short-path forwarding by dynamically moving the virtual servers between Host Server 1 and Host Server 2.

# Configuring VRRP-E load-balancing using short-path forwarding

VRRP-E traffic can be load-balanced using short-path forwarding on the backup devices.

Before configuring VRRP-E load-balancing, VRRP-E must be configured on all devices in the VRRP-E session.

Perform this task on all backup VRRP-E Layer 3 devices to allow load sharing within a VRRP extended group.

1. Use the **configure terminal** command to enter global configuration mode.

   ```
   device# configure terminal
   ```

2. To globally enable VRRP-E, enter the **router vrrp-extended** command.

   ```
   device(config)# router vrrp-extended
   ```

3. Enter the **interface ve** command with an associated VLAN number.

   ```
   device(config-vrrpe-router)# interface ve 10
   ```

   In this example, virtual Ethernet (ve) configuration mode is entered and the interface is assigned a VLAN number of 10.

4. Enter an IP address for the interface using the **ip address** command.

   ```
   device(config-vif-10)# ip address 192.168.4.1/24
   ```

5. Enter the **ip vrrp-extended vrid** command with a number to assign a VRRP-E virtual router ID to the device.

   ```
   device(config-vif-10)# ip vrrp-extended vrid 5
   ```

   In this example, VRRP-E group configuration mode is entered.

6. Enter the **backup** command with a **priority** value to configure the device as a VRRP-E backup device.

   ```
   device(config-vif-10-vrid-5)# backup priority 50
   ```

7. Enter the **ip-address** command with an IP address that is not used on any VRRP-E device interface to add a virtual IP address to the VRRP-E instance.

   ```
   device(config-vif-10-vrid-5)# ip-address 192.168.4.254
   ```

8. Enter the **short-path-forwarding** command with a **revert-priority** value to configure the backup VRRP-E device as an alternate path with a specified priority.

   ```
   device(config-vif-10-vrid-5)# short-path-forwarding revert-priority 50
   ```

   When the backup device priority is higher than the configured **revert-priority** value, the backup router is able to perform short-path forwarding. If the backup priority is lower than the revert priority, short-path forwarding is disabled.

9. Enter the **activate** command to activate the VRRP-E instance.

   ```
   device(config-vif-10-vrid-5)# activate
   ```

In the following example, short-path forwarding is configured on a backup VRRP-E device, and a revert priority threshold is configured. If the backup device priority falls below this threshold, short-path forwarding is disabled.

```
device# configure terminal
device(config)# router vrrp-extended
device(config-vrrpe-router)# interface ve 10
device(config-vif-10)# ip address 192.168.4.1/24
device(config-vif-10)# ip vrrp-extended vrid 5
device(config-vif-10-vrid-5)# backup priority 50
device(config-vif-10-vrid-5)# ip-address 192.168.4.254
device(config-vif-10-vrid-5)# short-path-forwarding revert-priority 50
device(config-vif-10-vrid-5)# activate
```

# VRRP-E slow start timer

In a VRRP extended (VRRP-E) configuration, if a master device goes offline, the backup router with the highest priority takes over after the expiration of the dead interval timer. When the original master device is back online, you can configure a slow-start timer interval that extends the time interval beyond the dead interval before the original master device transitions back to the role of master device.

The slow-start interval allows additional time for routing protocols, for example OSPF, to converge without causing route flapping during the transition from backup device to master device. Included in the VRRP-E slow-start timer feature are track port state changes and restart options. The **use-track-port** option implements a slow-start timer for the first tracked port "up" state change, in addition to the VRRP-E initialization state. The **restart** option restarts the slow-start timer for subsequent tracked port "up" state changes.

> NOTE
> If you change the backup priority of a VRRP-E backup router to be higher than the priority of the original master device, the slow-start timer will not work. The original master device will take over from the backup device immediately.

## Configuring a VRRP-E slow-start timer

The slow-start timer is a VRRP-E interval timer that extends beyond the dead interval during a transition from the backup device that assumed the master role to the original master device that is back online and has a higher priority.

In a VRRP extended (VRRP-E) configuration, if a master device goes offline, the backup router with the highest priority takes over after the expiration of the dead interval timer. When the original master device is back online, you can configure a slow-start timer interval that extends the time interval beyond the dead interval before the original master device transitions back to the role of master device.

1. Use the **configure terminal** command to enter global configuration mode.

   ```
   device# configure terminal
   ```

2. To globally enable VRRP-E, enter the **router vrrp-extended** command.

   ```
   device(config)# router vrrp-extended
   ```

3. Enter the **slow-start** command with options to configure the interval, in seconds, and whether tracked-port state changes trigger the slow-start interval.

   ```
   device(config-vrrpe-router)# slow-start 40 use-track-port restart
   ```

In this example, the slow-start timer interval is set to 40 seconds, and the slow-start timer also runs after the first and subsequent tracked-port state changes.

```
device# configure terminal
device(config)# router vrrp-extended
device(config-vrrpe-router)# slow-start 40 use-track-port restart
```

# Multiple virtual IP address support for VRRP-E

Support for multiple virtual IPv4 addresses per interface is available for a VRRP extended (VRRP-E) router instance. For an interface that has multiple subnet IPv4 addresses, a single VRRP-E provides redundancy when all the IP addresses are configured for the VRRP-E instance.

All virtual IP addresses must be configured before activating the VRRP-E instance. If a virtual IP address is to be added, the VRRP-E router instance must be disabled, and the configuration changes made before reactivating the VRRP-E instance. Between master and backup VRRP-E devices the virtual IP address configuration must be consistent to allow the generation of the same virtual MAC address for a specific virtual router ID (VRID). Gratuitous ARP messages are sent for all the configured IP addresses of each VRRP-E instance. To avoid regeneration of the virtual MAC address when a lower value virtual IPv4 address is configured, configure a static virtual MAC address for the VRRP-E router instance.

NOTE
Multiple virtual IPv4 address support is not compatible with the VRRP-E scaling feature.

## Configuring multiple virtual IP addresses for VRRP-E

Configuring multiple virtual IPv4 addresses for an interface assigned as a VRRP extended (VRRP-E) router instance.

For an interface that has multiple subnet IPv4 addresses, a single VRRP-E provides redundancy when all the IP addresses are configured for the VRRP-E instance. In this task, configuring a static MAC address is an optional step.

1. From privileged EXEC mode, enter configuration mode by issuing the **configure terminal** command.

```
device# configure terminal
```

2. Globally enable the VRRP-E protocol by entering the **router vrrp-extended** command.

```
device(config)# router vrrp-extended
```

3. Configure the ethernet interface for the VRRP-E instance.

```
device(config)# interface ethernet 1/1
```

4. Configure an IP address on the interface.

```
device(config-if-e1000-1/1)# ip address 10.10.10.1/24
```

5. Repeat Step 4 for all IP addresses on the interface.

```
device(config-if-e1000-1/1)# ip address 10.20.20.1/24
```

6. Configure a VRRP extended instance using a virtual routing ID (VRID).

```
device(config-if-e1000-1/1)# ip vrrp-extended vrid 2
```

7. Configure the device as a backup VRRP-E device for VRID 2.

```
device(config-if-e1000-1/1-vrid-2)# backup
```

8. Configure an optional static MAC address for VRID 2.

```
device(config-if-e1000-1/1-vrid-2)# virtual-mac aaaa.bbbb.cccc
```

9. Configure a virtual IP address for VRID 2.

```
device(config-if-e1000-1/1-vrid-2)# ip-address 10.10.10.10
```

10. Repeat Step 9 to configure all the virtual IP addresses for VRID 2.

```
device(config-if-e1000-1/1-vrid-2)# ip-address 10.20.20.20
```

    If you want to add a new virtual IP address after the initial setup, you must first deactivate the virtual router and then reconfigure all the virtual IP addresses.

11. Activate the virtual IP address or addresses for VRID 2.

```
device(config-if-e1000-1/1-vrid-2)# activate
```

The following example configures seven virtual IP addresses for VRRP-E virtual router instance 2.

```
device# configure terminal
device(config)# router vrrp-extended
device(config)# interface ethernet 1/1
device(config-if-e1000-1/1)# ip address 10.10.10.1/24
device(config-if-e1000-1/1)# ip address 10.20.20.1/24
device(config-if-e1000-1/1)# ip address 10.30.30.1/24
device(config-if-e1000-1/1)# ip address 10.40.40.1/24
device(config-if-e1000-1/1)# ip address 10.50.50.1/24
device(config-if-e1000-1/1)# ip address 10.60.60.1/24
device(config-if-e1000-1/1)# ip address 10.70.70.1/24
device(config-if-e1000-1/1)# ip vrrp-extended vrid 2
device(config-if-e1000-1/1-vrid-2)# backup
device(config-if-e1000-1/1-vrid-2)# virtual-mac aaaa.bbbb.cccc
device(config-if-e1000-1/1-vrid-2)# ip-address 10.10.10.10
device(config-if-e1000-1/1-vrid-2)# ip-address 10.20.20.20
device(config-if-e1000-1/1-vrid-2)# ip-address 10.30.30.30
device(config-if-e1000-1/1-vrid-2)# ip-address 10.40.40.40
device(config-if-e1000-1/1-vrid-2)# ip-address 10.50.50.50
device(config-if-e1000-1/1-vrid-2)# ip-address 10.60.60.60
device(config-if-e1000-1/1-vrid-2)# ip-address 10.70.70.70
device(config-if-e1000-1/1-vrid-2)# activate
```

# Displaying multiple virtual IP addresses for VRRP-E information

Displays information about the multiple virtual IPv4 addresses that are configured for a VRRP-E instance.

Several options of the **show ip vrrp-extended** command can display information about the multiple virtual IP addresses configured for a VRRP-E instance. Use the steps below in any order.

1. Enter the **show ip vrrp-extended brief** command to display the number of configured virtual IPv4 addresses for each VRRP-E router instance and the virtual IPv4 addresses.

```
device# show ip vrrp-extended brief

Total number of VRRP-Extended routers defined: 3
Flags Codes - P:Preempt 2:V2 3:V3
Short-Path-Fwd Codes - ER: Enabled with revertible option, RT: reverted,
                       NR: not reverted

Intf VRID Curr Flags State  MasterIP BackupIP (No) VirtualIP Short-   Track MCT
          Prio              Address  Address       Address   Path-Fwd VPLS-State
-------------------------------------------------------------------------------
1/1  1    100  P2    Master Local    Unknown  ( 7) 10.10.10.10 Enabled Disable
                                                  10.20.20.20
                                                  10.30.30.30
                                                  10.40.40.40
                                                  10.50.50.50
                                                  10.60.60.60
                                                  10.70.70.70
```

2. Enter the **show ip vrrp-extended** command with a specific VRID and interface to display the number of configured IPv4 addresses for each VRRP-E router instance and the actual IPv4 addresses in a different format with the detailed information about each virtual router instance.

```
device# show ip vrrp-extended vrid 1 ethernet 1/1

Interface 1/1
----------------
 auth-type no authentication
 VRID 1 (index 1)
 interface 1/1
 state master
 administrative-status enabled
 mode non-owner(backup)
 virtual mac 02e0.527d.7c01
 priority 100
 current priority 100
 track-priority 5
 hello-interval 1 sec
 backup hello-interval 60 sec
 slow-start timer (configured) 0 sec
 advertise backup disabled
 dead-interval 3500 ms
 preempt-mode true
 number of configured virtual address 7
 virtual ip address 10.10.10.10 10.20.20.20 10.30.30.30 10.40.40.40
                    10.50.50.50 10.60.60.60 10.70.70.70
 next hello sent in 1000 ms
 Track MCT-VPLS-State: Disable
 short-path-forwarding enabled
```

# VRRP-E scaling using logical groups

Scaling the number of VRRP extended (VRRP-E) instances up to 4000 instances is allowed using a grouping mechanism. VRRP-E instances are configured into logical groups consistently across all the VRRP-E master and backup devices.

Each VRRP-E logical group is assigned with a group master and the configuration is performed to identify the group master for each VRRP-E virtual router ID (VRID) configured on a device. Group members stop advertising hello messages and they inherit the state of the group master. The significant reduction in load on the CPU when the hello advertisements are not sent or processed, allows more VRRP-E instances to be configured.

A gratuitous ARP router advertisement is sent from the group-master on behalf of its group members once every 30 seconds by default to allow the devices on the network to learn the virtual MAC address of the group master. The gratuitous ARP router advertisement interval can be configured.

> NOTE
> If the maximum number of VRRP-E instances exist, configuring a gratuitous ARP router advertisement interval lower than 5 seconds can result in high CPU usage.

The configuration for the following commands is inherited from the group master device configuration. VRRP-E ignores the configuration of any parameter that is to be inherited from the group master.

- advertise
- auth-type
- backup
- backup-hello-interval
- dead-interval
- hello-interval
- non-preempt-mode
- use-v2-checksum

The configuration for the following commands is not inherited from the group master. Group members retain their individual configuration parameters from these commands.

- activate
- enable
- ip-address
- ipv6-address
- short-path-forwarding
- virtual-mac

> NOTE
> The VRRP-E scaling feature is not compatible with the VRRP-E multiple virtual IP addresses feature.

# Configuring VRRP-E scaling

Configuring VRRP-E instances into logical groups using a group master and configuring a gratuitous ARP router advertisement interval allows the number of VRRP-E instances to scale up to 4000 instances.

This task has to be repeated for all VRRP-E instances that are to be part of the same group. VRRP-E instances are always physically grouped on the same device and the logical group is maintained consistently across all devices. VRRP-E scaling is supported for IPv4 and IPv6 VRRP-E sessions.

NOTE
The VRRP-E scaling feature is not compatible with the VRRP-E multiple virtual IP addresses feature.

1. From privileged EXEC mode, enter configuration mode by issuing the **configure terminal** command.

```
device# configure terminal
```

2. Globally enable the IPv6 VRRP-E protocol by entering the **ipv6 router vrrp-extended** command.

```
device(config)# ipv6 router vrrp-extended
```

3. Configure the gratuitous ARP interval for the IPv6 VRRP-E instance.

```
device(config-ipv6-VRRP-E-router)# garp-ra-interval 90
```

NOTE
In IPv4 VRRP-E the router prompt does not display the "-VRRP-E-router" text.

4. Configure the ethernet interface for the IPv6 VRRP-E instance.

```
device(config-ipv6-VRRP-E-router)# interface ethernet 1/5
```

5. Configure an IPv6 address on the interface.

```
device(config-if-e1000-1/5)# ipv6 address 3013::2/64
```

6. Configure an IPv6 VRRP extended instance using a virtual routing ID (VRID).

```
device(config-if-e1000-1/5)# ipv6 vrrp-extended vrid 2
```

7. Configure the device as a backup VRRP-E device for VRID 2.

```
device(config-if-e1000-1/5-ipv6-vrid-2)# backup priority 50 track-priority 20
```

8. Configure a virtual link-local IPv6 address for VRID 2.

```
device(config-if-e1000-1/5-ipv6-vrid-2)# ipv6-address fe80::768e:f8ff:fe2a:0099
```

9. Configure a virtual IPv6 address for VRID 2.

```
device(config-if-e1000-1/5-ipv6-vrid-2)# ipv6-address 3013::99
```

10. Configures the VRRP-E master for a group for VRID 2.

```
device(config-if-e1000-1/5-ipv6-vrid-2)# group-master interface ethernet 1/2 vrid 1
```

In this example,

11. Activate the virtual IP address or addresses for VRID 2.

```
device(config-if-e1000-1/5-ipv6-vrid-2)# activate
```

The following example configures virtual router 1 on interface Ethernet 1/2 as the VRRP-E group master of virtual router 2 on interface Ethernet 1/5. Virtual router 2 will stop sending and receiving VRRP-E packets because VRID 1 now maintains the session for virtual router 2 by sending and receiving VRRP-E packets. The interval between gratuitous ARP messages is set to 90 seconds.

```
device# configure terminal
device(config)# router ipv6 vrrp-extended
device(config-ipv6-VRRP-E-router)# garp-ra-interval 90
device(config-ipv6-VRRP-E-router)# interface ethernet 1/5
device(config-if-e1000-1/5)# ipv6 address 3013::2/64
device(config-if-e1000-1/5)# ipv6 vrrp-extended vrid 2
device(config-if-e1000-1/5-ipv6-vrid-2)# backup priority 50 track-priority 20
device(config-if-e1000-1/5-ipv6-vrid-2)# ipv6-address fe80::768e:f8ff:fe2a:0099
device(config-if-e1000-1/5-ipv6-vrid-2)# ipv6-address 3013::99
device(config-if-e1000-1/5-ipv6-vrid-2)# group-master interface ethernet 1/2 vrid 1
device(config-if-e1000-1/5-ipv6-vrid-2)# activate
```

## Displaying VRRP-E scaling information

Displays information about VRRP-E group members and group masters configured for the VRRP-E scaling feature.

Several options of the **show ip vrrp-extended** or **show ipv6 vrrp-extended** commands can display information about the VRRP-E scaling feature configuration. Use the steps below in any order.

1. Enter the **show ip vrrp-extended** or the **show ipv6 vrrp-extended** command with the **vrid** option to display group member information for the VRRP-E scaling feature for a specific VRID. In the example below, output for VRID 1 shows the total number of group members and in which VRIDs the members are configured.

```
device(config)# show ip vrrp-extended vrid 1

VRID 1 (index 1)
 interface 1/1
 state master
. administrative-status enabled
.
.
 group-member count 3
 group-members
   ethernet 1/2 vrid 2
   ethernet 1/2 vrid 3
   ethernet 1/2 vrid 4
```

2. Enter the **show ipv6 vrrp-extended** or the **show ip vrrp-extended** command with a specific VRID and interface to display group master information for the VRRP-E scaling feature for a specific interface. Only partial output is displayed.

```
device# show ipv6 vrrp-extended ve 100 vrid 2

VRID 2 (index 2)
 interface v100
 state backup
.
.
.
 group-master ve 100 vrid 1
```

# Displaying VRRPv2 information

Various show commands can be used to display statistical and summary information about VRRP and VRRP-E configurations.

Before displaying VRRP information, VRRPv2 must be configured and enabled in your VRRP or VRRP-E network to generate traffic.

Use one or more of the following commands to display VRRPv2 information. The commands do not have to be entered in this order.

1. Enter the **show ip vrrp** command with the **vrid** option and a virtual router ID (VRID) to display IPv4 VRRP configuration information about VRID 1.

```
device# show ip vrrp vrid 1

Interface 1/1
----------------
auth-type no authentication
VRID 1 (index 1)
interface 1/1
state master
administrative-status enabled
version v2
mode owner
virtual mac aaaa.bbbb.cccc (configured)
priority 255
current priority 255
track-priority 2
hello-interval 1 sec
backup hello-interval 6
```

2. Enter the **show ip vrrp brief** command.

```
device(config)# show ip vrrp brief

Total number of VRRP routers defined: 2
Flags Codes - P:Preempt 2:V2 3:V3 S:Short-Path-Fwd
Inte- VRID  Current  Flags    State   Master IP Backup IP  Virtual IP
rface       Priority                  Address   Address    Address
------------------------------------------------------------------------
1/1   10    255      P2-      Master  Local     Unknown    10.30.30.2
1/3   13    100      P2-      Master  Local     Unknown    10.13.13.3
```

   This example displays summary information about VRRP sessions.

3. Enter the **show ip vrrp-extended statistics** command for Ethernet interface 1/5.

```
device# show ip vrrp-extended interface Ethernet 1/5

Interface 1/5
----------------
VRID 2
- number of transitions to backup state = 1
- number of transitions to master state = 1
- total number of vrrp-extended packets received = 0
  . received backup advertisements = 0
  . received packets with zero priority = 0
  . received packets with invalid type = 0
  . received packets with invalid authentication type = 0
  . received packets with authentication type mismatch = 0
  . received packets with authentication failures = 0
  . received packets dropped by owner = 0
  . received packets with ttl errors = 0
  . received packets with ipv6 address mismatch = 0
  . received packets with advertisement interval mismatch = 0
  . received packets with invalid length = 0
- total number of vrrp-extended packets sent = 2004
  . sent backup advertisements = 0
  . sent packets with zero priority = 0
- received neighbor solicitation packets dropped = 0
- received proxy neighbor solicitation packets dropped = 0
- received ip packets dropped = 0
```

# Clearing VRRPv2 statistics

VRRPv2 session counters can be cleared using a CLI command.

Ensure that VRRPv2 or VRRP-Ev2 is configured and enabled in your network.

To determine the effect of clearing the VRRP statistics, an appropriate **show** command is entered before and after the **clear** command.

1. Enter the **end** or **exit** command to return to privileged EXEC mode.
2. Enter the **show ip vrrp statistics** command for Ethernet interface 1/5.

```
device# show ip vrrp statistics ethernet 1/5

Interface 1/5
----------------
VRID 2
- number of transitions to backup state = 1
- number of transitions to master state = 1
- total number of vrrp packets received = 0
  . received backup advertisements = 0
  . received packets with zero priority = 0
.
.
.
- total number of vrrp packets sent = 2004
  . sent backup advertisements = 6
  . sent packets with zero priority = 0
- received neighbor solicitation packets dropped = 0
```

3. Enter the **clear ip vrrp statistics** command.

```
device# clear ip vrrp statistics
```

4. Enter the **show ip vrrp statistics** command for Ethernet interface 1/5.

```
device# show ip vrrp statistics ethernet 1/5

Interface 1/5
----------------
VRID 2
- number of transitions to backup state = 0
- number of transitions to master state = 0
- total number of vrrp packets received = 0
  . received backup advertisements = 0
  . received packets with zero priority = 0
.
.
.
- total number of vrrp packets sent = 6
  . sent backup advertisements = 0
  . sent packets with zero priority = 0
- received neighbor solicitation packets dropped = 0
```

In this show output for a specified interface after the **clear ip vrrp statistics** command has been entered, you can see that the statistical counters have been reset. Although some of the counters are showing numbers because VRRP traffic is still flowing, the numbers are much lower (8 transmissions instead of 2004 transmissions) than in the initial **show ip vrrp statistics** command output.

# VRRPv3



# VRRPv3 overview

VRRP version 3 (VRRPv3) introduces IPv6 address support for both standard VRRP and VRRP enhanced (VRRP-E).

Virtual Router Redundancy Protocol (VRRP) is designed to eliminate the single point of failure inherent in a static default routed environment by providing redundancy to Layer 3 devices within a local area network (LAN). VRRP uses an election protocol to dynamically assign the default gateway for a host to one of a group of VRRP routers on a LAN. Alternate gateway router paths can be allocated without changing the IP address or MAC address by which the host device knows its gateway.

VRRPv3 implements support for IPv6 addresses for networks using IPv6, and it also supports IPv4 addresses for dual-stack networks configured with VRRP or VRRP-E. VRRPv3 is compliant with RFC 5798. The benefit of implementing VRRPv3 is faster switchover to backup devices than can be achieved using standard IPv6 neighbor discovery mechanisms. With VRRPv3, a backup router can become a master router in a few seconds with less overhead traffic and no interaction with the hosts.

When VRRPv3 is configured, the master device that owns the virtual IP address and a master device that does not own the virtual IP address can both respond to ICMP echo requests (using the **ping** command) and accept Telnet and other management traffic sent to the virtual IP address. In VRRPv2, only a master device on which the virtual IP address is the address of an interface on the master device can respond to ping and other management traffic.

The following are other IPv6 VRRPv3 functionality details:

- VRRPv2 functionality is supported by VRRPv3 except for VRRP authentication.
- Two VRRP and VRRP-E sessions cannot share the same group ID on the same interface.

NOTE

When implementing IPv6 VRRPv3 across a network with devices from other vendors, be aware of a potential interoperability issue with IPv6 VRRPv3 and other vendor equipment. Extreme has implemented IPv6 VRRPv3 functionality to comply with RFC 5798 and will interoperate comfortably with other vendors that support RFC 5798.

# Enabling an IPv6 VRRPv3 owner device

This task is performed on the device that is designated as the owner VRRP device because the IPv6 address of one of its physical interfaces is assigned as the IP address of the virtual router. For each VRRP session, there are master and backup routers, and the owner router is elected, by default, as the master router.

NOTE
When implementing IPv6 VRRPv3 across a network with devices from other vendors, be aware of a potential interoperability issue. IPv6 VRRPv3 functionality is implemented to comply with RFC 5798 and will interoperate well with other vendors that support RFC 5798.

1. On the device designated as the owner VRRP device, from privileged EXEC mode, enter global configuration mode by issuing the **configure terminal** command.

```
device# configure terminal
```

2. Before enabling IPv6 VRRP, you must globally enable IPv6 routing.

```
device(config)# ipv6 unicast-routing
```

3. Globally enable IPv6 VRRP.

```
device(config)# ipv6 router vrrp
```

4. Configure the Ethernet interface link for the owner device.

```
device(config)# interface ethernet 1/5
```

5. Configure the IPv6 address of the interface.

```
device(config-if-e1000-1/5)# ipv6 address fd2b::2/64
```

6. Assign the owner device to the virtual router ID (VRID) 2.

```
device(config-if-e1000-1/5)# ipv6 vrrp vrid 2
```

NOTE
You can assign a VRID number in the range of 1 through 255.

7. Designate this router as the VRRP owner device.

```
device(config-if-e1000-1/5-vrid-2)# owner
```

8. Assign an IPv6 link-local address to the VRID for use in the local network.

```
device(config-if-e1000-1/5-vrid-2)# ipv6-address fe80::768e:f8ff:fe2a:0099
```

9. Assign a global IPv6 address to the VRID.

```
device(config-if-e1000-1/5-vrid-2)# ipv6-address fd2b::2
```

10. Enable the VRRP session.

```
device(config-if-e1000-1/5-vrid-2)# activate
```

The following example configures a VRRP owner device.

```
device# configure terminal
device(config)# ipv6 unicast-routing
device(config)# ipv6 router vrrp
device(config-ipv6-vrrp-router)# interface ethernet 1/5
device(config-if-e1000-1/5)# ipv6 address fd2b::2/64
device(config-if-e1000-1/5)# ipv6 vrrp vrid 2
device(config-if-e1000-1/5-vrid-2)# owner
device(config-if-e1000-1/5-vrid-2)# ipv6-address fe80::768e:f8ff:fe2a:0099
device(config-if-e1000-1/5-vrid-2)# ipv6-address fd2b::2
device(config-if-e1000-1/5-vrid-2)# activate
```

# Enabling an IPv6 VRRPv3 backup device

This task is performed on all devices that are designated as backup VRRPv3 devices. Initially a backup priority is set to 100. For each VRRPv3 session, there are master and backup routers, and the IPv6 address assigned here to the VRID is the IPv6 address of the master router. The task is repeated on each backup VRRPv3 device with corresponding changes to the interface number and IPv6 address of the interface.

NOTE
When implementing IPv6 VRRPv3 across a network with devices from other vendors, be aware of a potential interoperability issue. IPv6 VRRPv3 functionality is implemented to comply with RFC 5798 and will interoperate well with other vendors that support RFC 5798.

1. On the device designated as a backup VRRPv3 device, from privileged EXEC mode, enter global configuration mode by issuing the **configure terminal** command.

   ```
   device# configure terminal
   ```

2. Globally enable IPv6 VRRP.

   ```
   device(config)# ipv6 router vrrp
   ```

3. Configure the Ethernet interface link for the owner device.

   ```
   device(config-ipv6-vrrp-router)# interface ethernet 1/4
   ```

4. Configure the IPv6 address of the interface.

   ```
   device(config-if-e1000-1/4)# ipv6 address fd2b::3/64
   ```

5. Assign the backup device to the virtual router ID (VRID) 2.

   ```
   device(config-if-e1000-1/4)# ipv6 vrrp vrid 2
   ```

   NOTE
   You can assign a VRID number in the range of 1 through 255.

6. Designate this router as a VRRPv3 backup device and assign it a priority of 100.

   ```
   device(config-if-e1000-1/4-vrid-2)# backup priority 100
   ```

7. By default, backup VRRP devices do not send hello messages to advertise themselves to the master. Use the following command to enable a backup router to send hello messages to the master VRRP device.

   ```
   device(config-if-e1000-1/4-vrid-2)# advertise backup
   ```

8. Assign the IPv6 link-local address to the VRID for use in the local network.

```
device(config-if-e1000-1/4-vrid-2)# ipv6-address fe80::768e:f8ff:fe2a:0099
```

9. Assign the global IPv6 address to the VRID.

```
device(config-if-e1000-1/4-vrid-2)# ipv6-address fd2b::2
```

10. Enable the VRRP session.

```
device(config-if-e1000-1/4-vrid-2)# activate
```

The following example configures an IPv6 VRRPv3 backup device.

```
device# configure terminal
device(config)# ipv6 router vrrp
device(config-ipv6-vrrp-router)# interface ethernet 1/4
device(config-if-e1000-1/4)# ipv6 address fd2b::3/64
device(config-if-e1000-1/4)# ipv6 vrrp vrid 2
device(config-if-e1000-1/4-vrid-2)# backup priority 100
device(config-if-e1000-1/4-vrid-2)# advertise backup
device(config-if-e1000-1/4-vrid-2)# ipv6-address fe80::768e:f8ff:fe2a:0099
device(config-if-e1000-1/4-vrid-2)# ipv6-address fd2b::2
device(config-if-e1000-1/4-vrid-2)# activate
```

# Enabling an IPv4 VRRPv3 owner device

VRRPv3 supports IPv4 sessions as well as IPv6 sessions. To configure a VRRPv3 session for IPv4, assign a virtual router group with the VRRP version set to 3. This task is performed on the device that is designated as the owner VRRP device because the IP address of one of its physical interfaces is assigned as the IP address of the virtual router.

1. On the device designated as the owner VRRP device, from privileged EXEC mode, enter global configuration mode by issuing the **configure terminal** command.

```
device# configure terminal
```

2. Globally enable VRRP.

```
device(config)# router vrrp
```

3. Configure an Ethernet interface.

```
device(config)# interface ethernet 1/6
```

4. Configure the IP address of the interface.

```
device(config-if-e1000-1/6)# ip address 10.53.5.1/24
```

5. Assign the virtual router ID (VRID) 1 to the interface.

```
device(config-if-e1000-1/6)# ip vrrp vrid 1
```

NOTE
You can assign a VRID number in the range of 1 through 255.

6. Designate this router as the VRRP owner device.

```
device(config-if-e1000-1/6-vrid-1)# owner
```

7. Configure the IP address of the VRID.

```
device(config-if-e1000-1/6-vrid-1)# ip-address 10.53.5.1
```

8. Enable the VRRP session.

```
device(config-if-e1000-1/6-vrid-1)# activate
```

The following example configures an IPv4 VRRPv3 owner device.

```
device# configure terminal
device(config)# router vrrp
device(config)# interface ethernet 1/6
device(config-if-e1000-1/6)# ip address 10.53.5.1/24
device(config-if-e1000-1/6)# ip vrrp vrid 1
device(config-if-e1000-1/6-vrid-1)# owner
device(config-if-e1000-1/6-vrid-1)# ip-address 10.53.5.1
device(config-if-e1000-1/6-vrid-1)# activate
VRRP router 1 for this interface is activating
```

# Enabling an IPv4 VRRPv3 backup device

VRRPv3 supports IPv4 sessions as well as IPv6 sessions. To configure a VRRPv3 session for IPv4, assign a virtual router group with the VRRP version set to 3. This task is performed on any device that is designated as an IPv4 backup VRRPv3 device. For each VRRP virtual routing instance, there is one master device and all other devices are backups. Repeat this task on all devices that are to be designated as backup devices.

1. On a device designated as a backup VRRP device, from privileged EXEC mode, enter global configuration mode by issuing the **configure terminal** command.

```
device# configure terminal
```

2. Globally enable VRRP.

```
device(config)# router vrrp
```

3. Configure the Ethernet interface.

```
device(config)# interface ethernet 1/5
```

4. Configure the IP address of the interface. All devices configured for the same virtual router ID (VRID) must be on the same subnet.

```
device(config-if-e1000-1/5)# ip address 10.53.5.3/24
```

5. Assign the same VRID as the VRID used by the owner device.

```
device(config-if-e1000-1/5)# ip vrrp vrid 1
```

   NOTE
   You can assign a VRID number in the range of 1 through 255.

6. Designate this router as a backup VRRP device.

```
device(config-if-e1000-1/5-vrid-1)# backup priority 110
```

   While configuring a backup device, you can set a priority that is used when a master VRRP device goes offline. The backup device with the highest priority will assume the role of master device.

7. Configure the IP address of the VRID.

```
device(config-if-e1000-1/5-vrid-1)# ip-address 10.53.5.1
```

   The VRID IP address is the same virtual IP address that you used for the VRRP owner device.

8. Enable the VRRP session.

```
device(config-if-e1000-1/5-vrid-1)# activate
VRRP router 1 for this interface is activating
```

The following example configures a VRRP owner device.

```
device# configure terminal
device(config)# router vrrp
device(config)# interface ethernet 1/5
device(config-if-e1000-1/5)# ip address 10.53.5.3/24
device(config-if-e1000-1/5)# ip vrrp vrid 1
device(config-if-e1000-1/5-vrid-1)# backup priority 110
device(config-if-e1000-1/5-vrid-1)# ip-address 10.53.5.1
device(config-if-e1000-1/5-vrid-1)# activate
VRRP router 1 for this interface is activating
```

# Tracked ports and track priority with VRRP and VRRP-E

Port tracking allows interfaces not configured for VRRP or VRRP-E to be monitored for link-state changes that can result in dynamic changes to the VRRP device priority.

A tracked port allows you to monitor the state of the interfaces on the other end of a route path. A tracked interface also allows the virtual router to lower its priority if the exit path interface goes down, allowing another virtual router in the same VRRP (or VRRP-E) group to take over. When a tracked interface returns to an up state, the configured track priority is added to the current virtual router priority value. The following conditions and limitations exist for tracked ports:

- Track priorities must be lower than VRRP or VRRP-E priorities.
- The dynamic change of router priority can trigger a master device switchover if preemption is enabled. However, if the router is an owner, the master device switchover will not occur.
- The maximum number of interfaces that can be tracked for a virtual router is 16.
- Port tracking is allowed for physical interfaces and port channels.

## Tracking ports and setting VRRP priority using VRRPv3

Configuring port tracking on an exit path interface and setting a priority on a VRRPv3 device enables VRRPv3 to monitor the interface. For VRRPv3, if the interface goes down, the device priority is set to the priority value and another backup device with a higher priority assumes the role of master. For VRRP-Ev3, if the interface goes down, the device priority is lowered by the priority value and another backup device with a higher priority assumes the role of master.

Before enabling IPv6 VRRPv3, you must globally enable IPv6 routing using the **ipv6 unicast-routing** command.

Configure this task on the device on which the tracked interface exists.

NOTE
Only IPv4/IPv6 IPsec tunnels can be configured with a specific **track-port** command priority. Other interfaces use the track priority associated with the **owner** or **backup** commands.

1. Enter the **configure terminal** command to enter global configuration mode.

```
device# configure terminal
```

2. Enter the **ipv6 router vrrp** command to configure VRRPv3 globally.

```
device(config)# ipv6 router vrrp
```

3. Configure the Ethernet interface.

```
device(config)# interface ethernet 1/6
```

4. Enter the IPv6 address for the interface to be used for the virtual router ID (VRID).

```
device(config-if-e10000-1/6)# ipv6 address fd2b::2/64
```

5. Enter the following command to enter the appropriate VRRPv3 virtual router ID (VRID) mode.

```
device(config-if-e10000-1/6)# ipv6 vrrp vrid 1
```

6. Enter the **track-port** command to set the tracked port and priority:

```
device(config-if-e10000-1/6-vrid-1)# track-port tunnel 1 priority 20
```

The priority value is used when a tracked port goes down and the new priority is set to this value. Ensure that the priority value is lower than the priority set for any existing master or backup device to force a renegotiation for the master device.

The following example shows how to configure tunnel interface 1 on virtual router 1 to be tracked; if the interface fails, the IPv6 VRRPv3 priority of the device becomes 20, forcing a negotiation for a new master device.

```
device# configure terminal
device(config)# ipv6 router vrrp
device(config)# interface ethernet 1/6
device(config-if-e10000-1/6)# ipv6 address fd2b::2/64
device(config-if-e10000-1/6)# ipv6 vrrp vrid 1
device(config-if-e10000-1/6-vrid-1)# track-port tunnel 1 priority 20
```

# Tracked IPsec tunnels and track priority with VRRP and VRRP-E

IPsec tunnel tracking using VRRP or VRRP Extended (VRRP-E) is helpful in monitoring network reachability changes that can result in dynamic changes to the master VRRP or VRRP-E device priority.

Business-critical network traffic requires security options to be configured in the network. VRRP or VRRP-E can be implemented together with IP security (IPsec) to provide redundancy and scalablity. The tunnels to be tracked are outgoing interfaces for the device on which they are configured. If any of the tracked tunnels goes down, VRRP or VRRP-E can quickly assign a new master device to minimize any loss of packets through the network. For VRRP, if the tracked tunnel goes down, the current router priority is reduced to the

priority set for the tracked tunnel resulting in renegotiation for the master device. For VRRP-E, if the tracked tunnel goes down, the current router priority is reduced by the priority set for the tracked tunnel resulting in renegotiation for the master device. When a tracked tunnel returns to an up state, the priority of the VRRP or VRRP-E device is restored to the original value to force a renegotiation for the master device. The following conditions and limitations exist for tracked tunnels:

- Track priorities must be lower than VRRP or VRRP-E priorities.
- The dynamic change of router priority can trigger a master device switchover if preemption is enabled and if the owner router goes down.
- The maximum number of tunnels or interfaces that can be tracked for a virtual router is 8.
- IPsec tunnel tracking is supported for IPv4 VRRP and IPv4 or IPv6 VRRP-E. IPv6 VRRP does not support IPsec tunnels.
- IPv4 VRRP or VRRP-E can track only IPv4 IPsec tunnels and IPv6 VRRP-E can track only IPv6 IPsec tunnels.

NOTE
Tracking IPsec tunnels is supported only on BR-MLX-10GX4-M-IPSEC modules running on MLXe devices.

# Configuring VRRP tracking for IPsec tunnels

IPsec tunnels can be tracked for a VRRP virtual routing instance.

This task is performed on any device configured with IPsec tunnels that is designated as a VRRP device. Before configuring IPsec tunnel tracking, IPsec tunnels must be configured. Repeat this task for all devices on which VRRP IPsec tunnel tracking is required up to a maximum of 8 tracked tunnels or interfaces.

NOTE
IPsec tunnel tracking is supported for IPv4 VRRP and IPv4 or IPv6 VRRP-E. IPv6 VRRP does not support IPsec tunnels.

1. Enter the **configure terminal** command to enter global configuration mode.

```
device# configure terminal
```

2. Enter the **router vrrp** command to configure VRRP globally.

```
device(config)# router vrrp
```

3. Configure the Ethernet interface.

```
device(config)# interface ethernet 1/6
```

4. Enter the IP address for the interface to be used for the virtual router ID (VRID).

```
device(config-if-e10000-1/6)# ip address 10.53.5.3/24
```

5. Enter the following command to enter the appropriate VRRP virtual router ID (VRID) mode.

```
device(config-if-e10000-1/6)# ip vrrp vrid 1
```

6. Enter the **track-port** command to set the track port and priority:

```
device(config-if-e10000-1/6-vrid-1)# track-port tunnel 1 priority 20
```

The priority value is used when a tracked port goes down and the new priority is set to this value. Ensure that the priority value is lower than the priority set for any existing master or backup device to force a renegotiation for the master device.

7. Exit to Privileged EXEC mode.

```
device(config-if-e10000-1/6)# end
```

8. Verify the tracked IPsec tunnel configuration using the **show ip vrrp** command.

```
device# show ip vrrp

VRID 1 (index 1)
 interface 1/6
 state MASTER
 administrative-status disabled
 version v2
 mode incomplete
 virtual mac 0000.5e00.0101
 priority 100
 current priority 100
 track-priority 20
 hello-interval 1 sec
 backup hello-interval 60 sec
 track-port tunnel 1(up)
```

The partial output shows the tracked port tunnel number and status.

The following example shows how to configure tunnel 1 on virtual router 1 to be tracked; if the tunnel fails, the VRRP priority of the device becomes 20, forcing a negotiation for a new master device.

```
device# configure terminal
device(config)# router vrrp
device(config)# interface ethernet 1/6
device(config-if-e10000-1/6)# ip address 10.53.5.1/24
device(config-if-e10000-1/6)# ip vrrp vrid 1
device(config-if-e10000-1/6-vrid-1)# track-port tunnel 1 priority 20
```

# Configuring VRRP-E tracking for IPsec tunnels

IPsec tunnels can be tracked for a VRRP-E virtual routing instance.

This task is performed on any device that is designated as a VRRP extended (VRRP-E) device. Before configuring IPsec tunnel tracking, IPsec tunnels must be configured. Repeat this task for all devices on which VRRP-E IPsec tunnel tracking is required up to a maximum of 8 tracked tunnels or interfaces.

NOTE

IPsec tunnel tracking is supported for IPv4 VRRP and IPv4 or IPv6 VRRP-E. IPv6 VRRP does not support IPsec tunnels.

1. On the device designated as a VRRP-E device, from privileged EXEC mode, enter global configuration mode by issuing the **configure terminal** command.

```
device# configure terminal
```

2. Globally enable VRRP-E.

```
device(config)# router vrrp-extended
```

3. Configure the Ethernet interface link.

```
device(config-vrrpe-router)# interface ethernet 1/5
```

4. Configure the IP address of the interface. All devices configured for the same virtual router ID (VRID) must be on the same subnet.

```
device(config-if-e1000-1/5)# ip address 10.53.5.3/24
```

5. Assign the device to VRID 1.

```
device(config-if-e1000-1/5)# ip vrrp-extended vrid 1
```

NOTE
You can assign a VRID number in the range of 1 through 255.

6. Designate this router as a backup VRRP device.

```
device(config-if-e1000-1/5-vrid-1)# backup priority 50 track priority 10
```

While configuring a backup device, you can set a priority that is used when a master VRRP device goes offline. The backup device with the highest priority will assume the role of master device.

7. Configure the IP address of the VRID.

```
device(config-if-e1000-1/5-vrid-1)# ip-address 10.53.5.254
```

The IP address associated with the VRID must not be configured on any of the devices used for VRRP-E.

8. Enter the **track-port** command to set the track port and priority for one end of the IPsec tunnel.

```
device(config-if-e1000-1/5-vrid-1)# track-port tunnel 1 priority 10
```

The priority value is used when a tracked tunnel goes down and the new priority is reduced by this value. Ensure that the priority value is lower than the priority set for any existing master or backup device to force a renegotiation for the master device.

9. Enter the **track-port** command to set the track port and priority of the IPsec tunnel.

```
device(config-if-e1000-1/5-vrid-1)# track-port tunnel 2 priority 20
```

The priority value is used when a tracked port goes down and the new priority is reduced by this value. Ensure that the priority value is lower than the priority set for any existing master or backup device to force a renegotiation for the master device.

10. Enable the VRRP-E session.

```
device(config-if-e1000-1/5-vrid-1)# activate
VRRP-E router 1 for this interface is activating
```

The following example configures a VRRP-E device to track IPsec tunnels 1 and 2.

```
device# configure terminal
device(config)# router vrrp-extended
device(config-vrrpe-router)# interface ethernet 1/5
device(config-if-e1000-1/5)# ip address 10.53.5.3/24
device(config-if-e1000-1/5)# ip vrrp-extended vrid 1
device(config-if-e1000-1/5-vrid-1)# backup
device(config-if-e1000-1/5-vrid-1)# ip-address 10.53.5.1
device(config-if-e1000-1/5-vrid-1)# track-port tunnel 1 priority 10
device(config-if-e1000-1/5-vrid-1)# track-port tunnel 2 priority 20
device(config-if-e1000-1/5-vrid-1)# activate
VRRP-E router 1 for this interface is activating
```

# Accept mode for backup VRRP devices

Accept mode allows a backup VRRP device to respond to ping, traceroute, and Telnet packets if the backup device becomes the master VRRP device.

For each VRRP virtual routing instance, there is one master device and all other devices are backups. Accept mode allows some network management functionality for backup VRRP devices, providing the ability to respond to ping, traceroute, and Telnet packets. By default, nonowner VRRP devices do not accept packets destined for the IPv4 or IPv6 VRID addresses. Troubleshooting network connections to the VRRP nonowner master device is difficult unless accept mode is enabled.

NOTE
The accept mode functionality enables a VRRP nonowner master device to respond to ping, Telnet, and traceroute packets, but the device will not respond to SSH packets.

## Enabling accept mode on a backup VRRP device

Enabling accept mode allows a backup VRRP device to respond to ping, traceroute, and Telnet packets if the backup device becomes the master VRRP device.

This task is performed on any device that is designated as a backup VRRP device, and the functionality is activated if the backup device becomes a master VRRP device. Repeat this task for all devices that are to be designated as backup devices.

NOTE
The accept mode functionality does not support SSH packets.

1. On the device designated as a backup VRRP device, from privileged EXEC mode, enter global configuration mode by issuing the **configure terminal** command.

```
device# configure terminal
```

2. Globally enable VRRP.

```
device(config)# router vrrp
```

3. Configure the Ethernet interface link.

```
device(config)# interface ethernet 1/1/5
```

4. Configure the IP address of the interface. All devices configured for the same virtual router ID (VRID) must be on the same subnet.

```
device(conf-if-e1000-1/1/5)# ip address 10.53.5.3/24
```

5. Assign this backup device to VRID 1, the same VRID as the VRRP owner device.

```
device(conf-if-e1000-1/1/5)# ip vrrp vrid 1
```

NOTE
You can assign a VRID number in the range of 1 through 255.

6. Designate this router as a backup VRRP device.

```
device(conf-if-e1000-1/1/5-vrid-1)# backup priority 110
```

While configuring a backup device, you can set a priority that is used when a master VRRP device goes offline. The backup device with the highest priority will assume the role of master device.

7. Enable accept mode for this device.

```
device(conf-if-e1000-1/1/5-vrid-1)# accept-mode
```

8. Exit configuration mode and return to privileged EXEC mode.

```
device(conf-if-e1000-1/1/5-vrid-1)# end
```

9. Verify that accept mode is enabled.

```
device# show ip vrrp vrid 1

Interface 1/1/5
----------------
auth-type no authentication
VRID 1 (index 1)
 interface 1/1/5
 state master
 administrative-status enabled
 version v2
 mode non-owner (backup)
 virtual mac aaaa.bbbb.cccc (configured)
 priority 110
 current priority 110
 track-priority 2
 hello-interval 1 sec
 accept-mdoe enabled
.
.
.
```

The following example enables accept mode for a backup VRRP device.

# Alternate VRRPv2 checksum for VRRPv3 IPv4 sessions

If VRRPv3 is configured on an Extreme device in a network with third-party peering devices using VRRPv2-style checksum calculations for IPv4 VRRPv3 sessions, a VRRPv2-style checksum must be configured for VRRPv3 IPv4 sessions on the device.

VRRPv3 introduced a new checksum method for both IPv4 and IPv6 sessions, and this version 3 checksum computation is enabled by default. To accommodate third-party devices that still use a VRRPv2-style checksum for IPv4 VRRPv3 sessions, a command-line interface (CLI) command is available for configuration on a device. The new version 2 checksum method is disabled by default and is applicable only to IPv4 VRRPv3 sessions. If configured for VRRPv2 sessions, the VRRPv2-style checksum command is accepted, but it has no effect.

# Enabling the VRRPv2 checksum computation method in a VRRPv3 IPv4 session

An alternate VRRPv2-style checksum can be configured in a VRRPv3 IPv4 session for compatibility with third-party network devices.

VRRPv3 uses the v3 checksum computation method by default for both IPv4 and IPv6 sessions on the NetIron OS devices. Third-party devices may have only a VRRPv2-style checksum computation available for a VRRPv3 IPv4 session. The **use-v2-checksum** command is entered in interface configuration mode.

1. Use the **configure terminal** command to enter global configuration mode.

```
device# configure terminal
```

2. Enable VRRP globally.

```
device(config)# router vrrp
```

3. Enter the **interface** command with an interface type and number.

```
device(config)# interface ethernet 2/4
```

4. To configure a VRRP virtual routing ID, use the **ip vrrp vrid** command with an associated ID number.

```
device(config-if-e1000-2/4)# ip vrrp vrid 14
```

5. To enable VRRP version 3 (VRRPv3), enter the **version** command with a version number of v3.

```
device(config-if-e1000-2/4-vrid-14)# version v3
```

6. To enable the v2 checksum computation method in an IPv4 VRRPv3 session, use the **use-v2-checksum** command in VRRP configuration mode.

```
device(config-if-e1000-2/4-vrid-14)# use-v2-checksum
```

7. Enter the IP address for the interface using the **ip-address** command.

```
device(config-if-e1000-2/4-vrid-14)# ip-address 10.14.14.99
```

8. To activate the interface, enter the **activate** command.

```
device(config-if-e1000-2/4-vrid-14)# activate
```

The following example shows the v2 checksum computation method enabled for an VRRPv3 IPv4 session on a device.

```
device# configure terminal
device(config)# router vrrp
device(config)# interface ethernet 2/4
device(config-if-e1000-2/4)# ip vrrp vrid 14
device(config-if-e1000-2/4-vrid-14)# version v3
device(config-if-e1000-2/4-vrid-14)# use-v2-checksum
device(config-if-e1000-2/4-vrid-14)# ip-address 10.14.14.99
device(config-if-e1000-2/4-vrid-14)# activate
```

## Displaying alternate VRRPv2 checksum settings

The verification of the use of the alternate VRRPv2-style checksum for VRRPv3 IPv4 sessions is achieved using several CLI commands.

The following steps are both optional and can be used to verify that the alternate VRRPv2-style checksum computation command, **use-v2-checksum**, has been set for VRRPv3 IPv4 sessions.

1. Use the **show running-config** command to verify that the **use-v2-checksum** command has been configured for a specified interface. Only part of the output is displayed.

```
device# show running-config

interface ethernet 2/4
 ip address 10.14.14.2/24
 ip vrrp vrid 14
  backup
  ip-address 10.14.14.99
  use-v2-checksum
  exit
```

2. Use the **show ip vrrp** command with a virtual router ID number to display the current settings of a specific VRRP session, including the **use-v2-checksum** command, if configured.

```
device# show ip vrrp vrid 14

Interface 2/4
----------------
auth-type no authentication

VRID 14 (index 1)
 interface 2/4
 state initialize
 administrative-status disabled
 version v3 - use-v2-checksum
 mode non-owner(backup)
 virtual mac 0000.5e00.010e
 priority 100
 current priority 100
 track-priority 1
 hello-interval 1 sec
 backup hello-interval 60 sec
 slow-start timer (configured) 0 sec
 advertise backup disabled
 dead-interval 3500 ms
 preempt-mode true
 ip-address 10.14.14.99
```

# Automatic generation of a virtual link-local address for VRRPv3

The virtual MAC address is used to automatically generate the IPv6 virtual link-local address to simplify the configuration of IPv6 VRRP and standardize implementations across vendor platforms. Subsequent VRRPv3 advertisements carry the auto-generated virtual link-local address.

The default VRRPv3 implementation allows only the link-local address that is configured on a physical interface to be used as the virtual IPv6 address of a VRRPv3 session. This limits configuring a link-local address for each VRRP instance on the same physical interface because there can be only one link-local address per physical interface.

When IPv6 link-local address auto-generation is configured for IPv6 VRRP, a virtual IPv6 link-local address is generated automatically using the EUI-64 result of the virtual MAC address. The virtual IPv6 link-local address is generated for a specific VRRP instance and the virtual link-local address is carried in VRRPv3 advertisements. The auto-generation process is defined in RFC 5798 allowing cross-vendor platform support. This ability to generate a link-local address automatically depends on the existence of a consistent virtual MAC address in the local network.

If the virtual link-local address is configured manually, the configured address takes precedence over the automatically generated address. The administrator should ensure that the configured virtual link-local address is consistent across all routers in the LAN. When the manually configured address is removed, the auto-generated address is used.

If there is a mismatch in the IPv6 addresses field, the devices drop the advertisements that are sent by backup VRRP routers. The advertisements from the master VRRP router are not dropped regardless of the IPv6 address comparison. The virtual MAC must be consistent on the local network. When the virtual MAC is modified, the virtual link-local address is regenerated.

As an Extreme proprietary protocol, VRRP Extended version 3 (VRRP-Ev3) is not supported.

## Enabling auto-generation of an IPv6 virtual link-local address

This task is performed on all the devices that are designated as backup VRRPv3 devices. Initially a backup priority is set to 100. For each VRRPv3 session there are master and backup routers, the IPv6 address assigned here to the VRID is the IPv6 address of the master router. The task is repeated on each backup VRRPv3 device with corresponding changes to the interface number and IPv6 address of the interface.

1. On the device designated as a backup VRRPv3 device, from privileged EXEC mode, enter configuration mode by issuing the **configure terminal** command.

   ```
   device# configure terminal
   ```

2. Globally enable the IPv6 VRRP protocol.

   ```
   device(config)# ipv6 router vrrp
   ```

3. Configure the ethernet interface link for the owner device.

   ```
   device(config)# interface ethernet 1/4
   ```

4. Configure the IPv6 address of the interface.

   ```
   device(config-if-e1000-1/4)# ipv6 address 3013::3/64
   ```

5. Assign the backup device to the virtual router ID (VRID) 2.

   ```
   device(config-if-e1000-1/4)# ipv6 vrrp vrid 2
   ```

   NOTE
   You can assign a VRID number in the range of 1 through 255.

6. Designate this router as the VRRPv3 backup device and assign a priority of 100.

   ```
   device(config-if-e1000-1/4-vrid-1)# backup priority 100
   ```

7. Automatically generate the IPv6 link-local address for the VRID for use in the local network.

   ```
   device(config-if-e1000-1/4-vrid-2)# ipv6-address auto-gen-link-local
   ```

8. Assign the global IPv6 address to the VRID.

```
device(config-if-e1000-1/4-vrid-2)# ipv6-address 3013::2
```

9. Enable the VRRP session.

```
device(config-if-e1000-1/4-vrid-2)# activate
```

10. To verify that link-local addresses are being automatically generated, enter the **show ipv6 vrrp** command for this VRID.

```
device(config-if-e1000-1/4-vrid-2)# show ipv6 vrrp vrid 2

VRID 2 (index 1)
 interface 1/1
 state master
 administrative-status enabled
 version v3
 mode owner
 virtual mac 0000.5e00.0101
 virtual link-local fe80::200:5eff:fe00:201
 priority 255
 current priority 255
 track-priority 2
 hello-interval 1000 ms
 backup hello-interval 60000 ms
 number of configured virtual address 2
 ipv6-address 1:2:45::2
 ipv6-address 1:2:46::2
 next hello sent in 300 ms
 Track MCT-VPLS-State: Disable
```

The following example configures an automatic generation of an IPv6 VRRPv3 link-local address for backup device.

```
device# configure terminal
device(config)# ipv6 router vrrp
device(config-ipv6-vrrp-router)# interface ethernet 1/4
device(config-if-e1000-1/4)# ipv6 address 3013::3/64
device(config-if-e1000-1/4)# ipv6 vrrp vrid 2
device(config-if-e1000-1/4-vrid-2)# backup priority 100
device(config-if-e1000-1/4-vrid-2)# ipv6-address auto-gen-link-local
device(config-if-e1000-1/4-vrid-2)# ipv6-address 3013::2
device(config-if-e1000-1/4-vrid-2)# activate
```

# Displaying VRRPv3 statistics

Various show commands can display statistical information about IPv6 VRRP configurations.

Before displaying statistics, VRRPv3 must be configured and enabled in your network to generate traffic.

Use one or more of the following commands to display VRRPv3 information. The commands do not have to be entered in this order.

1. Use the **exit** command to return to privileged EXEC mode, if required.

2. Enter the **show ipv6 vrrp** command to display IPv6 VRRPv3 configuration information.

```
device(config)# show ipv6 vrrp

Total number of VRRP routers defined: 1
Interface 1/3
----------------
auth-type no authentication
VRID 13 (index 2)
interface 1/3
state master
administrative-status enabled
version v3
mode non-owner(backup)
virtual mac 0000.5e00.0217
priority 100
current priority 100
track-priority 1
hello-interval 1000 ms
backup hello-interval 60000 ms
advertise backup disabled
dead-interval 3000 ms
preempt-mode true
ipv6-address fd2b::1
next hello sent in 700 ms
short-path-forwarding disabled
```

3. To view detailed statistical information about IPv6 VRRPv3, enter the **show ipv6 vrrp statistics** command.

```
device# show ipv6 vrrp statistics

Global IPv6 VRRP statistics
-------------------------------
- received vrrp packets with checksum errors = 0
- received vrrp packets with invalid version number = 0
- received vrrp packets with unknown or inactive vrid = 0
Interface 1/3
----------------
VRID 13
- number of transitions to backup state = 1
- number of transitions to master state = 1
- total number of vrrp packets received = 0
. received backup advertisements = 19
. received packets with zero priority = 0
. received packets with invalid type = 0
. received packets with invalid authentication type = 0
. received packets with authentication type mismatch = 0
. received packets with authentication failures = 0
. received packets dropped by owner = 0
. received packets with ttl errors = 0
. received packets with ipv6 address mismatch = 0
. received packets with advertisement interval mismatch = 0
. received packets with invalid length = 0
- total number of vrrp packets sent = 1175
. sent backup advertisements = 0
. sent packets with zero priority = 0
- received neighbor solicitation packets dropped = 0
- received proxy neighbor solicitation packets dropped = 0
- received ipv6 packets dropped = 0
```

# Clearing VRRPv3 statistics

VRRPv3 session counters can be cleared using a CLI command.

Ensure that VRRPv3 is configured and enabled in your network.

1. Enter the **end** command, if required, to return to privileged EXEC mode.
2. Enter the **clear ipv6 vrrp statistics** command.

```
device# clear ipv6 vrrp statistics
```

# VRRP-Ev3 Overview

VRRP Extended version 3 (VRRP-Ev3) introduces IPv6 address support to the Extreme Networks proprietary VRRP Extended version 2 (VRRP-Ev2) protocol. VRRP-Ev3 is designed to avoid the limitations in the standards-based VRRPv3 protocol.

To create VRRP-Ev3, Extreme Networks has implemented the following differences from the RFC 5798 that describes VRRPv3 to provide extended functionality and ease of configuration:

- VRRP-Ev3 does not include the concept of an owner device and a master VRRP-Ev3 device is determined by the priority configured on the device.
- While the VRRP-Ev3 virtual router IP address must belong in the same subnet as a real IP address assigned to a physical interface of the device on which VRRP-Ev3 is configured, it must not be the same as any of the actual IP addresses on any interface.
- Configuring VRRP-Ev3 uses the same task steps for all devices; no differences between master and backup device configuration. The device configured with the highest priority assumes the master role.

  NOTE
  VRRP-Ev3 is supported on the devices described in this guide. In a mixed-device environment, consult your documentation for the other devices to determine if VRRP-Ev3 is supported.
  VRRP-Ev3 does not interoperate with with VRRPv2 or VRRPv3 sessions.

# Enabling an IPv6 VRRP-Ev3 device

This task is performed on any device that is designated as a VRRP extended version 3 (VRRP-Ev3) device. For each VRRP-Ev3 virtual routing instance, there is one master device and all other devices are backups; but, unlike VRRPv3, every device is configured as a backup and the device with the highest priority becomes the master device. Repeat this task for all devices that are to be designated as VRRP-Ev3 devices.

NOTE
Only VRRPv3 or VRRP-Ev3 can be enabled in your network.

1. On the device designated as a VRRP-Ev3 device, from privileged EXEC mode, enter global configuration mode by issuing the **configure terminal** command.

```
device# configure terminal
```

2. Globally enable VRRP-Ev3.

```
device(config)# ipv6 router vrrp-extended
```

3. Configure the Ethernet interface link.

```
device(config-ipv6-vrrpe-router)# interface ethernet 1/7
```

4. Configure the IPv6 address of the interface. All devices configured for the same virtual router ID (VRID) must be on the same subnet.

```
device(config-if-e1000-1/7)# ipv6 address fd4b::4/64
```

5. Assign the device to VRID 4.

```
device(config-if-e1000-1/7)# ipv6 vrrp-extended vrid 4
```

   NOTE
   You can assign a VRID number in the range of 1 through 255.

6. Designate this router as a backup VRRPv3 device. All VRRP-Ev3 devices are initially configured as backup devices; the device with the highest priority assumes the role of master device.

```
device(config-if-e1000-1/7-vrid-4)# backup priority 110
```

   While configuring a backup device, you can set a priority that is used when the designated master VRRP device goes offline. The backup device with the highest priority will assume the role of master device.

7. Configure an IPv6 link-local address for the VRID.

```
device(config-if-e1000-1/7-vrid-4)# ipv6-address fe80::768e:f8ff:fe2a:0089
```

8. Configure a global IPv6 address for the VRID.

```
device(config-if-e1000-1/7-vrid-4)# ipv6-address fd4b::99
```

   The IPv6 address associated with the VRID must not be configured on any of the devices used for VRRP-Ev3.

9. Enable the VRRP session.

```
device(config-if-e1000-1/7-vrid-4)# activate
VRRP-E router 4 for this interface is activating
```

The following example configures a backup VRRP-Ev3 device.

```
device# configure terminal
device(config)# ipv6 router vrrp-extended
device(config-ipv6-vrrpe-router)# interface ethernet 1/7
device(config-if-e1000-1/7)# ipv6 address fd4b::4/64
device(config-if-e1000-1/7)# ipv6 vrrp-extended vrid 4
device(config-if-e1000-1/7-vrid-4)# backup  priority 50
device(config-if-e1000-1/7-vrid-4)# ipv6-address fe80::768e:f8ff:fe2a:0089
device(config-if-e1000-1/7-vrid-4)# ipv6-address fd4b::99
device(config-if-e1000-1/7-vrid-4)# activate
VRRP-E router 4 for this interface is activating
```

# VRRP-Ev3 sub-second failover

VRRP-Ev3 introduces a scale time factor to the advertisement interval that results in sub-second failover times.

In VRRP version 2, an advertisement interval can be set to decrease the time period between advertisements to allow for shorter or longer convergence times. In VRRPv3, a new CLI command is introduced to allow scaling of the advertisement interval timer. When a scaling value is configured, the existing advertisement interval timer value is divided by the scaling value. For example, if the advertisement interval is set to 1 second and the scaling value is set to 10, the new advertisement interval is 100 milliseconds. Using the timer scaling, VRRP-Ev3 sub-second convergence is possible if a master fails.

For each VRRP-Ev3 session, the same advertisement interval and scale value should be used. There are some limits on the number of VRRP sessions configured with advertisement intervals of one second or less, for details see the VRRPv3 Performance and Scalability Metrics section.

NOTE
MLX Series devices only support a scaling factor of 10. For interoperability with these devices, use an advertisement interval scale factor of 10.

## Configuring sub-second failover using VRRP-Ev3

Configuring a scale factor making the interval between VRRP advertisements to be set in milliseconds allows a sub-second convergence time if a master VRRP device fails.

The configuring sub-second failover using VRRP-Ev3 task is only supported by VRRP-Ev3.

NOTE
Increased timing sensitivity as a result of this configuration could cause protocol flapping during periods of network congestion.

1. Enter the **configure** command to access global configuration mode.

```
device# configure
```

2. Enter the **rbridge-id** command with an RBridge ID to enter RBridge ID configuration mode.

```
device(config)# rbridge-id 122
```

3. To globally enable VRRP-Ev3, enter the **ipv6 protocol vrrp-extended** command.

```
device(config-rbridge-id-122)# ipv6 protocol vrrp-extended
```

4. Enter the **interface ve** command with an associated VLAN number.

```
device(config-rbridge-id-122)# interface ve 2019
```

   In this example, virtual Ethernet (ve) configuration mode is entered and the interface is assigned with a VLAN number of 2019.

5. Enter an IPv6 address for the interface using the **ipv6 address** command.

```
device(config-ve-2019)# ipv6 address 2001:2019:8192::122/64
```

6. Enter the **ipv6 vrrp-extended-group** command with a number to assign a VRRP-E group to the device.

```
device(config-ve-2018)# ipv6 vrrp-extended-group 19
```

   In this example, VRRP-Ev3 group configuration mode is entered.

7. Enter the **advertisement-interval** command with a value to set the time period in seconds between VRRP advertisements.

```
device(config-vrrp-extended-group-19)# advertisement-interval 1
```

8. Enter the **advertisement-interval-scale** command with a value of 1, 2, 5, or 10. The VRRP advertisement interval is divided by this number to set the time period in milliseconds between VRRP advertisements.

```
device(config-vrrp-extended-group-19)# advertisement-interval-scale 10
```

   In this example, the scale number of 10 divided into the advertisement interval of 1 sets the interval between advertisements to 100 milliseconds. If a master VRRP-E device fails, the convergence time to a backup VRRP-E device may be in less than half a second.

The following example demonstrates how to configure a VRRP advertisement interval of 100 milliseconds for an IPv6 VRRP-Ev3 group.

```
device# configure
device(config)# rbridge-id 122
device(config-rbridge-id-122)# ipv6 protocol vrrp-extended
device(config-rbridge-id-122)# interface ve 2019
device(config-ve-2019)# ipv6 address 2001:2019:8192::122/64
device(config-ve-2019)# ipv6 vrrp-extended-group 19
device(config-vrrp-extended-group-19)# advertisement-interval 1
device(config-vrrp-extended-group-19)# advertisement-interval-scale 10
```

# Displaying and clearing VRRP-Ev3 statistics

Several show commands can display statistical information about IPv6 VRRP-Ev3 configurations. To reset the IPv6 VRRP-Ev3 statistics, there is a CLI command.

Before displaying statistics, VRRP-Ev3 must be configured and enabled in your network to generate traffic.

Use one or more of the following commands to display VRRP-Ev3 information. The commands do not have to be entered in this order.

1. Use the **exit** command to return to privileged EXEC mode, if required.
2. Enter the **show ipv6 vrrp-extended brief** command to display VRRP-Ev3 summary information.

```
device(config)# show ipv6 vrrp-extended brief

Total number of VRRP routers defined: 1
Flags Codes - P:Preempt 2:V2 3:V3 S:Short-Path-Fwd
Intf    VRID CurrPrio Flags State  Master-IPv6 Backup-IPv6 Virtual-IPv6
Address                                       Address     Address
-------------------------------------------------------------------------
1/3     2    100      P3-   Master Local       fd2b::2     fd2b::99
```

3. Enter the **show ipv6 vrrp-extended vrid 1** command to display detailed IPv6 VRRP-E configuration information about VRID 1.

```
device# show ipv6 vrrp-extended vrid 1
Interface 1/1
---------------
auth-type md5-authentication
VRID 1 (index 1)
interface 1/1
state master
administrative-status enabled
mode non-owner(backup)
virtual mac dddd.eeee.ffff (configured)
priority 100
current priority 100
track-priority 5
hello-interval 1 sec
backup hello-interval 60 sec
advertise backup disabled
dead-interval 0 ms
preempt-mode true
virtual ipv6 address 10:20:1::100
```

4. Enter the **clear ipv6 vrrp-extended statistics** command to reset the statistical counters for an IPv6 VRRP-Ev3 session.

```
device# clear ipv6 vrrp-extended statistics
```