CONFIGURATION GUIDE

Ruckus FastIron Layer 3 Routing Configuration Guide, 08.0.90

Supporting FastIron Software Release 08.0.90

Part Number: 53-1005567-01
Publication Date: 20 February 2019

Copyright, Trademark and Proprietary Rights Information

Export Restrictions

These products and associated technical data (in print or electronic form) may be subject to export control laws of the United States of America. It is your responsibility to determine the applicable regulations and to comply with them. The following notice is applicable for all products or technology subject to export control:

These items are controlled by the U.S. Government and authorized for export only to the country of ultimate destination for use by the ultimate consignee or end-user(s) herein identified. They may not be resold, transferred, or otherwise disposed of, to any other country or to any person other than the authorized ultimate consignee or end-user(s), either in their original form or after being incorporated into other items, without first obtaining approval from the U.S. government or as otherwise authorized by U.S. law and regulations.

Disclaimer

THIS CONTENT AND ASSOCIATED PRODUCTS OR SERVICES ("MATERIALS"), ARE PROVIDED "AS IS" AND WITHOUT WARRANTIES OF ANY KIND, WHETHER EXPRESS OR IMPLIED. TO THE FULLEST EXTENT PERMISSIBLE PURSUANT TO APPLICABLE LAW, ARRIS DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, TITLE, NON-INFRINGEMENT, FREEDOM FROM COMPUTER VIRUS, AND WARRANTIES ARISING FROM COURSE OF DEALING OR COURSE OF PERFORMANCE. ARRIS does not represent or warrant that the functions described or contained in the Materials will be uninterrupted or error-free, that defects will be corrected, or are free of viruses or other harmful components. ARRIS does not make any warranties or representations regarding the use of the Materials in terms of their completeness, correctness, accuracy, adequacy, usefulness, timeliness, reliability or otherwise. As a condition of your use of the Materials, you warrant to ARRIS that you will not make use thereof for any purpose that is unlawful or prohibited by their associated terms of use.

Limitation of Liability

IN NO EVENT SHALL ARRIS, ARRIS AFFILIATES, OR THEIR OFFICERS, DIRECTORS, EMPLOYEES, AGENTS, SUPPLIERS, LICENSORS AND THIRD PARTY PARTNERS, BE LIABLE FOR ANY DIRECT, INDIRECT, SPECIAL, PUNITIVE, INCIDENTAL, EXEMPLARY OR CONSEQUENTIAL DAMAGES, OR ANY DAMAGES WHATSOEVER, EVEN IF ARRIS HAS BEEN PREVIOUSLY ADVISED OF THE POSSIBILITY OF SUCH DAMAGES, WHETHER IN AN ACTION UNDER CONTRACT, TORT, OR ANY OTHER THEORY ARISING FROM YOUR ACCESS TO, OR USE OF, THE MATERIALS. Because some jurisdictions do not allow limitations on how long an implied warranty lasts, or the exclusion or limitation of liability for consequential or incidental damages, some of the above limitations may not apply to you.

Trademarks

ARRIS, the ARRIS logo, Ruckus, Ruckus Wireless, Ruckus Networks, Ruckus logo, the Big Dog design, BeamFlex, ChannelFly, EdgeIron, FastIron, HyperEdge, ICX, IronPoint, OPENG, SmartCell, Unleashed, Xclaim, ZoneFlex are trademarks of ARRIS International plc and/or its affiliates. Wi-Fi Alliance, Wi-Fi, the Wi-Fi logo, the Wi-Fi CERTIFIED logo, Wi-Fi Protected Access (WPA), the Wi-Fi Protected Setup logo, and WMM are registered trademarks of Wi-Fi Alliance. Wi-Fi Protected Setup™, Wi-Fi Multimedia™, and WPA2™ are trademarks of Wi-Fi Alliance. All other trademarks are the property of their respective owners.

Contents

Preface

Document Conventions

The following table lists the text conventions that are used throughout this guide.

TABLE 1 Text Conventions

Convention	Description	Example
`monospace`	Identifies command syntax examples	`device(config)# interface ethernet 1/1/6`
bold	User interface (UI) components such as screen or page names, keyboard keys, software buttons, and field names	On the **Start** menu, click **All Programs**.
italics	Publication titles	Refer to the *Ruckus Small Cell Release Notes* for more information.

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 some information that you must read before continuing with the current action or task.

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.

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.
	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.

Document Feedback

Ruckus is interested in improving its documentation and welcomes your comments and suggestions.

You can email your comments to Ruckus at ruckus-docs@arris.com.

When contacting us, include the following information:

- Document title and release number
- Document part number (on the cover page)
- Page number (if appropriate)

For example:

- Ruckus SmartZone Upgrade Guide, Release 5.0
- Part number: 800-71850-001 Rev A
- Page 7

Ruckus Product Documentation Resources

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

Release Notes and other user documentation are available at https://support.ruckuswireless.com/documents. You can locate the documentation by product or perform a text search. Access to Release Notes requires an active support contract and a Ruckus Support Portal user account. Other technical documentation content is available without logging in to the Ruckus Support Portal.

White papers, data sheets, and other product documentation are available at https://www.ruckuswireless.com.

Online Training Resources

To access a variety of online Ruckus training modules, including free introductory courses to wireless networking essentials, site surveys, and Ruckus products, visit the Ruckus Training Portal at https://training.ruckuswireless.com.

Contacting Ruckus Customer Services and Support

The Customer Services and Support (CSS) organization is available to provide assistance to customers with active warranties on their Ruckus products, and customers and partners with active support contracts.

For product support information and details on contacting the Support Team, go directly to the Ruckus Support Portal using https://support.ruckuswireless.com, or go to https://www.ruckuswireless.com and select **Support**.

What Support Do I Need?

Technical issues are usually described in terms of priority (or severity). To determine if you need to call and open a case or access the self-service resources, use the following criteria:

- Priority 1 (P1)—Critical. Network or service is down and business is impacted. No known workaround. Go to the **Open a Case** section.
- Priority 2 (P2)—High. Network or service is impacted, but not down. Business impact may be high. Workaround may be available. Go to the **Open a Case** section.
- Priority 3 (P3)—Medium. Network or service is moderately impacted, but most business remains functional. Go to the **Self-Service Resources** section.
- Priority 4 (P4)—Low. Requests for information, product documentation, or product enhancements. Go to the **Self-Service Resources** section.

Open a Case

When your entire network is down (P1), or severely impacted (P2), call the appropriate telephone number listed below to get help:

- Continental United States: 1-855-782-5871
- Canada: 1-855-782-5871
- Europe, Middle East, Africa, Central and South America, and Asia Pacific, toll-free numbers are available at https://support.ruckuswireless.com/contact-us and Live Chat is also available.
- Worldwide toll number for our support organization. Phone charges will apply: +1-650-265-0903

We suggest that you keep a physical note of the appropriate support number in case you have an entire network outage.

Self-Service Resources

The Ruckus Support Portal at https://support.ruckuswireless.com offers a number of tools to help you to research and resolve problems with your Ruckus products, including:

- Technical Documentation—https://support.ruckuswireless.com/documents

- Community Forums—https://forums.ruckuswireless.com/ruckuswireless/categories
- Knowledge Base Articles—https://support.ruckuswireless.com/answers
- Software Downloads and Release Notes—https://support.ruckuswireless.com/#products_grid
- Security Bulletins—https://support.ruckuswireless.com/security

Using these resources will help you to resolve some issues, and will provide TAC with additional data from your troubleshooting analysis if you still require assistance through a support case or RMA. If you still require help, open and manage your case at https://support.ruckuswireless.com/case_management.

About This Document

What's new in this document

The following table describes the changes to this guide for the FastIron 08.0.90 release.

TABLE 2 Summary of changes in FastIron release 08.0.90

Feature	Description	Location
Support for the Ruckus ICX 7850	Introduced support for the Ruckus ICX 7850	Changes occur throughout the text.
Bidirectional Forwarding Detection (BFD)	Bidirectional Forwarding Detection (BFD) is a lightweight hello protocol that rapidly detects link faults and improves network performance by providing fast forwarding path failure detection times.	Refer to BFD.
IPv6 Neighbor Discovery(ND) Proxy	IPv6 Neighbor Discovery(ND) Proxy enables the hosts in different broadcast domains or VLANs to communicate with each other.	Refer to IPv6 Neighbor Discovery Proxy on page 126.
Disabling the Opaque LSA Capability	The opaque link-state advertisement (LSA) capability is enabled by default and can be disabled. When the opaque LSA capability is disabled, the device does not accept opaque LSAs from a peer. These LSAs are not added to the OSPFv2 link state database and are not flooded to the opaque capable peers.	Refer to Disabling the Opaque LSA Capability on page 206.
Updates to address defects	Minor updates on content throughout to address defects.	All chapters.
Minor editorial updates	Minor editorial updates were made throughout the Configuration Guide.	All chapters.

Supported hardware

This guide supports the following Ruckus products:

- Ruckus ICX 7850 Series
- Ruckus ICX 7750 Series
- Ruckus ICX 7650 Series
- Ruckus ICX 7450 Series
- Ruckus ICX 7250 Series
- Ruckus ICX 7150 Series

For information about what models and modules these devices support, see the hardware installation guide for the specific product family.

ARP - Address Resolution Protocol

ARP parameter configuration

Address Resolution Protocol (ARP) is a standard IP protocol that enables an IP Layer 3 switch to obtain the MAC address of another device interface when the Layer 3 switch knows the IP address of the interface. ARP is enabled by default and cannot be disabled.

NOTE
Ruckus Layer 2 switches also support ARP. However, the configuration options described later in this section apply only to Layer 3 switches, not to Layer 2 switches.

How ARP works

A Layer 3 switch needs to know a destination MAC address when forwarding traffic, because the Layer 3 switch 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 Layer 3 switch. The device can be the packet final destination or the next-hop router toward the destination.

The Layer 3 switch encapsulates IP packets in Layer 2 packets regardless of whether the ultimate destination is locally attached or is multiple router hops away. Because the Layer 3 switch IP route table and IP forwarding cache contain IP address information but not MAC address information, the Layer 3 switch cannot forward IP packets based solely on the information in the route table or forwarding cache. The Layer 3 switch needs to know the MAC address that corresponds with the IP address of either the packet 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 Layer 3 switch 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 destination. In each case, the Layer 3 switch must encapsulate the packet and address it to the MAC address of a locally attached device, the next-hop router toward the IP packet destination.

To obtain the MAC address required for forwarding a datagram, the Layer 3 switch first 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 Layer 3 switch receives an ARP reply or receives an ARP request (which contains the sender IP address and MAC address). A static entry enters the ARP cache from the separate static ARP 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 Layer 3 switch 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.

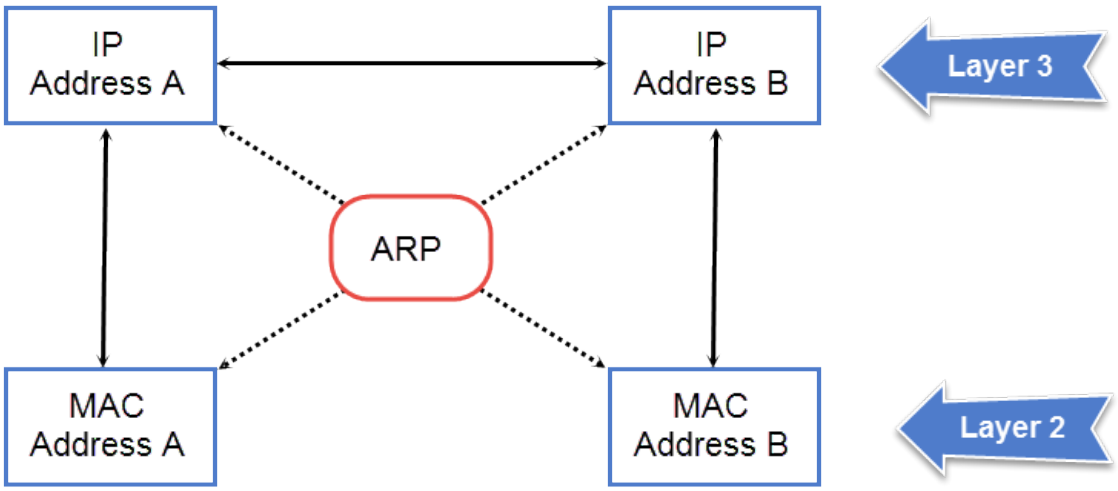
If the ARP cache does not contain an entry for the destination IP address, the Layer 3 switch broadcasts an ARP request out all its IP interfaces. The ARP request contains the IP address of the destination. If the device with the IP address is directly attached to the Layer 3 switch, the device sends an ARP response containing its MAC address. The response is a unicast packet addressed directly to the Layer 3 switch. The Layer 3 switch 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 Layer 3 switch. A MAC broadcast is not routed to other networks. However, some routers, including Ruckus Layer 3 switches, can be configured to reply to ARP requests from one network on behalf of devices on another network.

NOTE
If the router 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 Layer 3 switch knows of no route to the destination address), the router sends an ICMP Host Unreachable message to the source.

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



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.

Rate limiting ARP packets

To limit the number of ARP packets the device will accept each second, enter the **rate-limit-arp** command at the global CONFIG level of the CLI.

```
device(config)# rate-limit-arp 100
```

This command configures the device to accept up to 100 ARP packets each second. If the device receives more than 100 ARP packets during a one-second interval, the device drops the additional ARP packets during the remainder of that one-second interval.

The *num* variable specifies the number of ARP packets and can be from 0 through 100. If you specify 0, the device will not accept any ARP packets.

NOTE
If you want to change a previously configured the ARP rate limiting policy, you must remove the previously configured policy using the **no rate-limit-arp** command before entering the new policy.

ARP aging

When the Layer 3 switch places an entry in the ARP cache, the Layer 3 switch 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 ARP age affects dynamic (learned) entries only, not static entries. The default ARP age is ten minutes. On Layer 3 switches, 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. You cannot change the ARP age on Layer 2 switches.

NOTE
Host devices connected to an ICX 7750 that also have a valid IP address and reply periodically to the arp request are not timed out, even if no traffic is destined for the device. This behavior is restricted to only ICX 7750 devices.

Configuring global ARP aging

Specifies the global ARP aging time.

NOTE
If you specify 0, aging is disabled and entries do not age out. The default ARP age is 10 minutes. On Layer 3 switches, you can change the ARP age to a value from 0 through 240 minutes.

The following task configures the ARP aging time globally.

1. Enter the global configuration mode

```
device# configure terminal
```

2. Enter the **ip arp-age** command followed by the aging time in minutes.

```
device(config)# ip arp-age 100
```

The following example configures the global ARP aging time parameter to 100 minutes.

```
device# configure terminal
device(config)# ip arp-age 100
```

Configuring the ARP aging on a specific interface

Configures the ARP aging on an interface using the age value at the interface configuration mode.

The following task overrides the global ARP aging time.

1. Enter the global configuration mode

```
device# configure terminal
```

2. Specify the interface to be configured in the interface mode.

```
device(config)# interface ethernet 1/1/1
```

3. Enter the **ip arp-age** command followed by the aging time in minutes.

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

The following example configures ARP aging time to 30 minutes on Ethernet interface 1/1/1.

```
device# configure terminal
device(config)# interface ethernet 1/1/1
device(config-if-e1000-1/1/1)# ip arp-age 30
```

Enabling proxy ARP globally

Proxy ARP enables the Layer 3 switches to respond ARP requests for devices on one network on behalf of devices in another network.

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. Proxy ARP is disabled by default on Ruckus Layer 3 switches.

For example, if Proxy ARP is enabled on a Layer 3 switch connected to two subnets, 10.10.10.0/24 and 10.20.20.0/24, the Layer 3 switch 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
This feature is not supported on Ruckus Layer 2 switches.

1. Enter the global configuration mode.

```
device# configure terminal
```

2. Enable proxy ARP using the following command.

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

Enabling proxy ARP on an interface

Enabling proxy ARP at the interface level overrides the global proxy ARP.

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

1. Enter the global configuration mode

```
device# configure terminal
```

2. Specify the interface to be configured in the interface mode.

```
device(config)# interface ethernet 1/1/1
```

3. Enable the proxy ARP command on the specified interface.

```
device(config-if-e10000-1/1/1)# ip proxy-arp enable
```

By default, gratuitous ARP is disabled for local proxy ARP.

The following example enables proxy ARP on Ethernet interface 1/1/1.

```
device# configure terminal
device(config)# interface ethernet 1/1/1
device(config-if-e10000-1/1/1)# ip proxy-arp enable
```

Disabling proxy ARP on an interface

Use th following commands to disable the proxy ARP at the interface level.

1. Enter the global configuration mode

   ```
   device# configure terminal
   ```

2. Specify the interface to be configured in the interface mode.

   ```
   device(config)# interface ethernet 1/1/1
   ```

3. Disable the proxy ARP command on the specified interface.

   ```
   device(config-if-e10000-1/1/1)# ip proxy-arp disable
   ```

The following example disables proxy ARP on Ethernet interface 1/1/1.

```
device# configure terminal
device(config)# interface ethernet 1/1/1
device(config-if-e10000-1/1/1)# ip proxy-arp disable
```

Static ARP

Static ARP entries are added to the ARP cache when they are configured. Static ARP entries are useful in cases where you want to pre-configure an entry for a device that is not connected to the Layer 3 switch, or you want to prevent a particular entry from aging out.

Ruckus Layer 3 switches have a static ARP table, in addition to the regular ARP cache. Unlike static ARP entries, dynamic ARP entries are removed 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 Ruckus device receives an ARP request from the device that has the entry address.

NOTE
The maximum number of static ARP entries you can configure depends on the software version running on the device.

TABLE 3 Static ARP entry support

Device	Default	Maximum
ICX 7150	256	256
ICX 7250	512	2048
ICX 7450	512	6000
ICX 7650	512	1024
ICX 7750	512	1024
ICX 7850	512	1024

Creating static ARP entries

Static ARP entries are used to administratively define the IP to MAC address mappings or Layer 3 to Layer 2 address mappings.

NOTE
You cannot create static ARP entries on a Layer 2 switch.

1. Enter the Global configuration mode.

```
device# congigure terminal
```

2. Enter the following command to create a static ARP entry in the ARP cache.

```
device(config)# arp 10.53.4.2 0000.0054.2348 ethernet 1/1/2
```

The following example allows ARP responses from the IP address 10.53.4.2 associated with the MAC address 0000.0054.2348, for Ethernet interface 1/1/2.

```
device# configure terminal
device(config)# arp 10.53.4.2 0000.0054.2348 ethernet 1/1/2
```

Creating static ARP table entries

Static ARP table entries can be increased on a Ruckus Layer 3 switch.

NOTE
The basic procedure for changing the static ARP table size is the same as the procedure for changing other configurable cache or table sizes.

1. Enter the Global configuration mode.

```
device# congigure terminal
```

2. Enter the following command and indicate the maximum number of static ARP table entries depending on the software version.

```
device(config)# system-max ip-static-arp 1000
```

3. Save the running configuration to the startup-configuration file.

```
device(config)# write memory
```

4. Exit from the Global configuration mode.

```
device# end
```

5. Reload the software after changing the static ARP table size to place the change into effect.

```
device(config)# write memory
```

The following example configures the static table entry from its default value to 1000.

```
device# configure terminal
device(config)# system-max ip-static-arp 1000
device(config)# write memory
device(config)# end
device# reload
```

Enabling learning gratuitous ARP

Learning gratuitous ARP enables Ruckus Layer 3 devices to learn ARP entries from incoming gratuitous ARP packets from the hosts which are directly connected. This helps to achieve faster convergence for the hosts when they are ready to send traffic.

A new ARP entry is created when a gratuitous ARP packet is received. If the ARP is already existing, it will be updated with the new content.

NOTE
The **clear arp** command can be used to clear the learned ARP entries. Remember that the static ARP entries are not removed.

1. The following command will help to check whether the ARP is enabled or disabled.

```
device# show run
```

2. Enter the device Global configuration level.

```
device# configure terminal
```

3. Enable learning gratuitous ARP.

```
device(config)# ip arp learn-gratuitous-arp
```

4. The below command will help you to see the newly learned ARP entries.

```
device# show arp
```

Disabling next hop or ARP port movement syslog message generation

Whenever a port, on which a MAC address for an ARP is learned, is moved to a different port, a syslog message is generated by default.

This may cause flooding of the syslog server or console with syslog messages in certain deployments where next hop or ARP port movement occurs continuously. In such scenarios, the default behavior can be disabled and syslog messages can be prevented from being generated with every port movement for ARP entries using the **no ip arp port-move-syslog** command.

The status of the nexthop or ARP port movement syslog message (enabled or disabled) can be viewed in the output of the **show ip** command.

The following example shows sample output of the **show ip** command in which the status of the next hop or ARP port movement syslog message (enabled) is displayed.

```
device(config)# show ip
  ttl: 64, arp-age: 10, bootp-relay-max-hops: 4
  router-id : 10.1.1.1
  enabled : BGP4  UDP-Broadcast-Forwarding  Source-Route  Load-Sharing  RARP  VSRP
  arp-port-move-syslog
  disabled: Route-Only  Directed-Broadcast-Forwarding  IRDP  Proxy-ARP  RIP  OSPF
  VRRP  VRRP-Extended  ICMP-Redirect  add-host-route-first

device(config)# no ip arp port-move-syslog
device(config)# show ip
Global Settings
  ttl: 64, arp-age: 10, bootp-relay-max-hops: 4
  router-id : 10.1.1.1
  enabled : BGP4  UDP-Broadcast-Forwarding  Source-Route  Load-Sharing  RARP  VSRP
```

```
  disabled: Route-Only  Directed-Broadcast-Forwarding  IRDP  Proxy-ARP  RIP  OSPF
  VRRP  VRRP-Extended  ICMP-Redirect  add-host-route-first arp-port-move-syslog
```

The following example shows sample output of the **show ip** command in which the status of the next hop or ARP port movement syslog message (disabled) is displayed.

```
device(config)# no ip arp port-move-syslog
device(config)# show ip
Global Settings
  ttl: 64, arp-age: 10, bootp-relay-max-hops: 4
  router-id : 10.1.1.1
  enabled : BGP4  UDP-Broadcast-Forwarding  Source-Route  Load-Sharing  RARP  VSRP

  disabled: Route-Only  Directed-Broadcast-Forwarding  IRDP  Proxy-ARP  RIP  OSPF
  VRRP  VRRP-Extended  ICMP-Redirect  add-host-route-first arp-port-move-syslog
```

ARP Packet Validation

You can enable validation options that check incoming ARP packets to avoid traffic interruption or loss.

To avoid traffic interruption or loss, ARP Packet Validation allows the user to detect and drop ARP packets that do not pass the ARP validation process. ARP Packet Validation is disabled by default and can be enabled at the global configuration level. This functionality can be configured for the destination MAC address, the IP address and the source MAC address or with a combination of these parameters. The Ethernet header contains the destination MAC address and source MAC address, while the ARP packet contains the sender hardware address and target hardware address.

Follow these steps to perform checks on incoming ARP packets.

1. Enter global configuration mode.

   ```
   device# configure terminal
   ```

2. Enter the **ip arp inspection validate** command followed by one or more of the available options to perform a check on incoming ARP packets.
   - **dst-mac**

     The destination MAC address in the Ethernet header must match the target hardware address in the body of ARP response packets. Packets with different MAC addresses are classified as invalid and are dropped.
   - **src-mac**

     The source MAC address in the Ethernet header must match the sender hardware address in the body of ARP request and response packets. Packets with different MAC addresses are classified as invalid and are dropped.
   - **ip**

     Each ARP packet has a valid sender IP address and target IP address. In ARP response packets, the target IP address cannot be an invalid or unexpected IP address. The sender IP address cannot be an invalid or unexpected IP address in ARP request or response packets. Addresses include 0.0.0.0, 255.255.255.255, and all IP multicast addresses. Packets with invalid and unexpected IP addresses are classified as invalid and are dropped.

The following example enables validation of ARP packets based on the destination MAC address.

```
device(config)# configuration terminal
device(config)# ip arp inspection validate dst-mac
```

Ingress ARP packet priority

You can configure the priority of ingress ARP packets to an optimum value based on network configuration and traffic volume. Ingress ARP packets have a default priority value of 4. At the default priority value, ingress ARP packets may be dropped because of high traffic volume or non-ARP packets with higher priority values. This can cause devices to become unreachable. If the ingress ARP packets are set to a higher priority than the default priority value, a high volume of ARP traffic may lead to control traffic being dropped. This may cause traffic loops in the network.

NOTE
You cannot change the priority of the ingress ARP packets on the management port.

Configuring the priority of ingress ARP packets

NOTE
Stacking packets are given the highest priority (7). When ARP packet priority is set to 7, the packets are treated as lower priority than stacking packets.

To change the priority of incoming ARP packets, complete these steps.

1. Enter global configuration mode.

```
device# configure terminal
```

2. Enter the **arp-internal-priority** command followed by the appropriate value. The default is 4. You may choose a value from 0 through 7, with 7 being the highest priority.

```
device(config)# arp-internal-priority 4
```

This example resets the priority of incoming ARP packets to the default priority of 4.

The following example shows the priority of incoming ARP packets set to level 7.

```
device# configure terminal
device(config)# arp-internal-priority 7
```

Displaying the ARP table

To display the ARP table, enter the **show arp** command.

```
device# show arp
Total number of ARP entries: 2
Entries in default routing instance:
No.   IP Address       MAC Address    Type     Age Port               Status
1     10.1.1.100       0000.0000.0100 Dynamic  0    1/1/1*2/1/25      Valid
2     10.37.69.129     02e0.5215.cae3 Dynamic  0    mgmt1             Valid
```

The command displays all ARP entries in the system.

Reverse Address Resolution Protocol configuration

Reverse Address Resolution Protocol (RARP) allows an IP host that does not have a means of storing its IP address across power cycles or software reloads to query a directly-attached router for an IP address. This provides a simple mechanism for directly-attached IP hosts to boot over the network.

RARP is enabled by default. However, you must create a RARP entry for each host that will use the Layer 3 device for booting. A RARP entry consists of the following information:

- Entry sequence number in the RARP table
- MAC address of the boot client
- IP address the Layer 3 device assigns to the client

When a client sends a RARP broadcast to request an IP address, the Layer 3 device responds to the request by looking in the RARP table for an entry that contains the client MAC address. If the RARP table contains an entry for the client, the Layer 3 device sends a unicast response to the client that contains the IP address associated with the client MAC address in the RARP table. If the RARP table does not contain an entry for the client, the Layer 3 device silently discards the RARP request and does not reply to the client.

How RARP Differs from BootP and DHCP

RARP, BootP, and DHCP are different methods for providing IP addresses to IP hosts when they boot. These methods differ in the following ways:

- Location of configured host addresses
  - RARP requires static configuration of the host IP addresses on the Layer 3 device. The Layer 3 device replies directly to a host request by sending an IP address you have configured in the RARP table.
  - The Layer 3 device forwards BootP and DHCP requests to a third-party BootP/DHCP server that contains the IP addresses and other host configuration information.
- Connection of host to boot source (Layer 3 device or BootP/DHCP server)
  - RARP requires the IP host to be directly attached to the Layer 3 device.
  - An IP host and the BootP/DHCP server can be on different networks and on different routers as long as the routers are configured to forward ("help") the host boot request to the boot server.
  - You can centrally configure other host parameters on the BootP/DHCP server and supply those parameters to the host along with its IP address.

To configure the Layer 3 device to forward BootP/DHCP requests when boot clients and boot servers are on different subnets on different Layer 3 device interfaces, refer to the DHCP client section in the *Ruckus FastIron DHCP Configuration Guide*.

Disabling RARP

RARP is enabled by default.

To disable RARP, enter the following command at the global CONFIG level.

```
device(config)# no ip rarp
```

To re-enable RARP, enter the following command.

```
device(config)# ip rarp
```

Creating static RARP entries

Layer 3 switch can send an IP address in reply to a client RARP request only after creating a RARP entry for that client.

NOTE
The RARP entry number can be any unused number from 1 to the maximum number of RARP entries supported on the device. To determine the maximum number of entries supported on the device, refer to the section "Displaying and modifying system parameter default settings" in the *Ruckus FastIron Layer 2 Switching Configuration Guide*.

1. Enter the global configuration mode.

```
device# configure terminal
```

2. Assign a static IP RARP entry for static routes.

```
device(config)# rarp 1 0000.0054.2348 10.53.4.2
```

The following example creates RARP entry. The Layer 3 switch send IP address 192.53.4.2 in response to the RARP request from the client with MAC address 0000.0054.2348 having an RARP entry number 1.

```
device# configure terminal
device(config)# rarp 1 0000.0054.2348 10.53.4.2
```

Changing the maximum number of static RARP entries supported

The number of RARP entries the Layer 3 switch supports depends on how much memory the Layer 3 switch has. To determine how many RARP entries your Layer 3 switch can have, display the system default information using the procedure in the section "Displaying system parameter default values" in the *Ruckus FastIron Layer 2 Switching Configuration Guide*.

If your Layer 3 switch allows you to increase the maximum number of RARP entries, you can use a procedure in the same section to do so.

NOTE
You must save the configuration to the startup-config file and reload the software after changing the RARP cache size to place the change into effect.

Dynamic ARP Inspection overview

For enhanced network security, you can configure the Ruckus device to inspect and keep track of Dynamic Host Configuration Protocol (DHCP) assignments.

Dynamic ARP Inspection (DAI) enables the Ruckus device to intercept and examine all ARP request and response packets in a subnet and discard 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 misconfiguration of client IP addresses.

DAI allows only valid ARP requests and responses to be forwarded and supports Multi-VRFs with overlapping address spaces.

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 ARP table, it creates an ARP request to resolve the mapping. All computers on the subnet receive and process the ARP requests, and the host whose IP address matches the IP address in the request sends 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 ARP 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 (for example, the default router). After the attack, all traffic from the device under attack flows through the attacker computer and then to the router, switch, or host.

How Dynamic ARP Inspection works

Dynamic ARP Inspection (DAI) allows only valid ARP requests and responses to be forwarded.

A Ruckus device on which DAI is configured does the following:

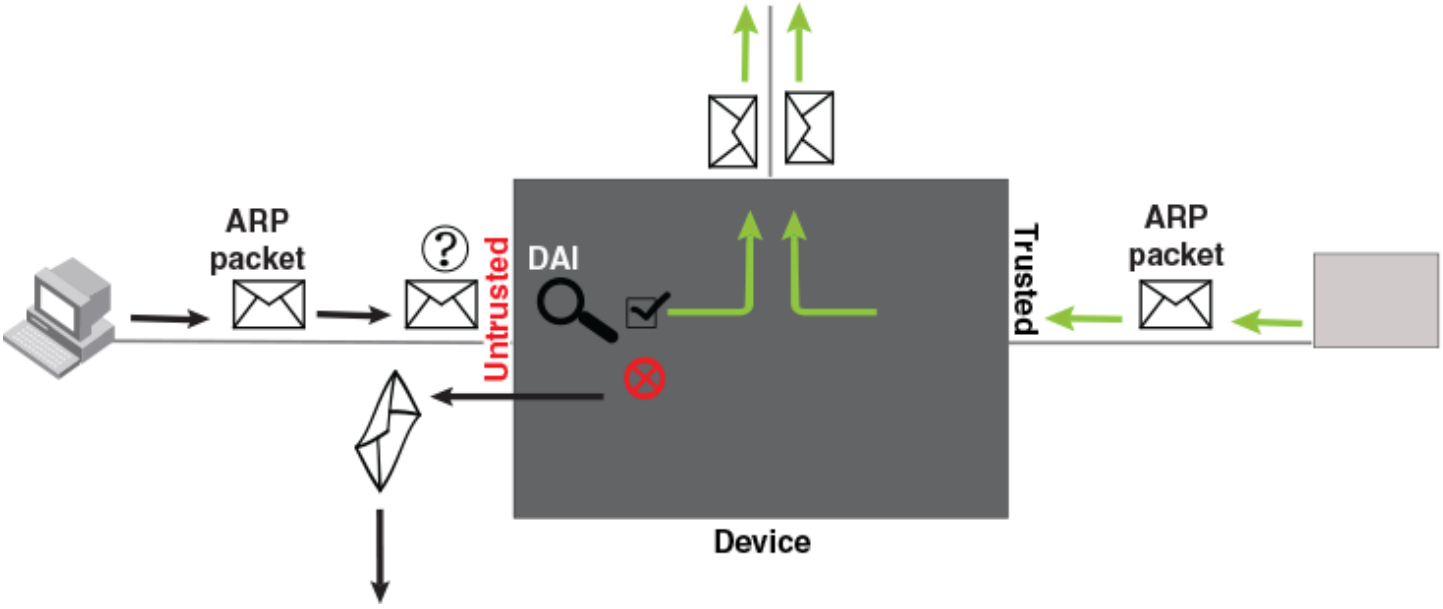
- Intercepts ARP packets received by the system CPU
- Inspects all ARP requests and responses received on untrusted ports
- Verifies that each of the intercepted packets has a valid IP-to-MAC address binding before updating the local ARP table, or before forwarding the packet to the appropriate destination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

When you enable DAI 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 the following figure. DAI carries out the inspection based on IP-to-MAC address bindings stored in a trusted binding database. For the Ruckus device, the binding database is the ARP table and the DHCP snooping table, which supports DAI, DHCP snooping, and IP Source Guard. To inspect an ARP request packet, DAI checks the source IP address 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 the following figure.

FIGURE 2 Dynamic ARP Inspection at work



ARP and DHCP snoop entries

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

- Dynamic ARP - Normal ARP learned from trusted ports.
- Static ARP - Statically configured IP/MAC/port mapping.
- Inspection ARP - Statically configured IP-to-MAC mapping, where the port is initially unspecified. The actual physical port mapping will be resolved and updated from validated ARP packets. Refer to Configuring an inspection ARP entry on page 36.
- DHCP-Snooping ARP - Information collected from snooping DHCP packets when DHCP snooping is enabled on VLANs. DHCP snooping entries are stored in a different table and are not part of the ARP table.

The status of an ARP entry is either pending or valid:

- Valid - The mapping is valid, and the port is resolved. This is always the case for static ARP entries.
- Pending - For normal dynamic ARP entries before they are resolved, and the port is mapped. Their status changes to valid when they are resolved, and the port is mapped.

Refer to System reboot and the binding database section in the *Ruckus FastIron DHCP Configuration Guide*.

Configuration notes and feature limitations for DAI

The following configuration notes and limitations apply when configuring DAI:

- There is a limit on the number of static ARP inspection entries that can be configured. The limit is determined by the **system-max max-static-inspect-arp-entries** command. The maximum value is 2048. The default value is 512.

Changing the system maximum values requires a system reload. Refer to the *Ruckus FastIron Command Reference* for more information.

- ACLs are supported on member ports of a VLAN on which DHCP snooping and Dynamic ARP Inspection (DAI) are enabled.
- DAI is supported on a VLAN without a VE, or on a VE with or without an assigned IP address.
- DAI is supported on LAG ports.

Dynamic ARP Inspection configuration

Configuring DAI consists of the following steps.

1. Configure inspection ARP entries for hosts on untrusted ports. Refer to Configuring an inspection ARP entry on page 36.
2. Enable DAI on a VLAN to inspect ARP packets. Refer to Enabling DAI on a VLAN on page 37.
3. 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. Refer to Enabling ARP inspection trust on a port on page 37.
4. Enable DHCP snooping to populate the DHCP snooping IP-to-MAC address binding database.

   Dynamic ARP inspection is disabled by default and the trust setting for ports is by default untrusted.

Configuring an inspection ARP entry

Dynamic ARP inspection must be enabled to use static ARP inspection entries.

Static ARP and static inspection ARP entries must be configured for hosts on untrusted ports. Otherwise, when DAI checks ARP packets from these hosts against entries in the ARP table, it will not find any entries for them, and the Ruckus device will not allow and learn ARP from an untrusted host.

1. Enter the global configuration mode.

   ```
   device# configure terminal
   ```

2. Define the ARP inspection entry in the static ARP table by mapping a device IP address with its MAC address.

   ```
   device(config)# arp 10.20.20.12 0000.0002.0003 inspection
   ```

   The ARP entry will be moved to the ARP table once the DAI receives a valid ARP packet.

The following example defines an inspection ARP entry in the static ARP table, specifies a device IP address 10.20.20.12 and MAC address 0000.0002.0003 pairing.

```
device# configure terminal
device(config)# arp 10.20.20.12 0000.0002.0003 inspection
```

Enabling DAI on a VLAN

Dynamic ARP inspection validates ARP packets from the untrusted ports of a VLAN.

1. **NOTE**
   DAI is disabled by default.

   Enter the global configuration mode.

   ```
   device# configure terminal
   ```

2. Enable DAI on an existing VLAN.

   ```
   device(config)# ip arp inspection vlan 2
   ```

The following example enable dynamic ARP inspection on an already configured VLAN 2.

```
device# configure terminal
device(config)# ip arp inspection vlan 2
```

Enabling ARP inspection trust on a port

Dynamic ARP inspection trust can be enabled on a port. The following task enables dynamic ARP inspection trust for Ethernet interface 1/1/4.

The default trust setting for a port is untrusted. For ports that are connected to host ports, leave their trust settings as untrusted. If the port is part of a LAG, enable ARP inspection trust on the LAG virtual interface.

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 1/1/4
   ```

3. Enter the **arp inspection trust** command to configure dynamic ARP inspection trust for the interface.

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

The following task enables dynamic ARP inspection trust for Ethernet interface 1/1/4.

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

Disabling syslog messages for DAI

You can disable syslog messages for Dynamic ARP Inspection. Syslog messages are enabled by default on FastIron devices.

Follow these steps to disable DAI messages.

1. Enter global configuration mode.

   ```
   device# configure terminal
   ```

2. To disable syslog messages enter the following command.

   ```
   device(config)# ip arp inspection syslog disable
   ```

   If you need to reenable DAI syslog messages, use the **no** form of the command.

The following example disables the syslog messages for DAI.

```
device# configure terminal
device(config)# ip arp inspection syslog disable
```

The following example reenables the syslog messages for DAI.

```
device# configure terminal
device(config)# no ip arp inspection syslog disable
```

Multi-VRF support for DAI

DAI supports Multi-VRF (Virtual Routing and Forwarding) instances. You can deploy multiple VRFs on a Ruckus Ethernet switch. Each VLAN having a Virtual Ethernet (VE) interface is assigned to a VRF.

You can enable DAI on individual VLANs and assign any interface as the ARP inspection trust interface. If an interface is a tagged port in this VLAN, you can turn on the trust port per VRF, so that traffic intended for other VRF VLANs will not be trusted.

NOTE
ICX 7150 devices do not support VRFs.

To configure DAI to support a VRF instance, do the following:

- Enable the **acl-per-port-per-vlan** setting. DAI requires the **acl-per-port-per-vlan** setting to be enabled.

```
device(config)# enable acl-per-port-per-vlan
Reload required. Please write memory and then reload or power cycle.
```

- Configure DAI on a VLAN using the **ip arp inspection vlan** *vlan-id* command.

```
device(config)# ip arp inspection vlan 2
```

- To add a static ARP inspection entry for a specific VRF, use the arp *ip-address mac-address* inspection command in the VRF CLI context.

```
device(config-vrf-one-ipv4)# arp 5.5.5.5 00a2.bbaa.0033 inspection
```

Enabling ARP inspection trust on a port for a nondefault VRF

Dynamic ARP inspection trust can be enabled on a port for a nondefault VRF instance. The following task enables dynamic ARP inspection trust for Ethernet interface 1/1/4 for VRF green.

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

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 1/1/4
```

3. Enter the **arp inspection trust** command with the **vrf** keyword, specifying a VRF, to configure dynamic ARP inspection trust on the interface for the specified VRF.

```
device(config-if-e10000-1/1/4)# arp inspection trust vrf green
```

The following task enables dynamic ARP inspection trust on Ethernet interface 1/1/4 for VRF green.

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

Displaying ARP inspection status and ports

To display the ARP inspection status for VLAN 2 and the trusted or untrusted port, enter the following command.

```
device# show ip arp inspection vlan 2
IP ARP inspection VLAN 2: Disabled
  Trusted Ports :   ethe 1/1/4
  Untrusted Ports : ethe 2/1/1 to 2/1/3 ethe 4/1/1 to 4/1/24 ethe 6/1/1 to 6/1/4 ethe 8/1/1 to 8/1/4
```

IP Addressing

IP addressing overview

IPv4 is the Internet protocol that is most commonly used currently throughout the world. IPv4 uses a 32-bit addressing system designed for use in packet-switched networks and is represented in a 4-byte dotted decimal format: x.x.x.x.

Ruckus Layer 3 devices support Internet Protocol version 4 (IPv4) and IPv6. IP support on Ruckus Layer 2 switches consists of host services and functionality to support management access and access to a default gateway.

NOTE
This section describes IPv4 addresses. For information about IPv6 addresses, refer to the *IPv6 addressing* chapter.

All Ruckus IP addresses can be configured and displayed in two formats.

- Classical subnet format with the IP address in dotted decimal with a subnet mask (for example, 192.168.1.1 255.255.255.0).
- Classless Interdomain Routing (CIDR) format with the IP address in dotted decimal but followed by a forward slash and a number representing the subnet mask (for example, 192.168.1.1/24).

You can use either format when configuring IP address information. IP addresses are displayed in classical subnet format by default, but you can change the display format to CIDR.

IP support on Ruckus devices operating at Layer 3 includes basic IP services such as Address Resolution Protocol (ARP), ICMP Router Discovery Protocol (IRDP), and Reverse ARP (RARP) and the following types of networking protocols:

- Dynamic routing protocols—All these protocols provide routes to the IP route table. You can use one or more of these protocols, in any combination. These protocols are disabled by default.
  - Routing Information Protocol (RIP)
  - Open Shortest Path First (OSPF)
  - Border Gateway Protocol version 4 (BGP4)
- Router redundancy protocols—Only one of these protocols can be enabled at the same time.
  - Virtual Router Redundancy Protocol Extended (VRRP-E)
  - Virtual Router Redundancy Protocol (VRRP)
- Multicast protocols—Ruckus devices can forward IP multicast packets.
  - Internet Group Management Protocol (IGMP)
  - Protocol Independent Multicast Dense (PIM-DM)
  - Protocol Independent Multicast Sparse (PIM-SM)

For more details on these protocols, see specific protocol chapters in this book. For multicast protocols, see the *Ruckus FastIron IP Multicast Configuration Guide*.

IP interfaces

IP addresses can be configured on a variety of interfaces.

Ruckus Layer 3 switches allow you to configure IP addresses on the following types of interfaces:

- Ethernet ports
- Virtual routing interfaces (used by VLANs to route among one another)
- Loopback interfaces
- GRE tunnels

Each IP address on a Layer 3 switch must be in a different subnet. You can have only one interface that is in a given subnet. For example, you can configure IP addresses 192.168.1.1/24 and 192.168.2.1/24 on the same Layer 3 device, but you cannot configure 192.168.1.1/24 and 192.168.1.2/24 on the same Layer 3 device.

You can configure multiple IP addresses on the same interface. The number of IP addresses you can configure on an individual interface depends on the Layer 3 device model. You can use any of the IP addresses you configure for Telnet, Web management, or SNMP access.

IP interfaces on Layer 2 switches

You can configure an IP address on a Ruckus Layer 2 switch for management access to the Layer 2 switch. An IP address is required for Telnet access, Web management access, and SNMP access.

You also can specify the default gateway for forwarding traffic to other subnets.

IP route table

The IP route table contains paths to IP destinations.

The IP route table can receive the paths from the following sources and it can contain leaked routes as well (from other VRFs).

- 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 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 through 255.
- When the software receives two or more best paths to a destination and the paths have the same metric (cost), the software can load share traffic among the paths based on destination host or network address (based on the configuration and the Layer 3 device model).

Here is an example of an entry in the IP route table.

```
Destination       NetMask          Gateway          Port   Cost   Type
10.1.0.0          255.255.0.0      10.1.1.2         1/1/1    2      R
```

Each IP route table entry contains the destination 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 IP metric (cost), and the type. The type indicates how the IP route table received the route.

ACLs and IP access policies to filter IP traffic

You can use several methods to filter IP traffic.

Ruckus Layer 3 switches provide two mechanisms for filtering IP traffic:

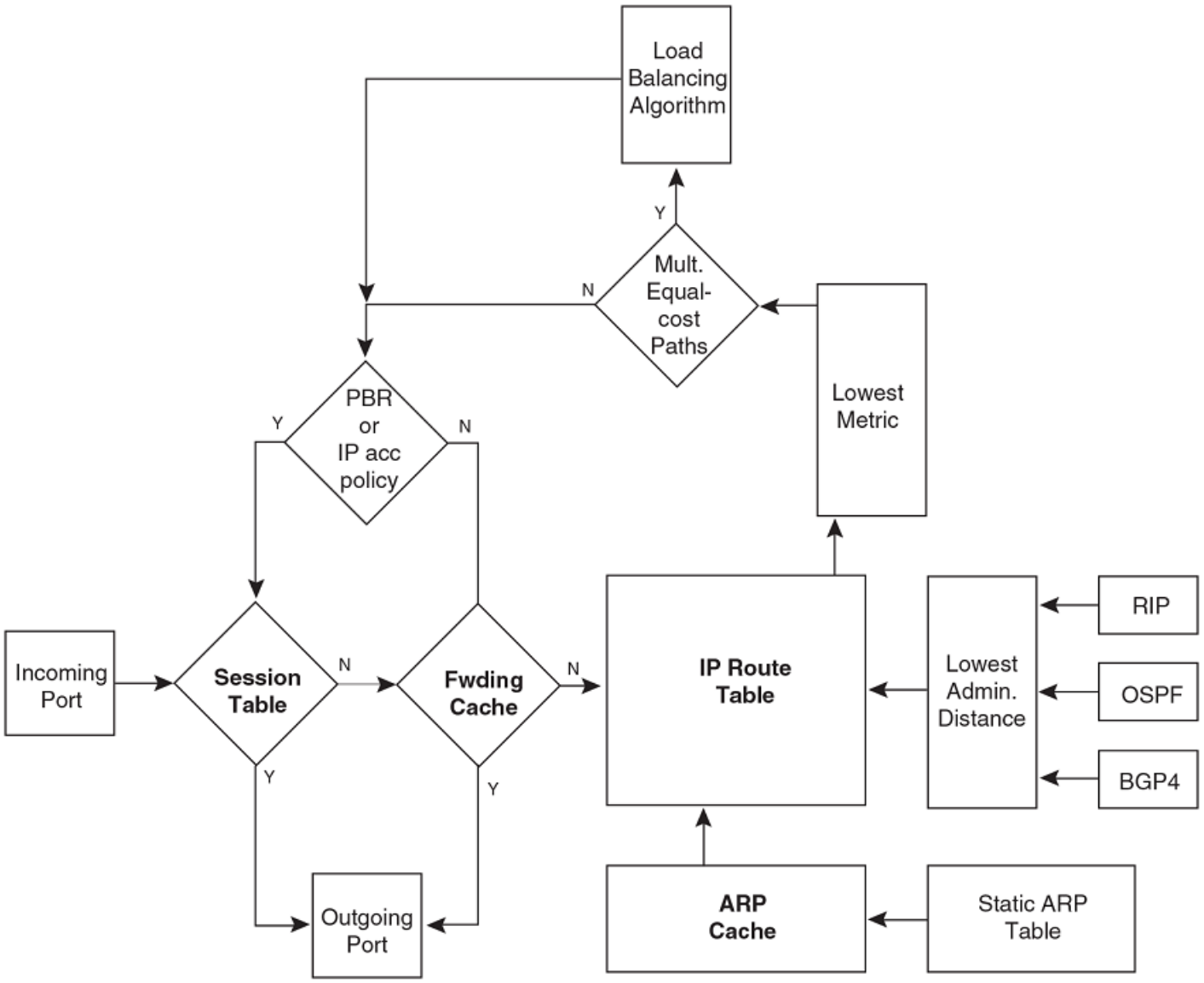
- Access Control Lists (ACLs)—Provide flexibility by providing the input to various other filtering mechanisms such as route maps, which are used by BGP4.
- IP access policies—Allow you to configure QoS based on sessions (Layer 4 traffic flows).

Only one of these filtering mechanisms can be enabled on a Ruckus device at a time. Ruckus devices can store forwarding information for both methods of filtering in the session table.

For configuration information, refer to the chapter "ACLs" in the *Ruckus FastIron Security Configuration Guide*.

IP packet flow through a Layer 3 device

FIGURE 3 IP Packet flow through a Ruckus Layer 3 device



1. When the Layer 3 device receives an IP packet, the device checks for filters on the receiving interface. The filter may be an Access Control List (ACL) or an IP access policy. If a deny filter on the interface denies the packet, the Layer 3 device discards the packet and performs no further processing, except for generating a Syslog entry and an SNMP message, if logging is enabled for the filter.
2. If the packet is not denied at the incoming interface, the Layer 3 device looks in the session table for an entry that has the same source IP address and TCP or UDP port as the packet. If the session table contains a matching entry, the Layer 3 device immediately forwards the packet, by addressing it to the destination IP address and TCP or UDP port listed in the session table entry and sending the packet to a queue on the outgoing ports listed in the session table. The device selects the queue based on the Quality of Service (QoS) level associated with the session table entry.

3. If the session table does not contain an entry that matches the packet source address and TCP or UDP port, the Layer 3 device looks in the IP forwarding cache for an entry that matches the packet destination IP address. If the forwarding cache contains a matching entry, the device forwards the packet to the IP address in the entry. The device sends the packet to a queue on the outgoing ports listed in the forwarding cache. The device selects the queue based on the Quality of Service (QoS) level associated with the forwarding cache entry.
4. If the IP forwarding cache does not have an entry for the packet, the Layer 3 device checks the IP route table for a route to the packet destination. If the IP route table has a route, the device makes an entry in the session table or the forwarding cache, and sends the route to a queue on the outgoing ports:
   - If the running-config contains an IP access policy for the packet, the software makes an entry in the session table. The Layer 3 device uses the new session table entry to forward subsequent packets from the same source to the same destination.
   - If the running-config does not contain an IP access policy for the packet, the software creates a new entry in the forwarding cache. The Layer 3 device uses the new cache entry to forward subsequent packets to the same destination.

IP forwarding cache

The IP forwarding cache contains entries for IP destinations, and provides a fast-path mechanism for forwarding IP packets.

When a Ruckus Layer 3 device has completed the processing and addressing for a packet and is ready to forward the packet, the device checks the IP forwarding cache for an entry to the packet destination:

- If the cache contains an entry with the destination IP address, the device uses the information in the entry to forward the packet out the ports listed in the entry. The destination IP address is the address of the packet final destination. The port numbers are the ports through which the destination can be reached.
- If the cache does not contain an entry and the traffic does not qualify for an entry in the session table instead, the software can create an entry in the forwarding cache.

Each entry in the IP forwarding cache has an age timer. If the entry remains unused for ten minutes, the software removes the entry. The age timer is not configurable.

Each entry in the IP forwarding cache has an age timer. The age timer is not configurable.

Here is an example of an entry in the IP forwarding cache.

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

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 Layer 3 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.

NOTE
You cannot add static entries to the IP forwarding cache. You can increase the number of entries the cache can contain.

Layer 4 session table

The Layer 4 session provides a fast path for forwarding packets. A session is an entry that contains complete Layer 3 and Layer 4 information for a flow of traffic. Layer 3 information includes the source and destination IP addresses. Layer 4 information includes the source and destination TCP and UDP ports. For comparison, the IP forwarding cache contains the Layer 3 destination address but does not contain the other source and destination address information of a Layer 4 session table entry.

The Layer 2 switch or Layer 3 switch selects the session table instead of the IP forwarding table for fast-path forwarding for the following features:

- Layer 4 Quality-of-Service (QoS) policies
- IP access policies

To increase the size of the session table, refer to the section "Displaying and modifying system parameter default settings" in the *Ruckus FastIron Layer 2 Switching Configuration Guide*. The ip-qos-session parameter controls the size of the session table.

Encapsulation type and MTU packet parameters

Several packet parameters can be configured for your network.

You can configure the following packet parameters on Layer 3 devices. These parameters control how the Layer 3 device sends IP packets to other devices on an Ethernet network. The Layer 3 device always places IP packets into Ethernet packets to forward them on an Ethernet port.

- Encapsulation type—The format for the Layer 2 packets within which the Layer 3 device sends IP packets.
- Maximum Transmission Unit (MTU)—The maximum length of IP packet that a Layer 2 packet can contain.

Encapsulation type

The Layer 3 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 source address of a Layer 2 packet is the MAC address of the Layer 3 switch interface sending the packet. The destination address can be one of the following:

- The MAC address of the IP packet destination. In this case, the destination device is directly connected to the Layer 3 switch.
- The MAC address of the next-hop gateway toward the packet destination.
- An Ethernet broadcast address.

The entire IP packet, including the source and destination address and 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. Ruckus Layer 3 devices use Ethernet II by default. You can change the IP encapsulation to Ethernet SNAP on individual ports if needed.

MTU

The Maximum Transmission Unit (MTU) is the maximum size an IP packet can be when encapsulated in a Layer 2 packet. If an IP packet is larger than the MTU allowed by the Layer 2 packet, the Layer 3 switch fragments the IP packet into multiple parts that will fit into the 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.

When packets are larger than the default MTU, they are referred to as jumbo packets. Ruckus devices contain the following enhancements to jumbo packet support:

- Hardware forwarding of Layer 3 jumbo packets—Layer 3 IP unicast jumbo packets received on a port that supports the frame MTU size and forwarded to another port that also supports the frame MTU size are forwarded in hardware. Previous releases support hardware forwarding of Layer 2 jumbo frames only.

- ICMP unreachable message if a frame is too large to be forwarded—If a jumbo packet has the Do not Fragment (DF) bit set, and the outbound interface does not support the packet MTU size, the Ruckus device sends an ICMP unreachable message to the device that sent the packet.

  NOTE
  These enhancements apply only to transit traffic forwarded through the Ruckus device.

You can change the MTU globally or an individual ports:

- Global MTU—The default MTU value depends on the encapsulation type on a port and is 1500 bytes for Ethernet II encapsulation and 1492 bytes for SNAP encapsulation.
- Port MTU—A port default MTU depends on the encapsulation type enabled on the port.

Globally changing the MTU

You can increase the MTU size to accommodate jumbo packet sizes up to 10,200 bytes.

To globally enable jumbo support on all ports of a FastIron device, enter commands such as the following.

```
device(config)# jumbo
device(config)# write memory
device(config)# end
device# reload
```

NOTE
You must save the configuration change and then reload the software to enable jumbo support.

Basic IP configuration

IP is enabled by default. Basic configuration consists of adding IP addresses for Layer 3 devices, configuring DNS, enabling a route exchange protocol, such as the Routing Information Protocol (RIP), and enabling and configuring routing protocols.

NOTE
The term, device, used in this chapter can represent a switch operating at Layer 3 or a router. A switch operating at Layer 2 can be configured with an IP address for management access through the network and to specify the default gateway.

To configure RIP and other routing protocols, see the appropriate chapter in this book.

There are a number of IP parameters and features that can be configured. Some IP parameters can be configured globally while others can be configured on individual interfaces. Some parameters that are configured globally can be overridden for individual interfaces. Use the information in this chapter if you need to change some of the IP parameters from their default values, to configure specific IP features, or if you want to view configuration information or statistics.

When IP parameter changes take effect

Most IP parameters described in this chapter are dynamic. They take effect immediately, as soon as you enter the CLI command or select the Web Management Interface option. You can verify that a dynamic change has taken effect by displaying the running-config file. To display the running-config, enter the **show running-config** or **write terminal** command at any CLI prompt. (You cannot display the running-config from the Web Management Interface.)

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

- To save configuration changes to the startup-config file, enter the **write memory** command from the Privileged EXEC level or any configuration level of the CLI.
- To save the configuration changes using the Web Management Interface, select the **Save link** at the bottom of the dialog. Select **Yes** when prompted to save the configuration change to the startup-config file on the device flash memory. You also can access the dialog for saving configuration changes by clicking on **Command** in the tree view, then clicking on **Save to Flash**.

Changes to memory allocation require you to reload the software after you save the changes to the startup-config file. When reloading the software is required to complete a configuration change described in this chapter, the procedure that describes the configuration change includes a step for reloading the software.

Assigning an IP address to a loopback interface

Loopback interfaces can be assigned an IPv4 address using this task.

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 a Layer 3 switch and other devices. You can configure up to eight loopback interfaces on a chassis Layer 3 switch devices. You can configure up to four loopback interfaces on a compact Layer 3 switch.

NOTE
If you configure a Layer 3 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 Layer 3 device.

1. Enter the **show default values** command to view the maximum number of IP addresses you can assign to a loopback interface for the current device.

```
device# show default values

sys log buffers:50  mac age time:300 sec   telnet sessions:5
ip arp age:10 min   bootp relay max hops:4 ip ttl:64 hops
ip addr per intf:24
(output truncated)
```

   From the output, this device can support up to 24 IP addresses added to each loopback interface.

2. Enter global configuration mode.

```
device# configure terminal
```

3. Enter interface configuration mode and specify a loopback interface.

```
device(config)# interface loopback 1
```

4. Assign an IP address to the loopback interface.

```
device(configconfig-lbif-1)# ip address 10.0.0.1/24
```

5. Enter the **show ip interface** command to display IP interface information.

```
device(config-lbif-1)# show ip interface

Interface         IP-Address      OK?  Method    Status            Protocol
Ethernet 1/1/1    10.95.6.173     YES  NVRAM     up                up
Ethernet 1/1/2    10.3.3.3        YES  manual    up                up
Loopback 1        10.0.0.1        YES  NVRAM     up                up
```

The following example assigns an IP address to a loopback interface.

```
device# configure terminal
device(config)# interface loopback 1
device(config-lbif-1)# ip address 10.0.0.1/24
```

Assigning an IPv4 address to an Ethernet port

Ethernet ports can be assigned an IPv4 address using this procedure.

NOTE
All physical IP interfaces on Ruckus FastIron Layer 3 devices share the same MAC address. For this reason, if more than one connection is made between two devices, one of which is a Ruckus FastIron Layer 3 device, Ruckus recommends the use of virtual interfaces. It is not recommended to connect two or more physical IP interfaces between two routers.

1. Enter global configuration mode.

```
device# configure terminal
```

2. Enter interface configuration mode and specify an Ethernet interface.

```
device(config)# interface ethernet 1/1/1
```

3. Assign an IP address to Ethernet port 1/1/1.

```
device(config-if-e10000-1/1/1)# ip address 10.45.6.1 255.255.255.0
```

4. Assign a second IP address to Ethernet port 1/1/1 using CIDR.

```
device(config-if-e10000-1/1/1)# ip address 10.45.6.2/24 secondary
```

5. Enter the **show ip interface** command to display IP interface information.

```
device(config-if-e10000-1/1/1)# show ip interface

Interface         IP-Address      OK?  Method    Status            Protocol
Ethernet 1/1/1    10.95.6.173     YES  NVRAM     up                up
Ethernet 1/1/2    10.3.3.3        YES  manual    up                up
Loopback 1        10.2.3.4        YES  NVRAM     down              down
```

The following example assigns IP addresses to an Ethernet interface.

```
device# configure terminal
device(config)# interface ethernet 1/1/1
device(config-if-e10000-1/1/1)# ip address 10.45.6.1 255.255.255.0
device(config-if-e10000-1/1/1)# ip address 10.45.6.2/24 secondary
```

Assigning an IP address to a virtual interface

Virtual interfaces can be assigned an IPv4 address using this task.

A virtual interface is a logical port associated with a Layer 3 Virtual LAN (VLAN) configured on a Layer 3 switch. You can configure routing parameters on the virtual interface to enable the Layer 3 switch to route protocol traffic from one Layer 3 VLAN to the other, without using an external router.

You can configure IP routing interface parameters on a virtual interface.

NOTE
A Layer 3 switch uses the lowest MAC address on the device (the MAC address of port 1 or 1/1/1) as the MAC address for all ports within all virtual interfaces you configure on the device.

1. Enter global configuration mode.

```
device# configure terminal
```

2. Create a VLAN.

```
device(config)# vlan 2 name Marketing
```

VLAN 2 is named for the Marketing department.

3. Add a range of untagged Ethernet ports to the VLAN.

```
device(config-vlan-2)# untag ethernet 1 to 4
```

4. Create virtual interface 1 as the routing interface for the VLAN.

```
device(config-vlan-2)# router-interface ve 1
```

5. Enter interface configuration mode for the virtual interface.

```
device(config-vlan-2)# interface ve 1
```

6. Assign an IP address to the virtual interface.

```
device(config-vif-1)# ip address 10.1.2.1/24
```

The following example adds a virtual interface to a VLAN and configures an IP address on the virtual interface.

```
device# configure terminal
device(config)# vlan 2 name Marketing
device(config-vlan-2)# untag ethernet 1 to 4
device(config-vlan-2)# router-interface ve 1
device(config-vlan-2)# interface ve 1
device(config-vif-1)# ip address 10.1.2.1/24
```

Domain Name System (DNS)

The Domain Name System (DNS) resolver is a feature in a Layer 2 or Layer 3 device that sends and receives queries to and from the DNS server on behalf of a client.

You can create a list of domain names that can be used to resolve host names. This list can have more than one domain name. When a client sends a DNS query, all hosts within the domains in the list can be recognized and queries can be sent to any domain on the list.

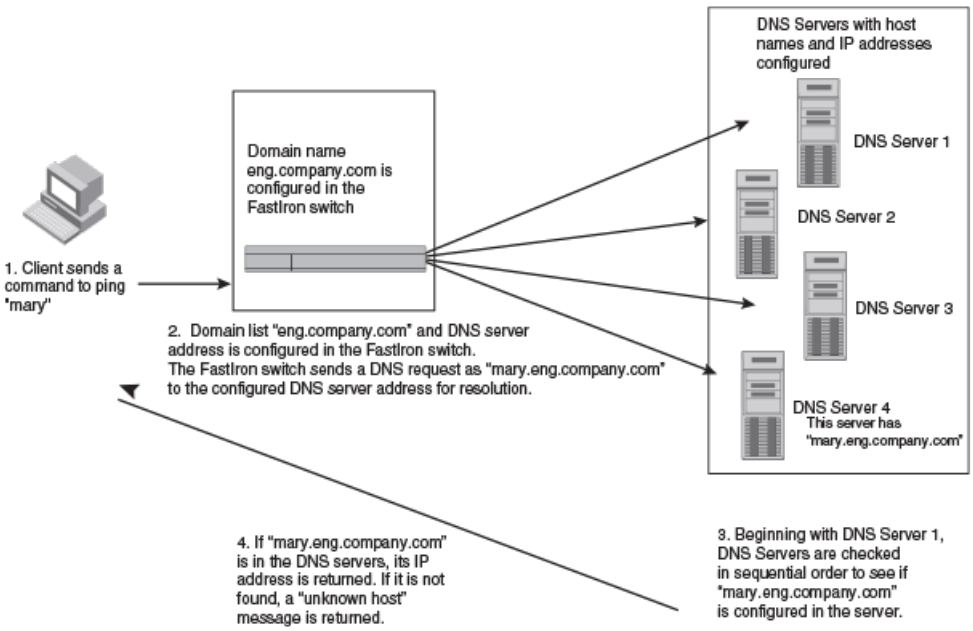
After you define a domain name, the Ruckus device automatically appends the appropriate domain to a host and forwards it to the DNS servers for resolution.

For example, if the domain "ds.company.com" is defined on a Layer 2 or Layer 3 switch and you want to initiate a ping to "mary", you must reference only the host name instead of the host name and its domain name. For example, you could use the following command to initiate the ping.

```
device:> ping mary
```

The device qualifies the host name by appending a domain name (for example, mary.ds1.company.com). This qualified name is sent to the DNS server for resolution. If there are four DNS servers configured, it is sent to the first DNS server. If the host name is not resolved, it is sent to the second DNS server. If a match is found, a response is sent back to the client with the host IP address. If no match is found, an "unknown host" message is returned.

FIGURE 4 DNS resolution with one domain name



Configuring DNS

You can configure a Domain Name System (DNS) domain name and server addresses to resolve host names.

Using DNS can simplify some network operations through the use of a domain name instead of the numbers in an IP address.

1. Enter global configuration mode.

```
device# configure terminal
```

2. Configure the Ruckus device to recognize up to four DNS servers.

```
device(config)# ip dns server-address 10.157.22.199 10.96.7.15 10.95.7.25 10.98.7.15
```

   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 DNS address is queried (also up to three times). This process continues for each defined DNS address until the query is resolved. The order in which the default DNS addresses are polled is the same as the order in which you enter them.

   Alternatively, you can configure the DNS servers on separate lines.

3. Configure a domain name to resolve host names.

```
device(config)# ip dns domain-list swcompany.com
```

   If you want to use more than one domain name to resolve host names, you can create a list of domain names. The domain names are tried in the order in which they are entered.

4. Return to EXEC configuration mode.

```
device(config)# exit
```

5. Initiate a trace route using a DNS name.

```
device# traceroute utahtown

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
```

   This example shows how to trace the route from a Ruckus Layer 3 device to a remote server identified as utahtown on domain swcompany.com. Assuming that utahtown@company.com domain is already defined on the Layer 3 switch, you need to enter only the host name, utahtown.

The following example shows to how to configure four DNS domain names and two servers.

```
device# configure terminal
device(config)# ip dns server-address 10.157.22.199
device(config)# ip dns server-address 10.96.7.15
device(config)# ip dns server-address 10.95.7.25
device(config)# ip dns server-address 10.98.7.15
device(config)# ip dns domain-list swcompany.com
device(config)# ip dns domain-list westcoastcompany.com
```

ICMP

Internet Control Message Protocol (ICMP) is used to send error and operational messages and is enabled by default.

The Ruckus Layer 3 device can send the following types of ICMP messages:

- Echo messages (ping messages)—The Layer 3 device replies to IP pings from other IP devices.
- Destination Unreachable messages—If the Layer 3 device receives an IP packet that it cannot deliver to its destination, the Layer 3 switch discards the packet and sends a message back to the device that sent the packet to the Layer 3 switch. The message informs the device that the destination cannot be reached by the Layer 3 switch.

ICMP is used to send error messages and operational information indicating, for example, that a requested service is not available or that a hoat is not available. ICMP differs from transport protocols such as TCP or UDP because data is not forwarded between systems. Diagnostic tools such as ping and traceroute.

ICMP echo messages

By default, Ruckus devices are enabled to reply to broadcast ICMP echo messages. Broadcast echo packets are ping requests and are triggered by **ping** or **traceroute** commands. You can disable the response to echo packets.

ICMP destination Unreachable messages

By default, when a Ruckus 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 Ruckus device response to the following types of ICMP Unreachable messages:

- **Host**—The destination network or subnet of the packet is directly connected to the Ruckus device, but the host specified in the destination IP address of the packet is not on the network.
- **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.
- **Administration**—The packet was dropped by the Ruckus 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 Ruckus device cannot forward the packet without fragmenting it.
- **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 Ruckus device, which in turn sends the message to the host that sent the packet.
- **Source-route-fail**—The device received a source-routed packet but cannot locate the next-hop IP address indicated in the packet Source-Route option.

Disabling ICMP messages

By default, Ruckus devices are enabled to respond to broadcast ICMP echo packets or ICMP Unreachable messages.

You can selectively disable the following types of Internet Control Message Protocol (ICMP) messages:

- Echo messages (ping messages)
- Destination Unreachable messages—you can disable all types of ICMP unreachable messages, or individual types.

NOTE
Disabling an ICMP Unreachable message type does not change the Ruckus device ability to forward packets. Disabling ICMP Unreachable messages prevents the device from generating or forwarding the Unreachable messages.

1. Enter global configuration mode.

```
device# configure terminal
```

2. To disable the response to broadcast ICMP echo packets (ping requests)

```
device(config)# no ip icmp echo broadcast-request
```

3. To disable all ICMP Unreachable messages.

```
device(config)# no ip icmp unreachable
```

4. To disable ICMP Host Unreachable messages.

```
device(config)# no ip icmp unreachable host
```

For more options, see the **ip icmp unreachable** command in the *Ruckus FastIron Command Reference*.

Enabling ICMP redirect messages

You can enable and disable IPv4 ICMP redirect messages globally or on individual Virtual Ethernet (VE) interfaces but not on individual physical interfaces.

By default, IP ICMP redirect globally is disabled and a Ruckus Layer 3 device does not send an ICMP redirect message to the source of a misdirected packet, but it does forward the packet to the appropriate router. To enable ICMP redirect messages globally, enter the following commands.

1. Enter global configuration mode.

```
device# configure terminal
```

2. To enable ICMP redirect messages choose one of the following options.
   - Enable ICMP redirect messages globally, go to Step 3.
   - Enable ICMP redirect messages on a specific interface, go to Step 4.
3. To enable ICMP redirect messages globally.

```
device(config)# ip icmp redirect
```

ICMP redirect messages are enabled globally on all interfaces.

4. To enable ICMP redirect messages on a specific virtual interface, enter the following command at the configuration level for the virtual interface:

```
device(config-vlan-10)# interface ve 10
device(config-vif-10)# ip icmp redirect
```

You can enable ICMP redirect messages on a specific virtual interface, but not on a physical interface.

ICMP Router Discovery Protocol configuration

The ICMP Router Discovery Protocol (IRDP) is used by Ruckus Layer 3 devices to advertise the IP addresses of its router interfaces to directly attached hosts.

IRDP is disabled by default. You can enable the feature on a global basis or on an individual port basis:

- If you enable the feature globally, all ports use the default values for the IRDP parameters.
- If you leave the feature 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 Layer 3 device periodically sends Router Advertisement messages out the IP interfaces on which the feature is enabled. The messages advertise the IP addresses of the device to directly attached hosts who listen for the messages. In addition, hosts can be configured to query the Layer 3 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 on the Ruckus Layer 3 device, the device responds to the Router Solicitation messages. Some clients interpret this response to mean that the Layer 3 device is the default gateway. If another router is actually the default gateway for these clients, leave IRDP disabled on the Ruckus Layer 3 device.

IRDP parameters

IRDP uses the following parameters. If you enable IRDP on individual ports instead of enabling the feature globally, you can configure these parameters on an individual port basis:

- **Packet type**—The Layer 3 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 Layer 3 device sends the Router Advertisement messages every 450 - 600 seconds by default. The time within this interval that the 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 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 0-4294967296. The default is 0.

Enabling IRDP

ICMP Router Discovery Protocol (IRDP) can be enabled globally or on a specific interface.

When IRDP is enabled globally, the parameters are set to default values and cannot be changed. If you enable IRDP on a specific interface, you can configure the following parameters:

- Packet type
- Maximum message interval
- Minimum message interval
- Hold time
- Preference

For more details about these IRDP parameters, see the IRDP parameters section of the ICMP Router Discovery Protocol configuration on page 54 topic.

1. Enter global configuration mode.

```
device# configure terminal
```

2. To enable IRDP choose one of the following options.
   - Enable IRDP globally, go to Step 3.
   - Enable IRDP on a specific interface, go to Step 4.
3. Enable IRDP in global configuration mode.

   ```
   device(config)# ip irdp
   ```

   IRDP is enabled 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.
4. Enable IRDP on a specific interface.

   ```
   device(config)# interface ethernet 1/1/3
   device(config-if-1/1/3)# ip irdp maxadvertinterval 400
   ```

   After entering interface configuration mode for Ethernet interface 1/1/3, IRDP is enabled for the interface and the maximum advertisement interval is set to 400 seconds.

Configuring IP follow on a virtual routing interface

Configuring IP Follow can help conserve IP address space.

IP Follow allows multiple virtual routing interfaces to share the same IP address. With this feature, one virtual routing interface is configured with an IP address, while the other virtual routing interfaces are configured to follow that IP address; they use (follow) the same virtual routing interface that owns that IP address.

The following limitations apply to the IP Follow feature:

- When configuring IP Follow, the primary virtual routing interface should not have ACL or DoS Protection configured. It is recommended that you create a dummy virtual routing interface as the primary and use the IP follow virtual routing interface for the network.
- Global Policy Based Routing is not supported when IP Follow is configured.
- IPv6 does not support IP Follow.
- Ruckus FastIron devices support IP Follow with OSPF and VRRP protocols only.

1. Enter global configuration mode.

   ```
   device# configure terminal
   ```

2. Create a VLAN.

   ```
   device(config)# vlan 2 name Marketing
   ```

   VLAN 2 is named for the Marketing department.
3. Add a range of untagged Ethernet ports to the VLAN.

   ```
   device(config-vlan-2)# untag ethernet 1 to 4
   ```

4. Create virtual interface 1 as the routing interface for the VLAN.

   ```
   device(config-vlan-2)# router-interface ve 1
   ```

5. Enter interface configuration mode for the virtual interface.

```
device(config-vlan-2)# interface ve 1
```

6. Assign an IP address to the virtual interface.

```
device(config-vif-1)# ip address 10.1.2.1/24
```

7. Configure a second virtual interface.

```
device(config-vif-1)# interface ve 2
```

8. Configure the second virtual interface to follow the IP address assigned to virtual interface 1.

```
device(config-vif-2)# ip follow ve 1
```

9. Configure a third virtual interface.

```
device(config-vif-1)# interface ve 3
```

10. Configure the third virtual interface to follow the IP address assigned to virtual interface 1.

```
device(config-vif-3)# ip follow ve 1
```

The following example allows virtual interface 2 and 3 to follow the IP address assigned to virtual interface 1. Virtual routing interface 2 and 3 do not have their own IP subnet addresses, but share the IP address of virtual routing interface 1.

```
device(config)# vlan 2 name Marketing
device(config-vlan-2)# untag ethernet 1 to 4
device(config-vlan-2)# router-interface ve 1
device(config-vlan-2)# interface ve 1
device(config-vif-1)# ip address 10.10.2.1/24
device(config-vif-1)# interface ve 2
device(config-vif-2)# ip follow ve 1
device(config-vif-2)# interface ve 3
device(config-vif-3)# ip follow ve 1
```

IP address replacement

An interface supports multiple primary address configuration. However, you can configure only one primary IP address in a subnet. Beginning with Ruckus FastIron release 08.0.50, you can configure the primary IP address on a physical interface (Ethernet), management interface, virtual interface (VE or loopback), or a tunnel interface even if the primary IP address was already configured.

When you configure an IP address, using the **ip address** command with the **replace** option, the existing IP address is removed and the new IP address is applied to the interface. After the IP address replacement, you must re-establish all the Telnet and Secure Shell (SSH) sessions because the current sessions get either terminated or timed out.

Limitations and restrictions for IP address replacement

- The IP address replacement feature is not supported for an IPv6 address configuration.
- The IP address replacement feature is supported on a Ruckus FastIron router image only.
- You cannot change the subnet mask by using the **replace** parameter.
- You can replace IP addresses only on the management interface and data ports.
- You cannot replace a secondary IP address. You can only replace the primary IP address.

Replacing an IP address

You must remove all secondary IP addresses, if they exist, before replacing the existing primary IP address with a new IP address.

1. **NOTE**
   The device does not prompt, if no primary IP address matching subnet of the new IP address is already configured on interface and the user use the **replace** option. When the **ip address** command is used with the **replace** option, the new IP address is configured.

   Enter global configuration mode.

   ```
   device# configure termainal
   ```

2. Use the **interface** command to enter interface configuration mode.

   ```
   device (config)# interface ethernet 1/1/21
   ```

3. Use the **ip address** command to replace an existing primary IP address with a new IP address. Note that when you choose the replace option to remove the existing IP address on an interface, the action cannot be undone.

   ```
   device(config-if-e1000-1/1/21)# ip address 10.1.1.1/24 replace
   ```

4. Use the **show running interface** command to display the newly configured IP address on the interface.

   ```
   device(config-if-e1000-1/1/21)# show running interface ethernet 1/1/21
   ```

   The **replace** option is not displayed in running or startup configurations.

The following example shows how to replace the primary IP address of an interface.

```
device# configure terminal
device(config)# interface ethernet 1/1/21
device(config-if-e1000-1/1/21)# ip address 10.1.1.1/24 replace
```

The following example shows how to display the primary IP address of an interface.

```
device(config-if-e1000-1/1/21)# show running interface ethernet 1/1/21
 interface ethernet 1/1/21
 ip address 11.1.1.2 255.255.255.0!
```

Changing packet encapsulation type and MTU

The MTU and the encapsulation type of an IP packet can be changed from the default values.

The entire IP packet, including the source and destination address and 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 (encapsulation types) 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. Ruckus Layer 3 devices use Ethernet II by default. You can change the IP encapsulation to Ethernet SNAP on individual ports if needed.

You can change the MTU globally, or an individual ports:

- Global MTU—The default MTU value depends on the encapsulation type on a port and is 1500 bytes for Ethernet II encapsulation and 1492 bytes for SNAP encapsulation.
- Port MTU—A port default MTU depends on the encapsulation type enabled on the port.

The default MTU is 1500 bytes for Ethernet II packets and 1492 for Ethernet SNAP packets. You can increase the MTU size to accommodate jumbo packet sizes up to 10,200 bytes. When jumbo mode is enabled, the maximum Ethernet MTU sizes are as follows:

- 10,178 bytes - The maximum for Ethernet II encapsulation (Default MTU: 9216)
- 10,174 bytes - The maximum for SNAP encapsulation (Default MTU: 9216)

NOTE
For ICX 7850 devices, the maximum jumbo frame size supported is 9380. When jumbo mode is enabled, the maximum ethernet MTU size is 9358 bytes.

In this task the encapsulation type of Ethernet SNAP is configured on Ethernet interface 1/1/1 and jumbo packet sizes are allowed.

1. Enter global configuration mode.

```
device# configure terminal
```

2. Increase the MTU size to allow jumbo packet sizes.

```
device(config)# jumbo
```

3. Enter interface configuration mode.

```
device(config)# interface ethernet 1/1/1
```

   The interface is configured as Ethernet interface 1/1/1.

4. Change the encapsulation type to Ethernet SNAP

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

   All devices connected to the Layer 3 device port must use the same encapsulation type.

5. Return to global configuration mode to save the configuration.

```
device(config-if-e1000-1/1/1)# exit
```

6. Save the configuration to enable the global MTU change to allow jumbo packet sizes.

```
device(config)# write memory
```

7. Return to privileged EXEC mode.

```
device(config)# end
```

8. Reload the software to enable the MTU parameter change.

```
device# reload
```

The following example configures the packet encapsulation type to be Ethernet SNAP and allows jumbo packets sizes.

```
device# configure terminal
device(config)# jumbo
device(config)# interface ethernet 1/1/1
device(config-if-e1000-1/1/1)# ip encapsulation snap
device(config-if-e1000-1/1/1)# exit
device(config)# write memory
device(config)# end
device# reload
```

IPv4 point-to-point GRE tunnels

This section describes support for point-to-point Generic Routing Encapsulation (GRE) tunnels and how to configure them on a device.

GRE tunnels support includes the following:

- IPv4 over GRE tunnels. IPv6 over GRE tunnels is not supported.
- Static and dynamic unicast routing over GRE tunnels
- Multicast routing over GRE tunnels
- Hardware forwarding of IP data traffic across a GRE tunnel.
- Path MTU Discovery (PMTUD)

NOTE
ICX 7150 devices do not support Generic Routing Encapsulation (GRE) tunnels.

GRE tunnel overview

Generic Routing Encapsulation is described in RFC 2784. Generally, GRE provides a way to encapsulate arbitrary packets (payload packet) inside of a transport protocol, and transmit them from one tunnel endpoint to another. The payload is encapsulated in a GRE packet. The resulting GRE packet is then encapsulated in a delivery protocol, then forwarded to the tunnel destination. At the tunnel destination, the packet is decapsulated to reveal the payload. The payload is then forwarded to its final destination.

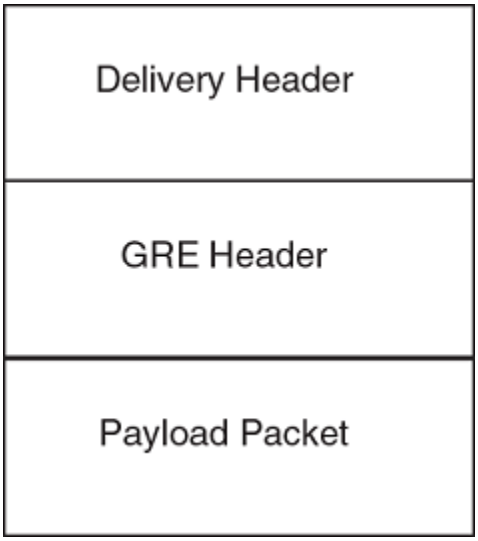
Ruckus devices allow the tunneling of packets of the following protocols over an IPv4 network using GRE:

- OSPF V2
- BGP4
- RIP V1 and V2

GRE packet structure and header format

The following figure displays the basic format of the GRE encapsulated packet. The payload packet is encapsulated inside a GRE packet which is carried inside a delivery protocol packet.

FIGURE 5 GRE encapsulated packet structure



The GRE header, displayed in the following figure, contains a series of fields.

FIGURE 6 GRE header format

Checksum	Reserved0	Ver	Protocol Type	Checksum (optional)	Reserved (optional)

- Checksum—1 bit. This field is assumed to be zero in this version. If set to 1, this means that the Checksum **(optional)** and Reserved **(optional)** fields are present and the **Checksum (optional)** field contains valid information.
- Reserved0—12 bits. If bits 1 - 5 are non-zero, then a receiver must discard the packet unless RFC 1701 is implemented. Bits 6 - 12 are reserved for future use and must be set to zero in transmitted packets. This field is assumed to be zero in this version.
- Ver—3 bits. The GRE protocol version. This field must be set to zero in this version.
- Protocol Type—16 bits. The Ethernet protocol type of the packet, as defined in RFC 1700.
- Checksum (optional)—16 bits. This field is optional. It contains the IP checksum of the GRE header and the payload packet.
- Reserved (optional)—16 bits. This field is optional. It is reserved for Ruckus internal use.

Restrictions for GRE tunnel configuration

When GRE is enabled on a Layer 3 switch, the following features are not supported on Virtual Ethernet (VE) ports, VE member ports (ports that have IP addresses), and GRE tunnel loopback ports

- ACL logging
- ACL statistics (also called ACL counting)

- MAC address filters
- IPv6 filters

NOTE
The above features are supported on VLANs that do not have VE ports.

When multiple IP addresses are configured on a tunnel source, the primary address of the tunnel is always used for forming the tunnel connections. Therefore, carefully check the configurations when configuring the tunnel destination.

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 FastIron 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 FastIron device cannot learn the tunnel destination route through OSPF. This could cause the system to become unstable.

The tunnel destination cannot be resolved to the tunnel itself or any other local tunnel. This is called recursive routing. This scenario would cause the tunnel interface to flap and the Syslog message TUN-RECURSIVE-DOWN to be logged. To resolve this issue, create a static route for the tunnel destination.

GRE MTU configuration considerations

When jumbo is enabled, the default Ethernet MTU size is 9216 bytes. The maximum Ethernet MTU size is 10178 bytes. The MTU of the GRE tunnel is compared with the outgoing packet before the packet is encapsulated. After encapsulation, the packet size increases by 24 bytes. Therefore, when changing the GRE tunnel MTU, set the MTU to at least 24 bytes less than the IP MTU of the outgoing interface. If the MTU is not set to at least 24 bytes less than the IP MTU, the size of the encapsulated packet will exceed the IP MTU of the outgoing interface. This will cause the packet to either be sent to the CPU for fragmentation, or the packet will be dropped if the DF (Do-Not-Fragment) bit is set in the original IP packet, and an ICMP message is sent.

NOTE
The fragmentation behavior depends on the mtu-exceed setting on the router.

GRE tunnel parameters

GRE link keepalive

When GRE tunnels are used in combination with static routing or policy-based routing, and a dynamic routing protocol such as RIP, BGP, or OSPF is not deployed over the GRE tunnel, a configured tunnel does not have the ability to bring down the line protocol of either tunnel endpoint, if the far end becomes unreachable. Traffic sent on the tunnel cannot follow alternate paths because the tunnel is always UP. To avoid this scenario, enable GRE link keepalive, which will maintain or place the tunnel in an UP or DOWN state based upon the periodic sending of keepalive packets and the monitoring of responses to the packets. If the packets fail to reach the tunnel far end more frequently than the configured number of retries, the tunnel is placed in the DOWN state.

To enable GRE link keepalive, configure it on one end of the tunnel and ensure the other end of the tunnel has GRE enabled.

NOTE
Keepalives are not supported when a tunnel interface is not within the default-VRF.

Maximum number of GRE tunnels supported

Use the following table to determine how many GRE tunnels are supported. You can configure the device to support up to the maximum number of GRE tunnels as displayed in the following table.

Device	Max # of GRE tunnels	Default # of GRE tunnels
ICX 7250	8	8
ICX 7450	64	16
ICX 7650	64	16
ICX 7750	64	16
ICX 7850	64	16

NOTE
You must save the configuration using the **write memory** command and reload the software to ensure the change is made.

GRE MTU

You can set an MTU value for packets entering the tunnel. When jumbo packet size is enabled, the default Ethernet MTU size is 9216 bytes. The maximum Ethernet MTU size is 10178 bytes. The MTU of the GRE tunnel is compared with the outgoing packet before the packet is encapsulated. After encapsulation, the packet size increases by 24 bytes. Therefore, when changing the GRE tunnel MTU, set the MTU to at least 24 bytes less than the IP MTU of the outgoing interface. If the MTU is not set to at least 24 bytes less than the IP MTU, the size of the encapsulated packet will exceed the IP MTU of the outgoing interface. This will cause the packet to either be sent to the CPU for fragmentation, or the packet will be dropped if the DF (Do-Not-Fragment) bit is set in the original IP packet, and an ICMP message is sent.

NOTE
For ICX 7850 devices, the maximum jumbo frame size supported is 9380. When jumbo mode is enabled, the maximum ethernet MTU size is 9358 bytes.

Path MTU Discovery support

The following RFCs for handling large packets over a GRE tunnel are supported:

- RFC 1191, Path MTU Discovery
- RFC 4459, MTU and Fragmentation Issues with In-the-Network Tunneling

RFC 1191 describes a method for dynamically discovering the maximum transmission unit (MTU) of an arbitrary internet path. When a FastIron device receives an IP packet that has its Do not Fragment (DF) bit set, and the packet size is greater than the MTU value of the outbound interface, then the FastIron device returns an ICMP Destination Unreachable message to the source of the packet, with the code indicating "fragmentation needed and DF set". The ICMP Destination Unreachable message includes the MTU of the outbound interface. The source host can use this information to help determine the minimum MTU of a path to a destination.

RFC 4459 describes solutions for issues with large packets over a tunnel. The following methods, from RFC 4459, are supported:

- If a source attempts to send packets that are larger than the lowest MTU value along the path, Path MTU Discovery (PMTUD) can signal to the source to send smaller packets. This method is described in Section 3.2 of RFC 4459.
- Inner packets can be fragmented before encapsulation, in such a manner that the encapsulated packet fits in the tunnel path MTU, which is discovered using PMTUD. This method is described in Section 3.4 of RFC 4459.

By default, PMTUD is enabled.

NOTE
ICX 7150 devices do not support tunnels.

Support for IPv4 multicast routing over GRE tunnels

PIM-DM and PIM-SM Layer 3 multicast protocols and multicast data traffic are supported over GRE tunnels. When a multicast protocol is enabled on both ends of a GRE tunnel, multicast packets can be sent from one tunnel endpoint to another. To accomplish this, the packets are encapsulated using the GRE unicast tunneling mechanism and forwarded like any other IPv4 unicast packet to the destination endpoint of the tunnel. The router that terminates the tunnel (i.e., the router where the tunnel endpoint is an ingress interface) de-encapsulates the GRE tunneled packet to retrieve the native multicast data packets. After de-encapsulation, data packets are forwarded in the direction of its receivers, and control packets may be consumed. This creates a PIM-enabled virtual or logical link between the two GRE tunnel endpoints.

Strict RPF check for multicast protocols

Ruckus software enforces strict Reverse Path Forwarding (RPF) check rules on an (s,g) entry on a GRE tunnel interface. The (s,g) entry uses the GRE tunnel as an RPF interface. During unicast routing transit, GRE tunnel packets may arrive at different physical interfaces. The strict RPF check limits GRE PIM tunnel interfaces to accept the (s,g) GRE tunnel traffic.

Configuring IP GRE tunnels

Generic Routing Encapsulation can be configured on tunnel interfaces.

When configuring a GRE tunnel you must create a tunnel interface, configure a source address or source interface for the tunnel interface, configure the destination address, enable GRE encapsulation for the tunnel interface, and configure an IP address for the tunnel. If a route to the tunnel destination does not exist, you must create a static route and specify that the route is through the tunnel interface. These configuration requirements are explained in more detail in the following steps.

NOTE
ICX 7150 devices do not support Generic Routing Encapsulation (GRE) tunnels. This task is not supported for ICX 7150 devices.

1. Enter global configuration mode.

```
device# configure terminal
```

2. Enter interface configuration mode and specify an Ethernet interface.

```
device(config)# interface ethernet 1/1/1
```

3. Assign an IP address to Ethernet port 1/1/1.

```
device(config-if-e10000-1/1/1)# ip address 10.45.6.1 255.255.255.0
```

4. Return to global configuration mode to configure the GRE tunnel interface.

```
device(config-if-e10000-1/1/1)# exit
```

5. Create a tunnel interface.

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

6. Configure the source address for the tunnel interface.

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

   The tunnel source address should be one of the device IP addresses configured on a physical, loopback, or virtual Ethernet (ve) interface, through which the other end of the tunnel is reachable.

   If the destination address for a tunnel interface is not resolved, Ruckus recommends that you either configure the source interface (instead of the source address) as the source for a tunnel interface, or enable GRE link keepalive on the tunnel interface.

7. Create the destination address for the tunnel interface.

```
device(config-tnif-1)# tunnel destination 131.108.5.2
```

   The destination address should be the address of the IP interface of the device at the other end of the tunnel.

   If the destination address for a tunnel interface is not resolved, Ruckus recommends that you either configure the source interface (instead of the source address in Step 6) as the source for a tunnel interface, or enable GRE link keepalive on the tunnel interface.

8. Enable GRE encapsulation on the tunnel interface.

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

9. Configure a tunnel loopback port for the tunnel interface.

```
device(config-tnif-1)# tunnel loopback 4/1
```

10. Configure an IP address for the tunnel interface.

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

    An IP address sets a tunnel interface as an IP port and allows the configuration of Layer 3 protocols, such as OSPF, BGP, and Multicast (PIM-DM and PIM-SM) on the port. Note that the subnet cannot overlap other subnets configured on other routing interfaces, and both ends of the tunnel should be in the same subnet.

11. Return to global configuration mode.

```
device(config-tnif-1)# exit
```

12. If a route to the tunnel destination does not exist, configure a static route to the tunnel destination.

```
device(config)# ip route 131.108.5.0/24 10.0.8.1
```

13. If you configured a static route, set the route to go through the tunnel interface.

```
device(config)# ip route 10.10.2.0/24 tunnel 1
```

The following example configures an IP GRE tunnel.

```
device# configure terminal
device(config)# interface ethernet 1/1/1
device(config-if-e1000-1/1/1)# ip address 10.0.8.108/24
device(config)# exit
device(config)# interface tunnel 1
device(config-tnif-1)# tunnel source 10.0.8.108
device(config-tnif-1)# tunnel destination 131.108.5.2
device(config-tnif-1)# tunnel mode gre ip
device(config-tnif-1)# tunnel loopback 4/1
device(config-tnif-1)# ip address 10.10.3.1/24
device(config-tnif-1)# exit
device(config)# ip route 131.108.5.0/24 10.0.8.1
device(config)# ip route 10.10.2.0/24 tunnel 1
```

Configuring optional IP GRE tunnel parameters

There are a few options that can be configured for IP GRE tunnels.

Before configuring this task, review the "Configuring IP GRE tunnels" task to view the required steps for creating GRE tunnels. In this task, it is assumed that you have already configured the GRE tunnel.

Optional parameters for GRE tunnels include changing the maximum transmission unit (MTU) value, enabling GRE keepalives, and changing the maximum number of supported GRE tunnels. These configuration options are explained in more detail in the following steps.

NOTE
ICX 7150 devices do not support Generic Routing Encapsulation (GRE) tunnels. This task is not supported for ICX 7150 devices.

1. Enter global configuration mode.

```
device# configure terminal
```

2. Enter interface configuration mode and specify an Ethernet interface.

```
device(config)# interface ethernet 1/1/1
```

3. Assign an IP address to Ethernet port 1/1/1.

```
device(config-if-e10000-1/1/1)# ip address 10.45.6.1 255.255.255.0
```

4. Return to global configuration mode to configure the GRE tunnel interface.

```
device(config-if-e10000-1/1/1)# exit
```

5. Create a tunnel interface.

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

6. (Optional) Change the MTU value for the tunnel interface.

```
device(config-tnif-1)# ip mtu 1200
```

When changing the GRE tunnel MTU, set the MTU to at least 24 bytes less than the IP MTU of the outgoing interface. If the MTU is not set to at least 24 bytes less than the IP MTU, the size of the encapsulated packet will exceed the IP MTU of the outgoing interface. This will cause the packet to either be sent to the CPU for fragmentation, or the packet will be dropped if the DF (Do-Not-Fragment) bit is set in the original IP packet, and an ICMP message is sent.

7. (optional) Configure a GRE link keepalive.

```
device(config-tnif-1)# keepalive 12 4
```

To enable GRE link keepalive, configure it on one end of the tunnel and ensure the other end of the tunnel has GRE enabled.

In this example, the device to waits for 4 consecutive lost keepalive packets before bringing the tunnel down. There will be a 12 second interval between each packet. Note that when the tunnel comes up, it would immediately (within one second) send the first keepalive packet.

8. (optional) Return to global configuration mode. The remaining steps in this task are required only if you want to change the maximum number of supported GRE tunnels for your device.

```
device(config-tnif-1)# exit
```

9. Change the maximum number of supported GRE tunnels.

```
device(config)# system-max gre-tunnels 16
Reload required. Please write memory and then reload or power cycle.
```

   Check the table in the GRE tunnel parameters section to determine how many GRE tunnels your device can support.

10. Save the current configuration to memory before reloading.

```
device(config)# write memory
```

11. Return to privileged EXEC mode.

```
device)# exit
```

12. Reload the device to ensure that the number of GRE tunnels supported is changed.

```
device)# reload
```

    Allow time for the device to reload the software.

The following example configures optional parameters for an IP GRE tunnel.

```
device# configure terminal
device(config)# interface ethernet 1/3/1
device(config-if-e1000-1/3/1)# ip address 10.0.8.108/24
device(config)# exit
device(config)# interface tunnel 1
device(config-tnif-1)# ip mtu 1200108
device(config-tnif-1)# keepalive 12 4
device(config-tnif-1)# exit
device(config)# system-max gre-tunnels 16
device(config)# write memory
Reload required. Please write memory and then reload or power cycle.
device(config)# exit
device(config)# reload
```

Configuring Path MTU Discovery on a GRE tunnel

Path MTU discovery (PMTUD) is enabled by default and has several configurable options.

PMTUD is enabled by default on tunnel interfaces. In this task, PMTUD is assumed to be disabled and a step shows how to reenable PMTUD. Options such as changing the PMTUD age timer, manually clear the tunnel PMTUD, and view the PMTUD configuration are also explained using the following steps.

Before configuring this task, review the “Configuring IP GRE tunnels” task to view the required steps for creating GRE tunnels. In this task, it is assumed that you have already configured the GRE tunnel. Full tunnel configuration is shown in the example at the end of this task.

1. Enter global configuration mode.

```
device# configure terminal
```

2. Enter interface configuration mode and specify an Ethernet interface.

```
device(config)# interface ethernet 1/3/1
```

3. Assign an IP address to Ethernet port 1/3/1.

```
device(config-if-e10000-1/3/1)# ip address 10.0.8.108 255.255.255.0
```

4. Return to global configuration mode to configure the GRE tunnel interface.

```
device(config-if-e10000-1/3/1)# exit
```

5. Create a tunnel interface.

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

6. (Optional) Reenable PMTUD for the tunnel interface.

```
device(config-tnif-1)# no tunnel path-mtu-discovery disable
```

   To disable PMTUD, use the **tunnel path-mtu-discovery disable** command.

7. (Optional) Change the age timer for PMTUD.

```
device(config-tnif-1)# tunnel path-mtu-discovery age-timer 20
```

   In this example, the device waits for 20 minutes before resetting the path MTU to its original value. By default, when PMTUD is enabled on a tunnel interface, the path MTU is reset to its original value every 10 minutes. If desired, you can change the reset time (default age timer) to a value of up to 30 minutes.
   comm

8. (Optional) Reset a dynamically-configured PMTUD on tunnel interface 1.

```
device(config-tnif-1)# clear ip tunnel pmtud 1
```

9. Display PMTUD infomration using the show interfaces tunnel command

```
show interfaces tunnel 1

Tunnel1 is up, line protocol is up
Hardware is Tunnel
Tunnel source 10.0.8.108
Tunnel destination is 131.108.5.2
Tunnel mode gre ip
Internet address is 10.10.3.1/24, MTU 1476 bytes, encapsulation GRE
Keepalive is not Enabled
Path MTU Discovery: Enabled, MTU is 1428 bytes, age-timer: 20 minutes
Path MTU will expire in 0 minutes 50 secs
```

   In this example, GRE tunnel configuration and PMTUD aging timer information is displayed.

The following example configures a GRE tunnel and changes the PMTUD aging timer to 20.

```
device# configure terminal
device(config)# interface ethernet 1/3/1
device(config-if-e1000-1/3/1)# ip address 10.0.8.108/24
device(config)# exit
device(config)# interface tunnel 1
device(config-tnif-1)# tunnel source 10.0.8.108
device(config-tnif-1)# tunnel destination 131.108.5.2
device(config-tnif-1)# tunnel mode gre ip
device(config-tnif-1)# tunnel path-mtu-discovery age-timer 20
device(config-tnif-1)# tunnel loopback 4/1
device(config-tnif-1)# ip address 10.10.3.1/24
device(config-tnif-1)# exit
device(config)# ip route 131.108.5.0/24 10.0.8.1
device(config)# ip route 10.10.2.0/24 tunnel 1
```

Enabling IPv4 multicast routing over a GRE tunnel

This section describes how to enable IPv4 multicast protocols, PIM Sparse (PIM-SM) and PIM Dense (PIM-DM), on a GRE tunnel. Perform the procedures in this section after completing the “Configuring IP GRE tunnels” task.

For an overview of multicast routing support over a GRE tunnel, refer to the “Support for IPv4 multicast routing over GRE tunnels” section.

NOTE
ICX 7150 devices do not support tunnels. This task is not supported for ICX 7150 devices.

Enabling PIM-SM on a GRE tunnel

To enable PIM-SM on a GRE tunnel interface, use the **ip pim-sparse** command in tunnel interface mode.

```
device(config)# interface tunnel 10
device(config-tnif-10)# ip pim-sparse
```

Enabling PIM-DM on a GRE tunnel interface

To enable PIM-DM on a GRE tunnel interface, use the **ip pim** command in tunnel interface mode.

```
device(config)# interface tunnel 10
device(config-tnif-10)# ip pim
```

Assigning a VRF routing instance to a GRE tunnel interface

A GRE tunnel interface can be assigned to an existing user defined VRF. When the VRF is configured on a tunnel, all IPv4 and IPv6 addresses are removed. The tunnel loopback configuration is removed.

NOTE
ICX 7150 devices do not support tunnels. This task is not supported for ICX 7150 devices.

To assign the VRF named VRF1 to tunnel 1, enter the following commands.

```
device(config)# interface tunnel 1
device(config-tnif-l)# vrf forwarding VRF1
```

Deleting an IP address from an interface configured as a tunnel source

To delete an IP address from an interface that is configured as a tunnel source, first remove the tunnel source from the tunnel interface then delete the IP address, as shown in the following example.

NOTE
ICX 7150 devices do not support tunnels. This task is not supported for ICX 7150 devices.

```
device(config-if-e1000-1/1/3)# interface tunnel 8
device(config-tnif-8)# no tunnel source 10.1.83.15
device(config-tnif-8)# interface ethernet 1/1/3
device(config-if-e1000-1/1/3)# no ip address 10.1.83.15/24
```

If you attempt to delete an IP address without first removing the tunnel source, the console will display an error message, as shown in the following example.

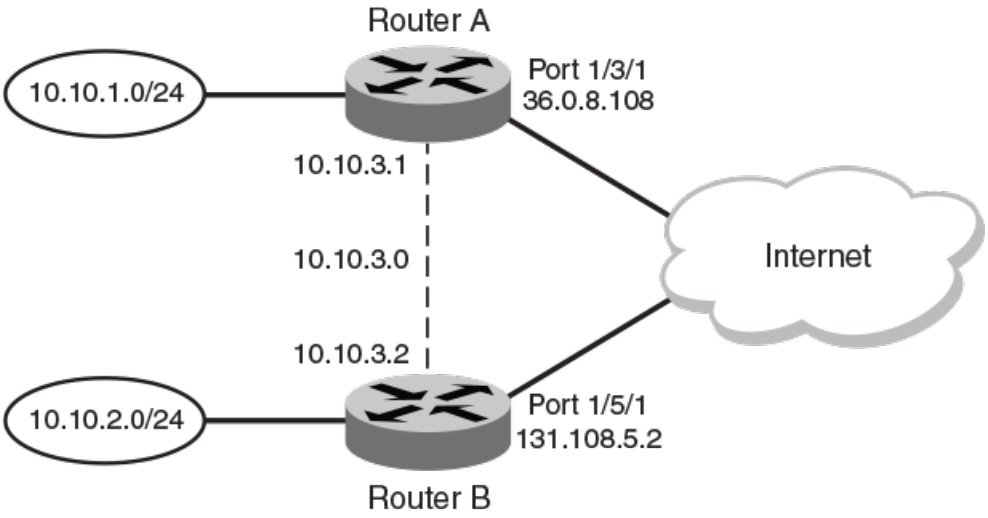
```
device# config terminal
device(config)# interface ethernet 1/1/3
device(config-if-e1000-1/1/3)# no ip address 10.1.83.15/24
Error - Please remove tunnel source from tnnl 8 before removing IP address
```

NOTE
The error message, shown in this example, will also display on the CLI when an interface is part of a VLAN. A VLAN cannot be deleted until the tunnel source is first removed.

Example point-to-point GRE tunnel configuration

A GRE Tunnel is configured between Router A and Router B. Traffic between networks 10.10.1.0/24 and 10.10.2.0/24 is encapsulated in a GRE packet 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 Layer 3 switch to go through the tunnel interface to the target network.

FIGURE 7 Point-to-point GRE tunnel configuration example



The following shows the configuration commands for this example.

NOTE
ICX 7150 devices do not support tunnels.

Configuring point-to-point GRE tunnel for Router A

```
device# configure terminal
device(config)# interface ethernet 1/3/1
device(config-if-e1000-1/3/1)# ip address 10.0.8.108/24
device(config)# exit
device(config)# interface tunnel 1
device(config-tnif-1)# tunnel source 10.0.8.108
device(config-tnif-1)# tunnel destination 131.108.5.2
device(config-tnif-1)# tunnel mode gre ip
device(config-tnif-1)# tunnel loopback 4/1
device(config-tnif-1)# ip address 10.10.3.1/24
device(config-tnif-1)# exit
device(config)# ip route 131.108.5.0/24 10.0.8.1
device(config)# ip route 10.10.2.0/24 tunnel 1
```

Configuring point-to-point GRE tunnel for Router B

```
device(config)# interface ethernet 1/5/1
device(config-if-e1000-1/5/1)# ip address 131.108.5.2/24
```

```
device(config)# exit
device(config)# interface tunnel 1
device(config-tnif-1)# tunnel source 131.108.5.2
device(config-tnif-1)# tunnel destination 10.0.8.108
device(config-tnif-1)# tunnel mode gre ip
device(config-tnif-1)# tunnel loopback 1/1
device(config-tnif-1)# ip address 10.10.3.2/24
device(config-tnif-1)# exit
device(config)# ip route 10.0.8.0/24 131.108.5.1
device(config)# ip route 10.10.1.0/24 tunnel
```

Displaying GRE tunneling information

This section describes the **show** commands that display the GRE tunnels configuration, the link status of the GRE tunnels, and the routes that use GRE tunnels. For field definitions of the commands shown below, refer to the *FastIron Command Reference*.

NOTE
ICX 7150 devices do not support tunnels.

To display GRE tunneling Information, use the following commands:

- **show ip interface**
- **show ip route**
- **show ip interface tunnel**
- **show ip tunnel traffic**
- **show interface tunnel**
- **show statistics tunnel**

To view basic information about GRE tunnels, use the **show ip interface** command.

```
device# show ip interface

Interface  IP-Address  OK?  Method  Status  Protocol VRF
Tunnel 1   101.1.1.1   YES  NVRAM   up         up       red
Tunnel 3   89.1.1.1    YES  NVRAM   up         up       default-vrf
```

To display routes that are pointing to a GRE tunnel, use the **show ip route** command.

```
device# show ip route

Total number of IP routes: 3, avail: 79996 (out of max 80000)
B:BGP D:Connected  R:RIP  S:Static  O:OSPF *:Candidate default
     Destination     NetMask         Gateway        Port       Cost    Type
1    10.1.1.0        255.255.255.0   0.0.0.0         7          1       D
2    10.1.2.0        255.255.255.0   10.1.1.3        7          1       S
3    10.34.3.0       255.255.255.0   0.0.0.0         tn3        1       D
```

To display the link status and IP address configuration for an IP tunnel interface, use the **show ip interface tunnel** command.

```
device# show ip interface tunnel 64

Interface Tunnel 64
  port enabled
  port state: UP
  ip address: 223.224.64.0/31
  Port belongs to VRF: default-vrf
  encapsulation: GRE, mtu: 1476, metric: 1
  directed-broadcast-forwarding: disabled
  proxy-arp: disabled
  ip arp-age:  10 minutes
  No Helper Addresses are configured.
  No inbound ip access-list is set
  No outgoing ip access-list is set
```

To display the GRE tunnel configuration and the Path MTU aging timer information, use the **show interface tunnel** command.

```
device# show interface tunnel 10

Tunnel10 is up, line protocol is up
  Hardware is Tunnel
  Tunnel source  1.1.41.10
  Tunnel destination is 1.1.14.10
  Tunnel mode gre ip
  Port name is GRE_10_to_VR1_on_ICX_STACK
  Internet address is 223.223.1.1/31, MTU 1476 bytes, encapsulation GRE
  Keepalive is not Enabled
  Path MTU Discovery: Enabled, MTU is 1428 bytes, age-timer: 10 minutes
  Path MTU will expire in 0 minutes 50 secs
```

To display the link status of the tunnel and the number of keepalive packets received and sent on the tunnel, use the **show ip tunnel traffic** command .

```
device# show ip tunnel traffic

IP GRE Tunnels
      Tunnel Status  Packet Received  Packet Sent  KA recv  KA sent
  1   up/up          362              0            362      362
  3   up/up          0                0            0        0
  10  down/down      0                0            0        0
```

To display GRE tunnel statistics for a specifc tunnel ID number, use the **show statistics tunnel** command with a number. for the specific tunnel ID.

```
device(config-tnif-10)# show statistics tunnel 1

IP GRE Tunnels
      Tunnel Status  Packet Received  Packet Sent  KA recv  KA sent
  1   up/up          87120            43943        43208    43855
```

RFC 2784 supports GRE tunnel ports. The show statistics tunnel command output now includes information from the hardware counters for each tunnel. For example:

```
IP GRE Tunnel 1 HW Counters:
      InOctets                    0          OutOctets               0
      InPkts                      0          OutPkts                 0
```

Clearing GRE statistics

Use the **clear ip tunnel** command to clear statistics related to GRE tunnels.

To clear GRE tunnel statistics, enter a command such as the following.

```
device(config)# clear ip tunnel stat 3
```

To reset a dynamically-configured MTU on a tunnel Interface back to the configured value, enter a command such as the following.

```
device(config)#clear ip tunnel pmtud 3
```

Use the **clear statistics tunnel** command to clear GRE tunnel statistics for a specific tunnel ID number. To clear GRE tunnel statistics for tunnel ID 3, enter a command such as the following.

```
device(config)# clear statistics tunnel 3
```

NOTE
ICX 7150 devices do not support tunnels.

Bandwidth for IP interfaces

The bandwidth for an IP interface can be specified so that higher level protocols, such as OSPFv2 and OSPFv3, can use this setting to influence the routing cost for routes learned on these interfaces.

When the interface bandwidth is configured, the number of network and router link state advertisement generation is reduced during an operation down or a shutdown of one or more of the associated interfaces of the VE interface. For OSPF, when the dynamic cost feature is enabled, the bandwidth for a VE interface is the sum of bandwidth for either all associated ports or all active associated ports. However, when the interface bandwidth is configured on the VE interface itself, the bandwidth of the associated ports are not used in the OSPF cost calculation. This means that even when one of the associated ports of the VE interface goes down, there is no OSPF cost recalculation.

The bandwidth for IP interfaces feature can be configured for a physical interface, Link aggregation (LAG) groups, a VE interface, and a tunnel interface.

The bandwidth for IP interfaces feature can be used to:

- Query the bandwidth for an interface.
- Help OSPF avoid generating numerous LSAs while updating the cost value for a VE interface due to changes in associated physical interfaces.
- Influence the cost on OSPF interfaces for specific tunnels, VE interfaces, and physical interfaces.

The bandwidth for IP interfaces feature enables OSPF to calculate its interface metric cost more precisely, based on the specified interface bandwidth. If the interface bandwidth feature is disabled, OSPF calculates the cost as the reference-bandwidth divided by the fixed port bandwidth, as outlined in the Changing the reference bandwidth for the cost on OSPFv2 interfaces on page 219 section. When the interface bandwidth feature is enabled, OSPF calculates the cost as the reference-bandwidth divided by the interface bandwidth. For a physical interface, the interface bandwidth is assigned by default to the port speed.

The interface bandwidth feature also enables OSPF to use the configured interface bandwidth for a VE interface to calculate its routing metric, without considering the bandwidth of the associated physical ports. When this feature is enabled, the bandwidth for a VE interface is the interface bandwidth value if it is configured under the VE. Alternatively, it is the sum of the interface bandwidth for all associated ports or all active ports when OSPF dynamic cost is enabled.

The bandwidth of a trunk port for OSPF is, by default, the sum of either all the associated ports or all active associated ports when OSPF dynamic cost is enabled. The interface bandwidth defined on the LAG virtual interface is used if the interface bandwidth is configured; otherwise it reverts to the default behavior.

NOTE
If the interface bandwidth configuration on the LAG virtual interface is different to any of the member ports, then the LAG does not become operational. When the LAG is deleted, the interface bandwidth value for all member ports is reset to the port speed.

The configured value is exposed in SNMP via ifSpeed (in ifTable) and ifHighSpeed (in ifXTable) objects.

NOTE
GRE or IPv6 tunnel bandwidth may limit routing protocol traffic propagating through the tunnel. For example, if the tunnel defaults to 8kbps , OSPF uses 50% of the tunnel bandwidth for Hello and update traffic. Therefore, it is good practice to increase the tunnel bandwidth when a routing protocol runs over it to eliminate flapping, and give the routing protocol more capacity to send its update and Hello messages.

From FastIron Release 08.0.30, this feature is supported on all platforms.

Limitations and pre-requisites

- The bandwidth for IP interfaces feature does not support setting and adjusting GRE or IPv6 receiving and transmission bandwidth.
- SNMP does not support any IP interface bandwidth related configurations.
- ICX 7150 devices do not support tunnels.

OSPF cost calculation with interface bandwidth

OSPF uses a formula to calculate a path cost when interface bandwidth is available.

If the interface bandwidth feature is disabled, OSPF calculates the cost as the reference-bandwidth divided by the fixed port bandwidth, as outlined in the Changing the reference bandwidth for the cost on OSPFv2 interfaces on page 219 section. When the interface bandwidth feature is enabled, OSPF calculates the cost as the reference-bandwidth divided by the interface bandwidth.

OSPF uses the following formula to calculate the path cost when interface bandwidth is available:

- OSPF path cost = ((auto-cost × reference-bandwidth + interface bandwidth) -1) / interface bandwidth.

In the above formula, the cost is calculated in megabits per second (Mbps). The auto-cost is configured using the **auto-cost reference-bandwidth** command in OSPF router configuration mode or OSPFv3 router configuration mode. For more information on changing the OSPF auto-cost reference-bandwidth, refer to the Changing the reference bandwidth for the cost on OSPFv3 interfaces on page 249 section.

Setting the bandwidth value for an Ethernet interface

The current bandwidth value for an Ethernet interface can be set and communicated to higher-level protocols such as OSPF.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **interface ethernet** command to configure an Ethernet interface and enter interface configuration mode.

```
device(config)# interface ethernet 1/1/1
```

3. Enter the **bandwidth** command and specify a value to set the bandwidth value on the interface.

```
device(config-if-e1000-1/1/1)# bandwidth 2000
```

This example sets the bandwidth to 2000 kbps on a specific Ethernet interface.

```
device# configure terminal
device(config)# interface ethernet 1/1/1
device(config-if-e1000-1/1/1)# bandwidth 2000
```

The bandwidth specified in this example results in the following OSPF cost, assuming the auto-cost is 100:

- OSPF cost is equal to ((100 * 1000) + (2000 - 1)/ 2000) = 50

Setting the bandwidth value for a VE interface

The current bandwidth value for a VE interface can be set and communicated to higher-level protocols such as OSPF.

1. Enter the **configure terminal** command to access global configuration mode.

   ```
   device# configure terminal
   ```

2. Enter the **vlan** command and specify a value to configure a VLAN.

   ```
   device(config)# vlan 10
   ```

3. Enter the **tagged ethernet** command and specify an interface to add a port that is connected to the device and host in the same port-based VLAN.

   ```
   device(config-vlan-10)# tagged ethernet 1/1/1
   ```

4. Enter the **router-interface ve** command and specify a value to create a virtual interface as the routing interface for the VLAN.

   ```
   device(config-vlan-10)# router-interface ve 10
   ```

   Creates VE 10 as the routing interface for the VLAN.

5. Enter the **interface ve** command and specify a value.

   ```
   device(config-vlan-10)# interface ve 10
   ```

   Creates a VE interface with the VLAN ID of 10.

6. Enter the **bandwidth** command and specify a value to set the bandwidth value on the interface.

   ```
   device(config-vif-10)# bandwidth 2000
   ```

This example sets the bandwidth to 2000 kbps on a specific VE interface .

```
device# configure terminal
device(config)# vlan 10
device(config-vlan-10)# tagged ethernet 1/1/1
device(config-vlan-10)# router-interface ve 10
device(config-vlan-10)# interface ve 10
device(config-vif-10)# bandwidth 2000
```

The bandwidth specified in this example results in the following OSPF cost, assuming the auto-cost is 100:

- OSPF cost is equal to ((100 * 1000) + (2000 - 1)/ 2000) = 50

Setting the bandwidth value for a tunnel interface

The current bandwidth value for a tunnel interface can be set and communicated to higher-level protocols such as OSPF.

NOTE
ICX 7150 devices do not support tunnels.

1. Enter the **configure terminal** command to access global configuration mode.

   ```
   device# configure terminal
   ```

2. Enter the **interface tunnel** command and specify a value to configure a tunnel interface.

   ```
   device(config)# interface tunnel 2
   ```

3. Enter the **tunnel mode gre ip** command to enable GRE IP encapsulation on the tunnel interface.

```
device(config-tnif-2)# tunnel mode gre ip
```

4. Enter the **tunnel source** command and specify an IP address to configure the source address for the tunnel interface.

```
device(config-tnif-2)# tunnel source 10.0.0.1
```

5. Enter the **tunnel destination** command and specify an IP address to configure the destination address for the tunnel interface.

```
device(config-tnif-2)# tunnel destination 10.10.0.1
```

6. Enter the **ip address** command and specify an IP address and a network mask to assign an IP address to the tunnel interface.

```
device(config-tnif-2)# ip address 10.0.0.1/24
```

7. Enter the **bandwidth** command and specify a value to set the bandwidth value on the interface.

```
device(config-tnif-2)# bandwidth 2000
```

This example sets the bandwidth to 2000 kbps on a specific tunnel interface .

```
device# configure terminal
device(config)# interface tunnel 2
device(config-tnif-2)# tunnel mode gre ip
device(config-tnif-2)# tunnel source 10.0.0.1
device(config-tnif-2)# tunnel destination 10.10.0.1
device(config-tnif-2)# ip address 10.0.0.1/24
device(config-tnif-2)# bandwidth 2000
```

The bandwidth specified in this example results in the following OSPF interface costs, assuming the auto-cost is 100:

- OSPF Interface Cost for the Trunk Group is equal to ((100 * 1000) + (2000 - 1)÷ 2000) = 50
- OSPF Interface Cost for the GRE/IPv6 tunnel is equal to ((100 * 1000) + (2000 - 1)÷ 2000) = 50

User-configurable MAC address per IP interface

Manual configuration of one IP MAC address for each Layer 3 physical or virtual Ethernet (VE) interface on a device is permitted. The configured MAC address is used as the source MAC address by routing protocols or hardware communication related to the IPv4 or IPv6 addresses on the interface, for example, in ARP or Neighbor Discovery (ND) packets to the interface. The IPv4 and IPv6 addresses use the same IP MAC address for any software and hardware communication.

If an IP MAC address is not configured, the IP interface uses the MAC address from the router or stack.

User-configurable MAC address per IP interface supports the following unicast and multicast protocols:

- IPv4 support: ARP, BGP, OSPF, RIP, PIM-SM, PIM-DM, IGMP, MSDP
- IPv6 support: BGP4+, Neighbor Discovery (ND), OSPFv3, RD, RIPng, PIM-SM, PIM-DM, MLD

In addition to the unicast protocol support, the configured MAC address is used by IPv4 and IPv6 unicast software-generated packets (for example, ping) and IPv4 and IPv6 hardware-forwarded packets. For IPv4 addresses that are configured on the IP interface, gratuitous ARP is generated when the IP MAC address is configured. For IPv6 addresses, Duplicate Address Detection (DAD) is started and link-local addresses are regenerated when the IP MAC address is configured.

If Virtual Router Redundancy Protocol (VRRP) IPv4 or IPv6 sessions are configured on an interface where an IP MAC address is configured, the VRRP sessions continue to use the virtual MAC address assigned to the virtual router ID (VRID) for any ARP or ND queries.

Some restrictions apply to user-configurable MAC address per IP interface:

- The IP MAC address must be unique on the device including any interfaces. If the device is configured as part of a stack, the IP MAC address must not be the same as the MAC address of other stack units. If a stack MAC address is configured, it must not be the same as the IP MAC address on any interface.
- The IP MAC address configured manually for a VE interface must be unique within the same VLAN.
- There is a maximum number of IP interfaces (248) on which an IP MAC address can be configured and the number of VRRP virtual interfaces that can be supported simultaneously is affected by any increase over the default number of 120 interfaces. If the **system-max max-ip-mac** command is set above 120, a reduction in the number of IPv4 VRRP entries supported is calculated as *configured-value* - 120. For example, if the **system-max max-ip-mac** value is set to 130, the number of IPv4 VRRP entries is reduced by 10 entries (130-120).

Manually configuring an IP MAC address

One IP MAC address can be manually configured for each Layer 3 physical or virtual ethernet (VE) interface on a device. The configured MAC address will be used for all the software and hardware communications related to unicast IPv4 or IPv6 addresses on the IP interface.

1. From privileged EXEC mode, enter configuration mode by issuing the **configure terminal** command.

```
device# configure terminal
```

2. Configure an ethernet interface link.

```
device(config)# interface ethernet 1/1/6
```

3. Configure the IP address of the interface.

```
device(config-if-e1000-1/1/6)# ip address 10.53.5.1/24
```

4. Configure a MAC address on the interface.

```
device(config-if-e1000-1/1/6)# ip-mac aaaa.bbbb.cccc
```

5. Exit to Privileged EXEC mode.

```
device(config-if-e1000-1/1/6)# end
```

6. Use the **show ip interface** command to verify the user-configured MAC address.

```
device# show ip interface ethernet 1/1/6

Interface Ethernet 1/1/6
  port enabled
  port state: DOWN
  ip address: 10.53.5.1         subnet mask: 255.255.255.0
  Port belongs to VRF: default-vrf
  encapsulation: ETHERNET, mtu: 1500, metric: 1
  directed-broadcast-forwarding: disabled
  ICMP redirect: disabled
  proxy-arp: disabled
  ip arp-age:  10 minutes
  No Helper Addresses are configured.
  No inbound ip access-list is set
  No outgoing ip access-list is set
  ip-mac: aaaa.bbbb.cccc
```

The user-configured MAC address is shown after the “ip-mac:” text.

Modifying and displaying Layer 3 system parameter limits

Before manually changing IP system parameter values, you must view the existing values.

Changing the system parameters reconfigures the device memory. Whenever you reconfigure the memory on a Ruckus device, you must save the change to the startup-config file using the **write memory** command, and then reload the software to place the change into effect.

NOTE
The Layer 3 system parameter limits for FastIron IPv6 models are automatically adjusted by the system and cannot be manually modified.

To display the Layer 3 system parameter defaults, maximum values, and current values, enter the **show default value** command at any level of the CLI. The following example shows output on an ICX 7450 with third generation modules.

```
device# show default value

sys log buffers:50         mac age time:300 sec       telnet sessions:5

ip arp age:10 min          bootp relay max hops:4     ip ttl:64 hops
ip addr per intf:24

when multicast enabled :
igmp group memb.:260 sec   igmp query:125 sec         hardware drop: enabled

when ospf enabled :
ospf dead:40 sec           ospf hello:10 sec          ospf retrans:5 sec
ospf transit delay:1 sec

when bgp enabled :
bgp local pref.:100        bgp keep alive:60 sec      bgp hold:180 sec
bgp metric:10              bgp local as:1             bgp cluster id:0
bgp ext. distance:20       bgp int. distance:200      bgp local distance:200

System Parameters    Default    Maximum    Current    Configured
ip-arp               4000       64000      4000       4000
ip-static-arp        512        6000       512        512
ip-cache             10000      32768      10000      10000
ip-filter-port       3071       3071       3071       3071
ip-filter-sys        3072       8192       3072       3072
```

```
l3-vlan              32         1024       32         32
ip-qos-session       1024       16000      1024       1024
mac                  32768      32768      32768      32768
ip-route             12000      15168      12000      12000
ip-static-route      64         2048       64         64
some lines omitted for brevity....
dot1x-mka-policy-gro 8          8          8          8
openflow-flow-entrie 3072       12288      3072       3072
openflow-pvlan-entri 40         40         40         40
openflow-unprotected 40         40         40         40
openflow-nexthop-ent 1024       3072       1024       1024
max-ip-mac           128        256        128        128
max-dhcp-snoop-entri 1024       3072       1024       1024
max-static-inspect-a 512        1024       512        512
```

Changing the Router ID

Although a Layer 3 device may have multiple IP addresses configured on different interfaces, some routing protocols need to identify the device using one IP address, the router ID.

In most configurations, a Layer 3 device has multiple IP addresses, usually configured on different interfaces. As a result, a Layer 3 device identity to other devices varies depending on the interface to which the other device is attached. Some routing protocols, including Open Shortest Path First (OSPF) and Border Gateway Protocol version 4 (BGP4), identify a Layer 3 device by just one of the IP addresses configured on the Layer 3 device, regardless of the interfaces that connect the Layer 3 devices. This IP address is the router ID.

NOTE
Routing Information Protocol (RIP) does not use the router ID.

By default, the router ID on a Ruckus Layer 3 switch is one of the following:

- If the router has loopback interfaces, the default router ID is the IP address configured on the lowest numbered loopback interface configured on the Layer 3 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 device does not have any loopback interfaces, the default router ID is the lowest numbered IP interface configured on the device.

If you prefer, you can explicitly set the router ID to any valid IP address. The IP address cannot be in use on another device in the network. Ruckus Layer 3 devices use the same router ID for both OSPF and BGP4. If the router is already configured for OSPF, you may want to use the router ID that is already in use on the router rather than set a new one. To display the router ID, enter the **show ip** command at any CLI level or select **IP** > **General links** from the Configure tree in the Web Management Interface.

NOTE
If you change the router ID, all current BGP4 sessions are cleared.

To assign a router ID, follow these steps.

1. Enter global configuration mode.

```
device# configure terminal
```

2. Assign the router ID.

```
device(config)# ip router-id 10.157.22.26
```

   You can specify an IP address used for an interface on the Ruckus Layer 3 device, but do not specify an IP address in use by another device.

3. To verify the router ID, use the **show ip** command.

```
device(config)# show ip

Global Settings
ttl: 64, arp-age: 10, bootp-relay-max-hops: 4
router-id : 10.157.22.26
enabled : BGP4 UDP-Broadcast-Forwarding Source-Route Load-Sharing RARP VSRP
arp-port-move-syslog
disabled: Route-Only Directed-Broadcast-Forwarding IRDP Proxy-ARP RIP OSPF
VRRP VRRP-Extended ICMP-Redirect add-host-route-first
```

   The router-id in the output is set to the configured IP address.

Configuring IP forwarding parameters

Various IP packet forwarding behaviors can be enabled or disabled.

The following configurable parameters control the forwarding behavior of Ruckus Layer 3 devices:

- **Time-To-Live (TTL) threshold**—The time to live (TTL) threshold prevents routing loops by specifying the maximum number of router hops an IP packet originated by the Layer 3 device can travel through. Each device capable of forwarding IP that receives the packet decrements (decreases) the packet 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.
- **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 of 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 Layer 3 device supports both types of IP source routing:
  - Strict source routing - Requires the packet to pass through only the listed routers. If the Layer 3 device receives a strict source-routed packet but cannot reach the next hop interface specified by the packet, the Layer 3 device discards the packet and sends an ICMP Source-Route-Failure message to the sender.
  - 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.
- **Ones-based and zero-based broadcasts**—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 Layer 3 device to treat IP packets with all zeros in the host portion of the destination IP address as broadcast packets.

All these parameters are optional and can be configured globally; they affect all IP interfaces configured on the Layer 3 device. The IP directed broadcast parameter can also be configured for a specific interface.

1. Enter global configuration mode.

```
device# configure terminal
```

2. (Optional) Change the TTL threshold.

```
device(config)# ip ttl 25
```

3. (Optional) Enable forwarding of IP directed broadcasts.

```
device(config)# ip directed-broadcast
```

   You can also configure the forwarding of IP directed broadcasts in interface configuration mode for a specific interface.

4. (Optional) IP source-routed packets forwarding is enabled by default. To disable the forwarding of IP source-routed packets, enter the **no ip source-route** command.

```
device(config)# no ip source-route
```

   You cannot enable or disable strict or loose source routing separately.

5. (Optional) To permit zero-based IP subnet broadcasts in addition to ones-based IP subnet broadcasts, use the **ip broadcast-zero** command.

```
device(config)# ip broadcast-zero
```

   If the **ip broadcast-zero** command is entered, you must save the configuration using the write memory command and use the reload command to reload the software. See the second example below for all the steps.

   NOTE
   The zero-based IP subnet broadcast feature applies only to IP subnet broadcasts, not to local network broadcasts. The local network broadcast address is still expected to be all ones.

The following example changes the TTL threshold, enables IP directed broadcasts, and disables IP source routing.

```
device# configure terminal
device(config)# ip ttl 25
device(config)# ip directed-broadcast
device(config)# no ip source-route
```

The following example permits zero-based IP subnet broadcasts, saves the configuration, and reloads the software.

```
device# configure terminal
device(config)# ip broadcast-zero
device(config)# write memory
device(config)# end
device# reload
```

Configuring a default network route

A Layer 3 device enables you to specify a candidate default route without the need to specify the next hop gateway. If the IP route table does not contain an explicit default route (for example, 0.0.0.0/0) or propagate an explicit default route through routing protocols, the software can use the default network route as a default route instead.

When the software uses the default network route, it also uses the default network route's next hop gateway as the gateway of last resort. This feature is useful in environments where network topology changes can make the next hop gateway unreachable. This feature allows a Layer 3 device to perform default routing even if the default network route's default gateway changes.

The default network route differs from standard default routes. When you configure a standard default route, you also specify the next hop gateway. If a topology change makes the gateway unreachable, the default route becomes unusable.

For example, if you configure 10.10.10.0/24 as a candidate default network route, if the IP route table does not contain an explicit default route (0.0.0.0/0), the software uses the default network route and automatically uses that route's next hop gateway as the default gateway. If a topology change occurs and as a result the default network route's next hop gateway changes, the software can still use the default network route. To configure a default network route, use the following CLI method.

If you configure more than one default network route, the Layer 3 switch uses the following algorithm to select one of the routes.

1. Use the route with the lowest administrative distance.
2. If the administrative distances are equal, determine if the routes are from different routing protocols (RIP, OSPF, or BGP). If so, use the route with the lowest IP address.
3. If the routes are from the same routing protocol, use the route with the best metric. The meaning of "best" metric depends on the routing protocol. Use one of the following choices.
   - RIP—The metric is the number of hops (additional routers) to the destination. The best route is the route with the fewest hops.
   - OSPF—The metric is the path cost associated with the route. The path cost does not indicate the number of hops but is instead a numeric value associated with each route. The best route is the route with the lowest path cost.
   - BGP—The metric is the Multi-exit Discriminator (MED) associated with the route. The MED applies to routes that have multiple paths through the same Autonomous System. The best route is the route with the lowest MED.
4. Enter global configuration mode.

   ```
   device# configure terminal
   ```

5. Configure a default network route.

   ```
   device(config)# ip default-network 10.157.22.0
   ```

   You can configure up to 4 default network routes.
6. To verify that the route is in the route table, enter the following command at any level of the CLI.

   ```
   device> show ip route

   Total number of IP routes: 2
   Start index: 1  B:BGP D:Connected  R:RIP  S:Static  O:OSPF *:Candidate default
       Destination      NetMask          Gateway          Port   Cost   Type
   1         10.157.20.0      255.255.255.0     0.0.0.0          lb1      1      D
   2         10.157.22.0      255.255.255.0     0.0.0.0          1/4/11   1      *D
   ```

   This example shows two routes. Both of the routes are directly attached, as indicated in the Type column. However, one of the routes is shown as type "*D", with an asterisk (*). The asterisk indicates that this route is a candidate for the default network route.

IP Load Sharing

The IP route table can contain more than one path to a given destination. When this occurs, the Layer 3 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 Layer 3 device uses IP load sharing to select a path to the destination.

IP load sharing uses a hashing algorithm based on the source IP address, destination IP address, and protocol field in the IP header, TCP, and UDP information. IP load sharing is also called "Equal-Cost Multi-Path (ECMP) load sharing or just ECMP.

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. 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
IP load sharing is based on next-hop routing, and not on source routing.

IP load sharing applies to equal-cost paths in the IP route table. Routes that are eligible for load sharing can enter the routing table from any of the following routing protocols:

- IP static routes
- Routes learned through OSPF
- Routes learned through BGP4

Administrative distance for each IP route

The administrative distance is a unique value associated with each type (source) of IP route. Each path has an administrative distance. The administrative distance is not used when performing IP load sharing, but the administrative distance is used when evaluating multiple equal-cost paths to the same destination from different sources, such as between static IP routes, OSPF, and BGP4.

The value of the administrative distance is determined by the source of the route. The Layer 3 device is configured with a unique administrative distance value for each IP route source.

When the software receives multiple paths to the same destination and the paths are from different sources, the software compares the administrative distances of the paths and selects the path with the lowest administrative distance. The software then places the path with the lowest administrative distance in the IP route table. For example, if the Layer 3 switch has a path learned from OSPF and a path learned from IBGP 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 Ruckus Layer 3 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
- Interior Gateway Protocol (IBGP) - 200
- Local BGP - 200
- Unknown - 255 (the router will not use this route)

Lower administrative distances are preferred over higher distances. For example, if the router receives routes for the same network from OSPF and from IBGP, the router 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 source. IP load sharing applies only when the IP route table contains multiple paths to the same destination, from the same IP route source.

IP load sharing does not apply to paths that come from different sources.

Path cost

The cost parameter provides a common basis of comparison for selecting from among multiple 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 Layer 3 switch 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 Layer 3 switch uses IP load sharing to select one of the lowest-cost paths.

The source of a path 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.
- **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 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 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.

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 all Ruckus Layer 3 devices, load sharing for static IP routes, OSPF routes, and BGP4 routes also is enabled by default.

In the table below, the default and the maximum number of paths for a static IP route and OSPF depend on the value for IP load sharing, and are not separately configurable.

NOTE
In the table below, the default and the maximum number of paths are not applicable for BGP4 using the Ruckus ICX 7210 and ICX 7250.

TABLE 4 Default load sharing parameters for route sources

Route source	Default maximum number of paths	Maximum number of paths	
		ICX 7450 / ICX 7250/ ICX 7150	ICX 7650 / ICX 7750/ ICX 7850
Static IP route	4	8	32

TABLE 4 Default load sharing parameters for route sources (continued)

Route source	Default maximum number of paths	Maximum number of paths	
		ICX 7450 / ICX 7250/ ICX 7150	ICX 7650 / ICX 7750/ ICX 7850
OSPF	4	8	32
BGP4	1	4	32

Changing the Number of Load Sharing (ECMP) Paths for IPv4

Although IP load sharing is enabled by default, you can configure the number of IP load sharing paths and the maximum number of load-sharing paths supported by the Layer 3 device.

You can change the maximum number of paths the Layer 3 device supports to a value from 2 through 8. On RuckusICX 7650, ICX 7750, and ICX 7850 devices, the value range for the maximum number of load-sharing paths is from 2 through 32.

TABLE 5 Maximum number of ECMP load sharing paths per device

ICX 7150/ ICX 7250 / ICX 7450	ICX 7650 / ICX 7750 / ICX 7850
8	32

For optimal results, set the maximum number of paths to a value at least as high as the maximum number of equal-cost paths your network typically contains. For example, if the Layer 3 switch you are configuring for IP load sharing has six next-hop routers, set the maximum paths value to six.

1. Enter global configuration mode.

```
device# configure terminal
```

2. Change the number of IP load sharing paths.

```
device(config)# ip load-sharing 6
```

3. Configure the maximum number of IP load sharing paths supported on the device.

```
device(config)# system-max max-ecmp 20
```

4. Save the configuration to memory.

```
device(config)# write memory
```

   Changes to any system parameters are not in effect until the configuration is saved and a reload of the software occurs.

5. Return to privileged EXEC mode

```
device(config)# exit
```

6. Reload the software.

```
device# reload
```

The following example sets the number of load-sharing paths to 6 and the maximum number of load-sharing paths to 20.

```
device# configure terminal
device(config)# ip load-sharing 6
device(config)# system-max max-ecmp 20
device(config)# write memory
device(config)# exit
device# reload
```

31-bit subnet masks on point-to-point networks

NOTE
31-bit subnet masks are supported on ICX 7150, ICX 7250, ICX 7450, ICX 7650, ICX 7750, and ICX 7850 devices running the full Layer 3 image.

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.

With IPv4, four IP addresses with a 30-bit subnet mask are allocated on point-to-point networks. In contrast, 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.

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.

When the 31-bit subnet mask address is configured on a point-to-point link, using network addresses for broadcast purposes is not allowed. For example, in an IPV4 broadcast scheme, the following subnets can be configured:

- 10.10.10.1 - Subnet for directed broadcast: {*Network-number*, -1}
- 10.10.10.0 - Subnet for network address: {*Network-number*, 0}

In a point-to-point link with a 31-bit subnet mask, the previous two addresses are interpreted as host addresses and packets are not rebroadcast.

Assigning a 31-bit subnet mask to an IPv4 address

To conserve address space, you can assign a 31-bit subnet mask to an IPv4 address.

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

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: `Error: Cannot assign /31 subnet address as secondary`.

1. Enter global configuration mode.

   ```
   device# configure terminal
   ```

2. Enter interface configuration mode.

   ```
   device(config)# interface ethernet 1/1/5
   ```

3. Configure an IP address with a 31-bit subnet mask.

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

   Using CIDR format, the IP address is 10.9.9.9/31.

4. Return to privileged EXEC mode.

   ```
   device(config-if-e1000-1/1/5)# end
   ```

5. To see when a 31-bit subnet mask is used, use the **show ip route** command.

```
device# show ip route

Total number of IP routes: 2
Type Codes - B:BGP D:Connected O:OSPF R:RIP S:Static; Cost - Dist/Metric
BGP  Codes - i:iBGP e:eBGP
OSPF Codes - i:Inter Area 1:External Type 1 2:External Type 2
        Destination        Gateway         Port          Cost          Type Uptime
1       10.9.9.8/31        DIRECT          e 1/1/5       0/0           D    1m53s
2       10.9.9.10/31       DIRECT          e 1/1/6       0/0           D    1m54s
```

The following example assigns a 31-bit subnet mask to an IP address. This example uses CIDR notation for the IP address and subnet mask.

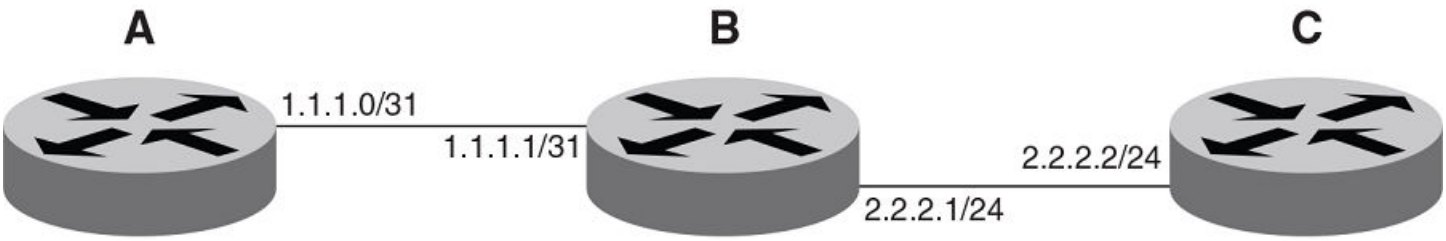
```
device# configure terminal
device(config)# interface ethernet 1/1/5
device(config)# ip address 10.9.9.9/31
```

Configuration example for 31-bit subnet mask configuration

31-bit subnet masks can be used to conserve address space. In the following diagram, Router A is connected to Router B as a point-to-point link with 10.1.1.0/31 subnet. There are only two available addresses in this subnet, 10.1.1.0 on Router A and 10.1.1.1 on Router B.

Routers B and C are connected by a regular 24-bit subnet. Router C can either be a switch with many hosts belonging to the 10.2.2.2/24 subnet connected to it, or it can be a router.

FIGURE 8 Configured 31- bit and 24-bit subnet masks



Router A

```
RouterA(config)# interface ethernet 1/1/1
RouterA(config-if-e1000-1/1/1)# ip address 10.1.1.0/31
```

Router B

```
RouterB(config)# interface ethernet 1/1/1
RouterB(config-if-e1000-1/1/1)# ip address 10.1.1.1/31
RouterB(config-if-e1000-1/1/1)# exit
RouterB(config# interface ethernet 1/3/1
RouterB(config-if-e1000-1/3/1)# ip address 10.2.2.1/24
```

Router C

```
RouterC(config# interface ethernet 1/3/1
RouterC(config-if-e1000-1/3/1)# ip address 10.2.2.2/24
```

UDP broadcast and IP helper

Some applications rely on client requests sent as limited IP broadcasts addressed to the UDP 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 request cannot reach the server.

You can configure the Layer 3 device to forward requests from clients to UDP application servers. To do so:

- Enable forwarding support for the UDP application port, if forwarding support is not already enabled.
- Configure a helper address 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 Layer 3 device forwards client requests for any of the application ports the Layer 3 device is enabled to forward to the helper address.

Forwarding support for the following application ports is enabled by default:

- dns (port 53)
- tftp (port 69)
- time (port 37)
- tacacs (port 65)
- BootP/DHCP

NOTE
The application names are the names for these applications that the Layer 3 device software 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.

You can enable forwarding for other applications by specifying the application port number. You also can disable forwarding for an application. 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 Layer 3 device is not also disabled.

Enabling UDP application forwarding

If you want the Layer 3 switch to forward client requests for UDP applications that the Layer 3 switch does not forward by default, you can enable forwarding support for the port. To enable forwarding support for a UDP application, use the following method. You also can disable forwarding for an application using this method.

NOTE
You also must configure a helper address on the interface that is connected to the clients for the application. The Layer 3 switch cannot forward the requests unless you configure the helper address.

In the following table, the UDP applications (from RFC 1340) are listed with the port number in parentheses. If you specify an application name, use the name only. You can also specify any UDP application by entering just the UDP port number.

TABLE 6 UDP applications and port numbers

UDP Application (port number)	UDP Application (port number)	UDP Application (port number)
bootpc (port 68)	echo (port 7)	tacacs (port 65)
bootps (port 67)	mobile-ip (port 434)	talk (port 517)
discard (port 9)	netbios-dgm (port 138)	time (port 37)

TABLE 6 UDP applications and port numbers (continued)

UDP Application (port number)	UDP Application (port number)	UDP Application (port number)
dns (port 53)	netbios-ns (port 137)	tftp (port 69)
dnsix (port 90)	ntp (port 123)	

To enable the forwarding of NTP broadcasts, enter the following commands.

1. Enter global configuration mode.

```
device# configure terminal
```

2. Enable the forwarding of NTP broadcasts.

```
device(config)# ip forward-protocol udp ntp
```

```
device(mode)# command executable
Command output
```

Configuring an IP helper address

To forward a client 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 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.

By default, IP helper does not forward client broadcast request to a server within the network. To forward a client broadcast request when the client and server are on the same network, configure an IP helper with the unicast option on the interface connected to the client.

To configure a helper address on unit 1, slot 1, port 2, enter the following commands.

1. Enter global configuration mode.

```
device# configure terminal
```

2. Enter interface configuration mode.

```
device(config)# interface ethernet 1/1/2
```

3. Configure an IP helper address.

```
device(config-if-e1000-1/1/2)# ip helper-address 10.95.7.6
```

   In this example, a help address is configured for server 10.95.7.6 to the port. If the port receives a client request for any of the applications that the Layer 3 switch is enabled to forward, the Layer 3 switch forwards the client request to the server.

In the following example, a unicast IP helper address is configured.

```
device(mode)# configure terminal
device(config)# interface ethernet 1/1/2
device(config-if-e1000-1/1/2)# ip helper-address 10.98.5.7 unicast
```

Enabling or disabling Layer 2 switching

By default, Ruckus Layer 3 devices support Layer 2 switching. If you want to disable Layer 2 switching, you can do so globally or on individual ports, depending on the version of software your device is running.

> **NOTE**
> Consult your reseller or Ruckus to understand the risks involved before disabling all Layer 2 switching operations.

Beginning with the Ruckus FastIron release 8.0.50, when the **route-only** command is entered in global configuration mode, the following syslogs appear to indicate the impact of the L2 functions already available on the ports.

- On tagged ports and virtual Ethernet (VE) interfaces

```
ROUTE-ONLY: Only would cause L2 functions non-functional on %p Port, Part of VE/Tagged Interface
```

- On generic attribute registration protocol (GARP) VLAN registration protocol (GVRP)-enabled ports

```
ROUTE-ONLY: Only would cause L2 functions non-functional on 1/1/15 Port, Part of VE/Tagged
Interface
```

- On virtual switch redundancy protocol (VSRP)-enabled ports

```
ROUTE-ONLY: Only would cause VSRP non-functional on 7th VLAN
```

- On metro ring protocol (MRP)-enabled ports

```
ROUTE-ONLY: Global Route-Only would cause MRP non-functional on %dth VLAN
```

Be aware of the following restrictions and limitations when disabling Layer 2 switching:

- Enabling or disabling Layer 2 switching is supported in Layer 3 software images only.
- Ruckus ICX devices support disabling Layer 3 switching at the interface configuration mode as well as the global configuration mode.
- Enabling or disabling Layer 2 switching is not supported on virtual interfaces.

Configuration examples for disabling Layer 2 switching

To globally disable Layer 2 switching on a Layer 3 device, enter commands such as the following.

```
device# configure terminal
device(config)# route-only
device(config)# exit
device# write memory
device# reload
```

To re-enable Layer 2 switching on a Layer 3 device, enter the following commands.

```
device# configure terminal
device(config)# no route-only
device(config)# exit
device# write memory
device# reload
```

To disable Layer 2 switching only on a specific interface, go to the interface configuration level for that interface, and then disable the feature. The following commands show how to disable Layer 2 switching on port 2.

```
device# configure terminal
device(config)# interface ethernet 2
device(config-if-e1000-2)# route-only
```

Configuring Delay Time for Notifying VE Down Event

When all the ports in the VLAN go into an inactive state (for example, the non-forwarding state), the device notifies the Layer 3 protocols of the VE down event only after the configured timer expires. Once the timer expires, the device checks if any of the ports is in the forwarding state. If no ports are in the forwarding state, the device notifies the Layer 3 protocols of the VE down event. If any of the ports is in the forwarding state, the device ignores the down event.

While the timer is running, if any of the ports comes into forwarding state, the device cancels the timer and does not notify the VE down event to the protocols.

NOTE
In the case of multiple flaps, if any of the ports comes into forwarding state before the delay notification timer expiry then the device cancels the timer and a fresh timer is started during port down event. Incase of continuous flaps where flap time is less than delay notification timer, the flaps can be detected by other methods like port statistics or drop in traffic or by the convergence logs of layer2 loop detection protocols.

Suppressing the link status notification allows a quick port status change and recovery to occur without triggering any of the changes that are necessary when a port stays down.

By default, the delay time is not configured.

NOTE
Configuring delayed Layer 3 notifications on the VE feature is supported on ICX 7150, ICX 7250, ICX 7450, ICX 7650, ICX 7750, and ICX 7850 product families from Ruckus.

Configuring VE down time notification

Perform the following steps to configure the delay time for notifying the Layer 3 protocols of the VE down event.

1. From global configuration mode, enter VE interface configuration mode.

```
device(config)# interface ve 50
```

2. Configure the delay notifications time value.

```
device(config-vif-50)# delay-notifications 20
```

3. Use the **show ip interface ve** command to confirm the configuration.

The following example shows how to configure the delay time for notifying the Layer 3 protocols of the VE down event.

```
device(config)# interface ve 50
device(config-vif-50)# delay-notifications 20
```

Specifying a single source interface for specified packet types

NOTE
This feature is supported on ICX 7150, ICX 7250, ICX 7450, ICX 7650, ICX 7750, and ICX 7850 switches.

When the Layer 3 switch originates a packet of one of the following types, the source address of the packet is the lowest-numbered IP address on the interface that sends the packet:

- Telnet
- TACACS/TACACS+
- TFTP
- RADIUS
- Syslog
- SNTP
- SNMP traps

You can configure the Layer 3 switch to always use the lowest-numbered IP address on a specific Ethernet, loopback, or virtual interface as the source addresses for these packets. When configured, the Layer 3 switch 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 specified packets provides the following benefits:

- If your 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 Ruckus device to always send the packets from the same link or source address.
- If you specify a loopback interface as the single source for specified 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 specific packets. You can configure a source interface for one or more of these types of packets separately.

The following sections show the syntax for specifying a single source IP address for specific packet types.

Telnet packets

To specify the lowest-numbered IP address configured on a virtual interface as the device source for all Telnet packets, enter commands such as the following.

```
device(config)# interface loopback 2
device(config-lbif-2)# ip address 10.0.0.2/24
device(config-lbif-2)# exit
device(config)# ip telnet source-interface loopback 2
```

The commands in this example configure loopback interface 2, assign IP address 10.0.0.2/24 to the interface, then designate the interface as the source for all Telnet packets from the Layer 3 switch.

The following commands configure an IP interface on an Ethernet port and designate the address port as the source for all Telnet packets from the Layer 3 switch.

```
device(config)# interface ethernet 1/1/4
device(config-if-1/1/4)# ip address 10.157.22.110/24
device(config-if-1/1/4)# exit
device(config)# ip telnet source-interface ethernet 1/1/4
```

TACACS/TACACS+ packets

To specify the lowest-numbered IP address configured on a virtual interface as the device source for all TACACS/TACACS+ packets, enter commands such as the following.

```
device(config)# interface ve 1
device(config-vif-1)# ip address 10.0.0.3/24
device(config-vif-1)# exit
device(config)# ip tacacs source-interface ve 1
```

The commands in this example configure virtual interface 1, assign IP address 10.0.0.3/24 to the interface, then designate the interface as the source for all TACACS/TACACS+ packets from the Layer 3 switch.

RADIUS packets

To specify the lowest-numbered IP address configured on a virtual interface as the device source for all RADIUS packets, enter commands such as the following.

```
device(config)# interface ve 1
device(config-vif-1)# ip address 10.0.0.3/24
device(config-vif-1)# exit
device(config)# ip radius source-interface ve 1
```

The commands in this example configure virtual interface 1, assign IP address 10.0.0.3/24 to the interface, then designate the interface as the source for all RADIUS packets from the Layer 3 switch.

TFTP packets

To specify the lowest-numbered IP address configured on a virtual interface as the device source for all TFTP packets, enter commands such as the following.

```
device(config)# interface ve 1
device(config-vif-1)# ip address 10.0.0.3/24
device(config-vif-1)# exit
device(config)# ip tftp source-interface ve 1
```

The commands in this example configure virtual interface 1, assign IP address 10.0.0.3/24 to the interface, then designate the interface's address as the source address for all TFTP packets.

The default is the lowest-numbered IP address configured on the port through which the packet is sent. The address therefore changes, by default, depending on the port.

Syslog packets

To specify the lowest-numbered IP address configured on a virtual interface as the device source for all Syslog packets, enter commands such as the following.

```
device(config)# interface 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.

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.

SNTP packets

To specify the lowest-numbered IP address configured on a virtual interface as the device source for all SNTP packets, enter commands such as the following.

```
device(config)# interface ve 1
device(config-vif-1)# ip address 10.0.0.5/24
device(config-vif-1)# exit
device(config)# ip sntp source-interface ve 1
```

The commands in this example configure virtual interface 1, assign IP address 10.0.0.5/24 to the interface, then designate the interface's address as the source address for all SNTP packets.

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.

SNMP packets

To specify a loopback interface as the SNMP single source trap, enter commands such as the following.

```
device(config)# interface loopback 1
device(config-lbif-1)# ip address 10.0.0.1/24
device(config-lbif-1)# exit
device(config)# snmp-server trap-source loopback 1
```

The commands in this example configure loopback interface 1, assign IP address 10.00.1/24 to the loopback interface, then designate the interface as the SNMP trap source for this device. Regardless of the port the Ruckus device uses to send traps to the receiver, the traps always arrive from the same source IP address.

Displaying IP configuration information

You can use various show commands to view the IP configuration on a device.

In this task, commands and options are shown to allow you to view the following IP configuration information statistics on Layer 3 devices:

- Global IP parameter settings and IP access policies
- IP interfaces
- IP forwarding cache
- IP route table

1. Enter global configuration mode.

   ```
   device# configure terminal
   ```

2. (Optional) To enable CIDR format for displaying network masks, entering the following command.

   ```
   device(config)# ip show-subnet-length
   ```

   By default, the CLI displays network masks in classical IP address format (example: 255.255.255.0). This command displays the network mask in CIDR format with a /prefix (for example /24).

3. Use the **show ip** command to display general IP configuration information.

```
device# show ip

Global Settings
  ttl: 64, arp-age: 10, bootp-relay-max-hops: 4
  router-id : 10.95.11.128
 enabled : UDP-Broadcast-Forwarding  Source-Route  Load-Sharing  RARP  OSPF  VRRP-Extended  VSRP
 disabled: Route-Only  Directed-Broadcast-Forwarding  BGP4 IRDP  Proxy-ARP  RIP  VRRP  ICMP-Redirect
Static Routes
  Index   IP Address        Subnet Mask       Next Hop Router   Metric Distance
  1       0.0.0.0           0.0.0.0                                    10.157.23.2       1
1
Policies
  Index   Action   Source           Destination      Protocol   Port  Operator
  1       deny               10.157.22.34             10.157.22.26     tcp        http  =
  64      permit   any              any
```

The display shows the local IP address, ARP aging timer, protocols that are enabled, static route information including the next hop address, any policies that are configured to filter the traffic, and other configurable options.

4. Use the **show ip interface** command to display IP interface information.

```
device# show ip interface

Interface         IP-Address      OK?  Method    Status            Protocol
Ethernet 1/1/1    10.95.6.173     YES  NVRAM     up                up
Ethernet 1/1/2    10.3.3.3        YES  manual    up                up
Loopback 1        10.2.3.4        YES  NVRAM     down              down
```

The example shows IP addresses configured for specific interfaces and the status of the interface.

5. Use the show ip interface command with specific options to display detailed IP information for a specific interface.

```
device# show ip interface ve 1

Interface Ve 1
members: ethe 1/1/4 to 1/1/24 ethe 1/1/27 to 1/1/48 ethe 1/2/1 to 1/2/2 ethe 2/1/1 to 2/1/2
ethe 2/1/4 to 2/1/12 ethe 2/1/15 to 2/1/24 ethe 2/2/1 to 2/2/2 ethe 3/1/1 to 3/1/2 ethe 3/1/4 to
3/1/12
ethe 3/1/14 to 3/1/24 ethe 3/2/3 to 3/2/4 ethe 4/1/1 to 4/1/12 ethe 4/1/15 to 4/1/24 ethe 4/2/3 to
4/2/4
ethe 5/1/1 to 5/1/12 ethe 5/1/14 to 5/1/24 ethe 5/2/3
active: ethe 4/2/4
port enabled
port state: UP
ip address: 10.66.66.66       subnet mask: 255.255.255.0
Port belongs to VRF: default-vrf
encapsulation: ETHERNET, mtu: 9216, metric: 1
directed-broadcast-forwarding: disabled
ICMP redirect: enabled
proxy-arp: disabled
ip arp-age:  10 minutes
No Helper Addresses are configured.
No inbound ip access-list is set
No outgoing ip access-list is set
```

The example shows more detailed information about virtual interface 1 and its configured, or default, options.

6. Use the **show ip route** command to display the IP route table.

```
device# show ip route

Total number of IP routes: 514
Start index: 1  B:BGP D:Connected  R:RIP  S:Static  O:OSPF *:Candidate default
Destination       NetMask          Gateway         Port   Cost   Type
10.1.0.0          255.255.0.0      10.1.1.2        1/1/1    2      R
10.2.0.0          255.255.0.0      10.1.1.2        1/1/1    2      R
10.3.0.0          255.255.0.0      10.1.1.2        1/1/1    2      R
10.4.0.0          255.255.0.0      10.1.1.2        1/1/1    2      R
10.5.0.0          255.255.0.0      10.1.1.2        1/1/1    2      R
10.6.0.0          255.255.0.0      10.1.1.2        1/1/1    2      R
10.7.0.0          255.255.0.0      10.1.1.2        1/1/1    2      R
10.8.0.0          255.255.0.0      10.1.1.2        1/1/1    2      R
10.9.0.0          255.255.0.0      10.1.1.2        1/1/1    2      R
10.10.0.0         255.255.0.0      10.1.1.2        1/1/1    2      S
```

7. Use the **direct** option of the **show ip route** command to display only direct routes that go to devices directly attached to the Layer 3 device.

```
device# show ip route direct

Start index: 1  B:BGP D:Connected  R:RIP  S:Static  O:OSPF *:Candidate default
   Destination       NetMask          Gateway         Port   Cost   Type
   10.157.22.0       255.255.255.0    0.0.0.0         1/4/11   1      D
```

Notice that the route displayed in this example has "D" in the Type field, indicating the route is to a directly connected device.

8. Use the **static** option of the **show ip route** command to display configured static IP routes.

```
device# show ip route static

Start index: 1  B:BGP D:Connected  R:RIP  S:Static  O:OSPF *:Candidate default
   Destination       NetMask          Gateway        Port   Cost   Type
   10.144.33.11     255.255.255.0     10.157.22.12   1/1/1    2      S
```

Notice that the route displayed in this example has "S" in the Type field, indicating the route is static.

9. Use the **show ip cache** command to display the IP forwarding cache.

```
device# show ip cache

Total number of cache entries: 3
D:Dynamic  P:Permanent  F:Forward  U:Us  C:Complex Filter
W:Wait ARP  I:ICMP Deny  K:Drop  R:Fragment  S:Snap Encap
      IP Address       Next Hop         MAC             Type  Port  Vlan  Pri
1     192.168.1.11     DIRECT           0000.0000.0000  PU    n/a         0
2     192.168.1.255    DIRECT           0000.0000.0000  PU    n/a         0
3     255.255.255.255  DIRECT           0000.0000.0000  PU    n/a         0
```

Disabling IP checksum check

The **disable-hw-ip-checksum-check** command traps a packet with bad checksum to the CPU. Previously, if the packet processor detected a packet with, for example, the checksum 0xFFFF, it would treat it as a bad checksum even if it was correct and it would drop the packet. Now, the command **disable-hw-ip-checksum-check** traps the packet at the CPU and if the checksum is correct, it forwards the packet.

NOTE
The **disable-hw-ip-checksum-check** command only functions with IPv4. IPv6 does not support this command.

To set disable hardware ip checksum check for all ports, enter the following command.

```
device# disable-hw-ip-checksum-check
disable-ip-header-check set for all ports
```

To clear disable hardware ip checksum check on all ports, enter the following command.

```
device# no disable-hw-ip-checksum-check ethernet 13
disable-hw-ip-checksum-check cleared for ports the 13 to 24
```

To set disable hardware ip checksum check on for example, port range 0-12, enter the following command.

```
device# disable-hw-ip-checksum-check ethernet 2
disable-ip-header-check set for ports ethe 1 to 12
```

To set disable hardware ip checksum check on, for example, port range 13-24, enter the following command.

```
device# disable-hw-ip-checksum-check ethernet 22
disable-ip-header-check set for ports ethe 13 to 24
```

To clear disable hardware ip checksum check on, for example, port range 13-24, enter the following command.

```
device# no disable-hw-ip-checksum-check ethernet 13
disable-hw-ip-checksum-check cleared for ports the 13 to 24
```

NOTE
The port range could be any consecutive range, it may not necessarily be a decimal number.

Clearing IP routes

The entire route table or specific individual routes can be cleared, if required.

When an interface subnet route with an interface address that directly matches a host route learned from a neighboring device is configured and subsequently removed, the **clear ip route** command should be used so that the learned route is updated in the Routing and Hardware Forwarding table.

To clear all routes from the IP route table, enter the following command.

```
device# clear ip route
```

To clear route 10.157.22.0/24 from the IP routing table, enter the **clear ip route** command.

```
device# clear ip route 10.157.22.0/24
```

Displaying IP traffic statistics

IP traffic information is tracked for IP packets, ICMP messages, and various protocol statistics such TCP, UDP and RIP. To display IP traffic statistics, enter the **show ip traffic** command at any CLI level.

```
device# show ip traffic

IP Statistics
  139 received, 145 sent, 0 forwarded
  0 filtered, 0 fragmented, 0 reassembled, 0 bad header
  0 no route, 0 unknown proto, 0 no buffer, 0 other errors
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 addr mask
```

```
  0 addr mask reply, 0 irdp advertisement, 0 irdp solicitation
Sent:
  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 addr mask
  0 addr mask reply, 0 irdp advertisement, 0 irdp solicitation
UDP Statistics
  1 received, 0 sent, 1 no port, 0 input errors
TCP Statistics
  0 active opens, 0 passive opens, 0 failed attempts
  0 active resets, 0 passive resets, 0 input errors
  138 in segments, 141 out segments, 4 retransmission
RIP Statistics
  0 requests sent, 0 requests received
  0 responses sent, 0 responses received
  0 unrecognized, 0 bad version, 0 bad addr family, 0 bad req format
  0 bad metrics, 0 bad resp format, 0 resp not from rip port
  0 resp from loopback, 0 packets rejected
```

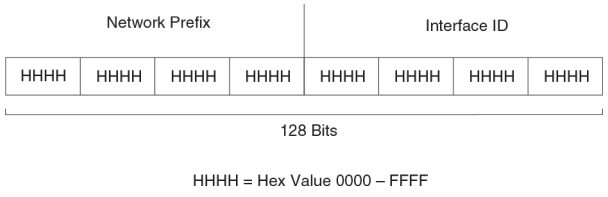
IPv6 Addressing

IPv6 addressing overview

IPv6 increases the number of network address bits from 32 (IPv4) to 128 bits, which provides more unique IP addresses to support increasing number of network devices.

An IPv6 address comprise 8 fields of 16-bit hexadecimal values separated by colons (:). The following figure shows the IPv6 address format.

FIGURE 9 IPv6 address format



As shown in the above figure, HHHH is a 16-bit hexadecimal value, while H is a 4-bit hexadecimal value. The following is an example of an IPv6 address.

2001:0000:0000:0200:002D:D0FF:FE48:4672

Note that this IPv6 address includes hexadecimal fields of zeros. To make the address manageable, you can:

- Omit the leading zeros. For example, 2001:0:0:200:2D: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::200:2D:D0FF:FE48:4672.

When specifying an IPv6 address in a command syntax, consider the following:

- You can use the two colons (::) only once in the address to represent the longest successive hexadecimal fields of zeros.
- The hexadecimal letters in IPv6 addresses are not case-sensitive.

As shown in Figure 9, 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/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 network portion 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 switch interface. IPv6 address types presents the three major types of IPv6 addresses that you can assign to a switch 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 site-local addresses and link-local addresses. IPv6 address types describes global, site-local, and link-local addresses and the topologies in which they are used.
- Multicast addresses support a scope field, which IPv6 address types describes.

TABLE 7 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: • 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. • Site-local address--An address used within a site or intranet. (This address is similar to a private IPv4 address.) A site consists of multiple network links. The address structure is as follows: a fixed prefix of FEC0::/10 (1111 1110 11), a 16-bit subnet ID, and a 64-bit interface ID. • Link-local address--An address used between directly connected nodes on 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. • 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. • Loopback address--An address (0:0:0:0:0:0:0:1 or ::1) that a switch can use to send an IPv6 packet to itself. You cannot assign a loopback address to a physical interface. • Unspecified address--An address (0:0:0:0:0:0:0:0 or ::) that a node can use until you configure an IPv6 address for it.
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).

TABLE 7 IPv6 address types (continued)

Address type	Description	Address structure
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. An anycast address can be assigned to a switch only. An anycast address must not be used as the source address of an IPv6 packet.

A switch 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.

NOTE
Ruckus FastIron devices support RFC 2526, which requires that within each subnet, the highest 128 interface identifier values reserved for assignment as subnet anycast addresses. Thus, if you assign individual IPv6 addresses within a subnet, the second highest IPv6 address in the subnet does not work.

IPv6 stateless auto-configuration

Ruckus routers use the IPv6 stateless autoconfiguration 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 switch on a local link periodically sends switch 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 switch advertisement messages must always be 64 bits.

The IPv6 stateless autoconfiguration 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 switch 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 switch advertisements. At this point, only addresses that contain the new prefix are used on the link.

Full Layer 3 IPv6 feature support

The following IPv6 Layer 3 features are supported only with the IPv6 Layer 3 PROM, Software-based Licensing, IPv6-series hardware, and the full Layer 3 image:

- OSPF V3
- RIPng
- IPv6 ICMP redirect messages
- IPv6 route redistribution
- IPv6 over IPv4 tunnels in hardware
- IPv6 Layer 3 forwarding
- BGP4+
- IPv6 Multicast routing
- DHCPv6 Relay Agent

NOTE
IPv6 static routes and IPv6 unicast routing (multicast routing is not supported) are not supported in the base Layer 3 software images.

IPv6 CLI command support

TABLE 8 IPv6 CLI command support

IPv6 command	Description	Switch code	Router code
clear ipv6 cache	Deletes all entries in the dynamic host cache.		X
clear ipv6 mld-snooping	Deletes MLD-snooping-related counters or cache entries.	X	X
clear ipv6 neighbor	Deletes all dynamic entries in the IPv6 neighbor table.	X	X
clear ipv6 ospf	Clears OSPF-related entries.		X
clear ipv6 rip	Clears RIP-related entries.		X
clear ipv6 route	Deletes all dynamic entries in the IPv6 route table.		X
clear ipv6 traffic	Resets all IPv6 packet counters.	X	X
clear ipv6 tunnel	Clears statistics for IPv6 tunnels		X
copy tftp	Downloads a copy of a Ruckus software image from a TFTP server into the system flash using IPv6.	X	X
debug ipv6	Displays IPv6 debug information.	X	X
ipv6 access-class	Configures access control for IPv6 management traffic.	X	X

TABLE 8 IPv6 CLI command support (continued)

IPv6 command	Description	Switch code	Router code
ipv6 access-list	Configures an IPv6 access control list for IPv6 access control.	X	X
ipv6 address	Configures an IPv6 address on an interface (router) or globally (switch)	X	X
ipv6 debug	Enables IPv6 debugging.	X	X
ipv6 dns domain-name	Configures an IPv6 domain name.	X	X
ipv6 dns server-address	Configures an IPv6 DNS server address.	X	X
ipv6 enable	Enables IPv6 on an interface.	X	X
ipv6 hop-limit	Sets the IPv6 hop limit.		X
ipv6 icmp	Configures IPv6 ICMP parameters		X
Ipv6 load-sharing	Enables IPv6 load sharing		X
Ipv6 mld-snooping	Configures MLD snooping	X	X
ipv6 mtu	Configures the maximum length of an IPv6 packet that can be transmitted on a particular interface.		X
ipv6 nd	Configures neighbor discovery.		X
ipv6 neighbor	Maps a static IPv6 address to a MAC address in the IPv6 neighbor table.		X
ipv6 ospf	Configures OSPF V3 parameters on an interface.		X
ipv6 prefix-list	Builds an IPv6 prefix list.		X
ipv6 redirects	Enables the sending of ICMP redirect messages on an interface.		X
ipv6 rip	Configures RIPng parameters on an interface		X
ipv6 route	Configures an IPv6 static route.		X
ipv6 router	Enables an IPv6 routing protocol.		X
ipv6 traffic-filter	Applies an IPv6 ACL to an interface.	X	X
ipv6 unicast-routing	Enables IPv6 unicast routing.		X
log host ipv6	Configures the IPv6 Syslog server.	X	X
ping ipv6	Performs an ICMP for IPv6 echo test.	X	X
show ipv6	Displays some global IPv6 parameters, such IPv6 DNS server address.	X	X
show ipv6 access-list	Displays configured IPv6 access control lists.	X	X
show ipv6 cache	Displays the IPv6 host cache.		X
show ipv6 interface	Displays IPv6 information for an interface.		X

TABLE 8 IPv6 CLI command support (continued)

IPv6 command	Description	Switch code	Router code
show ipv6 mld-snooping	Displays information about MLD snooping.	X	X
show ipv6 neighbor	Displays the IPv6 neighbor table.	X	X
show ipv6 ospf	Displays information about OSPF V3.		X
show ipv6 prefix-lists	Displays the configured IPv6 prefix lists.		X
show ipv6 rip	Displays information about RIPng.		X
show ipv6 route	Displays IPv6 routes.		X
show ipv6 router	Displays IPv6 local routers.		X
show ipv6 tcp	Displays information about IPv6 TCP sessions.	X	X
show ipv6 traffic	Displays IPv6 packet counters.	X	X
show ipv6 tunnel	Displays information about IPv6 tunnels	X	X
snmp-client ipv6	Restricts SNMP access to a certain IPv6 node.	X	X
snmp-server host ipv6	Specifies the recipient of SNMP notifications.	X	X
telnet	Enables a Telnet connection from the Ruckus device to a remote IPv6 host using the console.	X	X
traceroute ipv6	Traces a path from the Ruckus device to an IPv6 host.	X	X
web access-group ipv6	Restricts Web management access to certain IPv6 hosts as determined by IPv6 ACLs.	X	X
web client ipv6	Restricts Web management access to certain IPv6 hosts.	X	X

IPv6 host address on a Layer 2 switch

In a Layer 3 (router) configuration, each port can be configured separately with an IPv6 address. This is accomplished using the interface configuration process that is described in IPv6 configuration on each router interface on page 107.

Ruckus devices provide support for configuring an IPv6 address on the management port as described in Configuring the management port for an IPv6 automatic address configuration on page 106, and for configuring a system-wide IPv6 address on a Layer 2 switch. Configuration of the system-wide IPv6 address is exactly similar to configuration of an IPv6 address in router mode, except that the IPv6 configuration is at the Global CONFIG level instead of at the Interface level.

The process for defining the system-wide interface for IPv6 is described in the following sections:

- Configuring a global or site-local IPv6 address with a manually configured interface ID on page 106
- Configuring a link-local IPv6 address as a system-wide address for a switch on page 106

NOTE
When configuring an IPv6 host address on a Layer 2 switch that has multiple VLANs, make sure that the configuration includes a designated management VLAN that identifies the VLAN to which the global IP address belongs. Refer to "Designated VLAN for Telnet management sessions to a Layer 2 Switch" section in the *Ruckus FastIron Security Configuration Guide*.

Configuring a global or site-local IPv6 address with a manually configured interface ID

To configure a global or site-local IPv6 address with a manually-configured interface ID, such as a system-wide address for a switch, enter a command similar to the following at the Global CONFIG level.

```
(config)# ipv6 address 2001:DB8:12D:1300:240:D0FF:FE48:4000:1/64
```

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 in decimal value. A slash mark (/) must follow the ipv6-prefix parameter and precede the prefix-length parameter.

Configuring a link-local IPv6 address as a system-wide address for a switch

To enable IPv6 and automatically configure a global interface enter commands such as the following.

```
device(config)# ipv6 enable
```

This command enables IPv6 on the switch and specifies that the interface is assigned an automatically computed link-local address.

To override a link-local address that is automatically computed for the global interface with a manually configured address, enter a command such as the following.

```
device(config)# ipv6 address FE80::240:D0FF:FE48:4672 link-local
```

This command explicitly configures the link-local address FE80::240:D0FF:FE48:4672 for the global interface.

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 router interface should use the manually configured link-local address instead of the automatically computed link-local address.

Configuring the management port for an IPv6 automatic address configuration

You can have the management port configured to automatically obtain an IPv6 address for your Ruckus device. This process is the same for any other port and is described in detail in the section Configuring a global or site-local IPv6 address on an interface on page 108.

Configuring basic IPv6 connectivity on a Layer 3 switch

To configure basic IPv6 connectivity on a Ruckus Layer 3 Switch, you must:

- Enable IPv6 routing globally on the switch
- Configure an IPv6 address or specifically enable IPv6 on each router interface over which you plan to forward the IPv6 traffic
- Configure IPv4 and IPv6 protocol stacks. (This task is mandatory only if you want a router interface to send and receive both IPv4 and IPv6 traffic.)

All other configuration tasks in this chapter are optional.

NOTE

- When you configure an IPv6 address on a device, a syslog appears stating that the IPv6 address has been added. You cannot configure the same IPv6 address on the device again.
- When you attempt to configure the same IPv6 address on the device, this syslog appears: "`Error: duplicate IPv6 address!`".
- When you configure a different IPv6 address on the device, two different syslogs appear stating that the existing IPv6 address has been removed and the new IPv6 address has been configured.

Enabling IPv6 routing

By default, IPv6 routing is disabled. To enable the forwarding of IPv6 traffic globally on the Layer 3 switch, use the **ipv6 unicast-routing** command.

```
device# configure terminal
device(config)# ipv6 unicast-routing
```

The **no** form of the command disables the forwarding of IPv6 traffic globally on the device.

IPv6 configuration on each router interface

To forward IPv6 traffic on a router interface, the interface must have an IPv6 address, or IPv6 must be enabled. By default, an IPv6 address is not configured on a router interface. You must enable IPv6 and configure an IPV6 address to forward IPv6 traffic on a router interface.

If you choose to configure a global or site-local IPv6 address for an interface, IPv6 is also enabled on the interface. Further, when you configure a global or site-local IPv6 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 specifically enable IPv6 on the interface, 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 site-local 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

Configuring a global or site-local IPv6 address on an interface

Configuring a global or site-local IPv6 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.
- Solicited-node for subnet anycast address for each unicast assigned address
- Solicited-node for anycast address FF02:0:0:0:0:1:FF00::0000
- 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 IPv6 neighbor discovery configuration on page 120.

Enabling IPv6 on an interface

You can enable IPv6 at the interface level.

1. Enter global configuration mode.

```
device# configure terminal
```

2. Enter interface configuration mode.

```
device(config)# interface ethernet 3/1
```

3. Enable IPv6 for the interface.

```
device(config-if-e1000-3/1)# ipv6 enable
```

The following example enables IPv6 for an Ethernet interface.

```
device# configure terminal
device(config)# interface ethernet 3/1
device(config-if-e1000-3/1)# ipv6 enable
```

NOTE

In the example above, the interface is assigned an automatically computed link-local address. When configuring VLANs that share a common tagged interface with a physical or Virtual Ethernet (VE) interface, Ruckus recommends 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 physical and VE interfaces is derived from a global MAC address, all physical and 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, refer to Configuring a link-local IPv6 address on an interface on page 109.

Configuring a link-local IPv6 address on an interface

You can configure link-local IPv6 addresses at the interface level.

1. Enter global configuration mode.

```
device# configure terminal
```

2. Enter interface configuration mode.

```
device(config)# interface ethernet 3/1
```

3. Configure a link-local IPv6 address for the interface.

```
device(config-if-e1000-3/1)# ipv6 address FE80::240:D0FF:FE48:4672 link-local
```

The following example explicitly configures a link-local IPv6 address for an Ethernet interface.

```
device# configure terminal
device(config)# interface ethernet 3/1
device(config-if-e1000-3/1)# ipv6 address FE80::240:D0FF:FE48:4672 link-local
```

Configuring an IPv6 anycast address on an interface

In IPv6, an anycast address is 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 configured with the 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 1/2/1.

```
device(config)# interface ethernet 1/2/1
device(config-if-e1000-1/2/1)# ipv6 address 2001:DB8::/64 anycast
```

IPv6 anycast addresses are described in detail in RFC 1884. Refer to RFC 2461 for a description of how the IPv6 Neighbor Discovery mechanism handles anycast addresses.

Supporting both IPv4 and IPv6 addresses on an interface.

A router interface can be configured to support both the IPv4 and IPv6 protocol stacks.

One situation in which you must configure a router to run both IPv4 and IPv6 protocol stacks is if it is deployed as an endpoint for an IPv6 over IPv4 tunnel.

Each router interface that sends and receives both the IPv4 and IPv6 traffic must be configured with an IPv4 address and an IPv6 address. In this task, IPv6 is globally enabled, an IPv4 address and an IPv6 address are configured for a specific interface.

1. Enter global configuration mode.

```
device# configure terminal
```

2. Enable IPv6 routing globally.

```
device(config)# ipv6 unicast-routing
```

3. Enter Ethernet interface configuration mode.

```
device(config)# interface ethernet 1/3/1
```

4. Configure an IPv4 address on Ethernet interface 1/3/1.

```
device(config-if-e1000-1/3/1)# ip address 10.168.1.1 255.255.255.0
```

5. Configure an IPv6 address on Ethernet interface 1/3/1.

```
device(config-if-e1000-1/3/1)# ipv6 address 2001:DB8:12d:1300::/64 eui-64
```

The following example configures both an IPv4 and an IPv6 address on an interface to allow both IPv4 and IPv6 traffic to be sent and received over this interface.

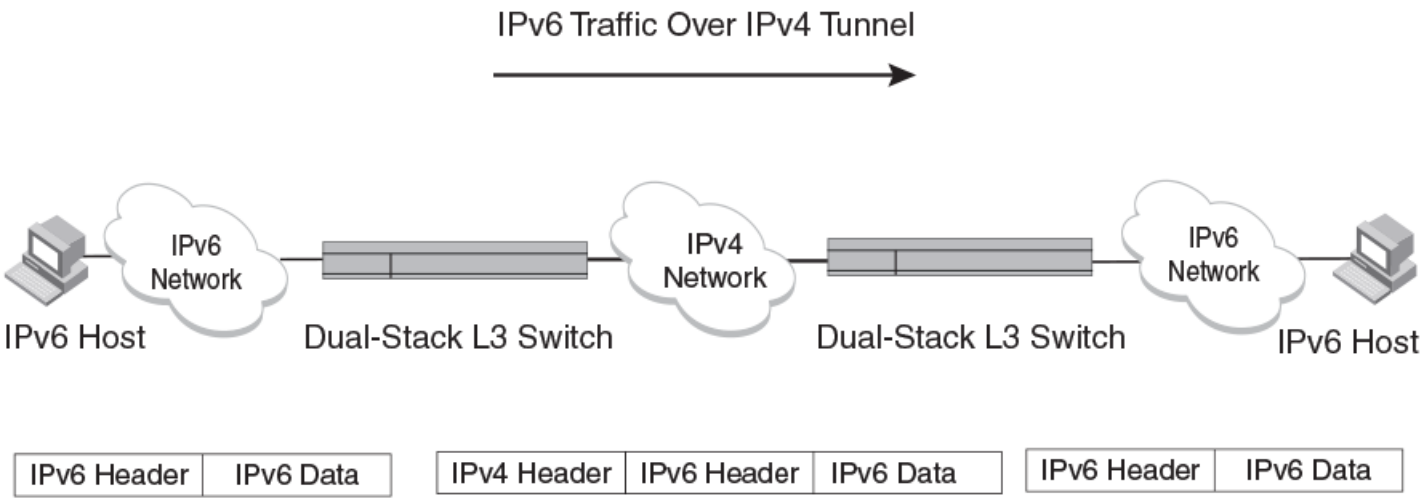
```
device# configure terminal
device(config)# ipv6 unicast-routing
device(config)# interface ethernet 1/3/1
device(config-if-e1000-1/3/1)# ip address 10.168.1.1 255.255.255.0
device(config-if-e1000-1/3/1)# ipv6 address 2001:DB8:12d:1300::/64 eui-64
```

IPv6 over IPv4 tunnels

To enable communication between isolated IPv6 domains using the IPv4 infrastructure, you can manually configure IPv6 over IPv4 tunnels that provide static point-point connectivity.

As shown in the following illustration, these tunnels encapsulate an IPv6 packet within an IPv4 packet.

FIGURE 10 IPv6 over an IPv4 tunnel



A manually configured tunnel establishes a permanent link between switches in IPv6 domains. A manually configured tunnel has explicitly configured IPv4 addresses for the tunnel source and destination.

This tunneling mechanism requires that the Layer 3 switch at each end of the tunnel run both IPv4 and IPv6 protocol stacks. The Layer 3 switches running both protocol stacks, or dual-stack routers, can interoperate directly with both IPv4 and IPv6 end systems and routers.

Refer to the "IPv4 and IPv6 protocol stacks" section.

NOTE
ICX 7150 devices do not support tunnels.

IPv6 over IPv4 tunnel configuration notes

- The local tunnel configuration must include both source and destination addresses.
- The remote side of the tunnel must have the opposite source/destination pair.
- A tunnel interface supports static and dynamic IPv6 configuration settings and routing protocols.
- Duplicate Address Detection (DAD) is not currently supported with IPv6 tunnels. Make sure tunnel endpoints do not have duplicate IP addresses.
- Neighbor Discovery (ND) is not supported with IPv6 tunnels.
- If a tunnel source port is a multi-homed IPv4 source, the tunnel will use the first IPv4 address only. For proper tunnel operation, use the **ip address** option.

NOTE
ICX 7150 devices do not support tunnels.

Configuring a manual IPv6 over IPv4 tunnel

You can use a manually configured tunnel to connect two isolated IPv6 domains. You must deploy this point-to-point tunneling mechanism if you need a permanent and stable connection across an IPv4 network.

Work in progress!

1. Enter global configuration mode.

```
device# configure terminal
```

2. Enter tunnel interface configuration mode.

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

3. Assign a port name to the tunnel.

```
device(config-tunnel1)# port-name ManualTunnel1
```

4. Configure tunnel mode for IPv6

```
device(config-tunnel1)# tunnel mode ipv6ip
```

5. a)

   b)

```
device(mode)# command executable
interface tunnel 1
 port-name ManualTunnel1
 tunnel mode ipv6ip
 tunnel source loopback 1
 tunnel destination 10.1.1.1
 ipv6 address 1011::1/64
 ipv6 address 1001::1/64
 ipv6 ospf area 0
```

Connecting IPv6 tunnels

You can use a manually configured tunnel to connect two isolated IPv6 domains. You must deploy this point-to-point tunneling mechanism if you need a permanent and stable connection.

Clearing IPv6 tunnel statistics

You can clear statistics (reset all fields to zero) for all IPv6 tunnels or for a specific tunnel interface.

For example, to clear statistics for tunnel 1, enter the following command at the Privileged EXEC level or any of the configuration levels of the CLI.

```
device# clear ipv6 tunnel 1
```

To clear statistics for all IPv6 tunnels, enter the following command.

```
device# clear ipv6 tunnel
```

NOTE
ICX 7150 devices do not support tunnels.

Displaying IPv6 tunnel information

Use the commands in this section to display the configuration, status, and counters associated with IPv6 tunnels.

NOTE
ICX 7150 devices do not support tunnels.

Displaying a summary of tunnel information

To display a summary of tunnel information, enter the **show ipv6 tunnel** command at any level of the CLI.

```
device# show ipv6 tunnel
IP6 Tunnels
  Tunnel  Mode         Packet Received  Packet Sent
  1       configured   0                0
  2       configured   0                22419
```

This display shows the following information.

TABLE 9 IPv6 tunnel summary information

Field	Description
Tunnel	The tunnel interface number.
Mode	The tunnel mode. Possible modes include the following: • configured - Indicates a manually configured tunnel.
Packet Received	The number of packets received by a tunnel interface. Note that this is the number of packets received by the CPU. It does not include the number of packets processed in hardware.
Packet Sent	The number of packets sent by a tunnel interface. Note that this is the number of packets sent by the CPU. It does not include the number of packets processed in hardware.

NOTE
ICX 7150 devices do not support tunnels.

Displaying interface level IPv6 settings

To display interface level IPv6 settings for tunnel interface, enter the **show ipv6 inter tunnel** command.

```
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 1011::1/64
 ipv6 address 1001::1/64
 ipv6 ospf area 0
```

TABLE 10 Interface level IPv6 tunnel information

Field	Description
Interface Tunnel status	The status of the tunnel interface can be one of the following: • up - IPv4 connectivity is established. • down - The tunnel mode is not set. • administratively down - The tunnel interface was disabled with the **disable** command.
Line protocol status	The status of the line protocol can be one of the following: • up - IPv6 is enabled through the **ipv6 enable** or **ipv6 address** command. • down - The line protocol is not functioning and is down.

NOTE
ICX 7150 devices do not support tunnels.

IPv6 management

You can configure a Ruckus device to serve as an IPv6 host in an IPv6 network. An IPv6 host has IPv6 addresses on its interfaces, but does not have full IPv6 routing enabled on it.

Configuring IPv6 management ACLs

When you enter the **ipv6 access-list** command, the Ruckus device enters the IPv6 Access List configuration level, where you can access several commands for configuring IPv6 ACL entries. After configuring the ACL entries, you can apply them to network management access features such as Telnet, SSH, Web, and SNMP.

NOTE
Unlike IPv4, there is no distinction between standard and extended ACLs in IPv6.

```
device(config)#ipv6 access-list netw
device(config-ipv6-access-list-netw)#
```

The *ACL-name* variable specifies a name for the IPv6 ACL. An IPv6 ACL name cannot start with a numeral, for example, 1access. Also, an IPv4 ACL and an IPv6 ACL cannot share the same name.

Restricting SNMP access to an IPv6 node

Use the **snmp-client ipv6** command to restrict SNMP access to the device to the IPv6 host whose IP address you specify.

```
device# config terminal
device(config)# snmp-client ipv6 2001:DB8:89::23
```

Specifying an IPv6 SNMP trap receiver

You can specify an IPv6 host as a trap receiver to ensure that all SNMP traps sent by the device will go to the same SNMP trap receiver or set of receivers, typically one or more host devices on the network. To do so, enter a command such as the following.

```
device# configure terminal
device(config)# snmp-server host ipv6 2001:DB8:89::13
```

Configuring SNMP V3 over IPv6

Ruckus devices support IPv6 for SNMP version 3. For more information about how to configure SNMP, refer to *Ruckus FastIron Management Configuration Guide* .

Secure Shell, SCP, and IPv6

Secure Shell (SSH) is a mechanism that allows secure remote access to management functions on the Ruckus device. SSH provides a function similar to Telnet. You can log in to and configure a Ruckus 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 Ruckus device.

To open an SSH session between an IPv6 host running an SSH client program and the Ruckus device, open the SSH client program and specify the IPv6 address of the device. For more information about configuring SSH on the Ruckus device, refer to "SSH2 and SCP" chapter in the *Ruckus FastIron Security Configuration Guide*.

IPv6 Telnet

Telnet sessions can be established between a Ruckus device to a remote IPv6 host, and from a remote IPv6 host to the Ruckus device using IPv6 addresses.

The **telnet** command establishes a Telnet connection from a Ruckus device to a remote IPv6 host using the console. Up to five read-access Telnet sessions are supported on the router at one time. Write-access through Telnet is limited to one session, and only one outgoing Telnet session is supported on the router at one time.

Use the **show telnet** command to see the number of open Telnet sessions at any time.

Establishing a Telnet session from an IPv6 host

To establish a Telnet session from an IPv6 host to the Ruckus device, open your Telnet application and specify the IPv6 address of the Layer 3 Switch.

IPv6 traceroute

Use the **traceroute** command to trace a path from the Ruckus device to an IPv6 host.

```
device# traceroute ipv6 2001:DB8:349e:a384::34
```

The **traceroute** command displays trace route information for each hop as soon as the information is received. The traceroute requests display all responses of a minimum TTL of 1 second and a maximum TTL of 30 seconds. In addition, if there are multiple equal-cost routes to the destination, the Ruckus device displays up to three responses.

IPv6 Web management using HTTP and HTTPS

When you have an IPv6 management station connected to a Ruckus device with an IPv6 address applied to the management port, you can manage the device from a Web browser by entering one of the following in the browser address field.

http://[<ipv6 address>]

or

https://[<ipv6 address>]

> **NOTE**
> You must enclose the IPv6 address with square brackets [ ] in order for the Web browser to work.

Restricting Web management access

You can restrict Web management access to include only management functions on a Ruckus device that is acting as an IPv6 host, or restrict access so that the **Ruckus** host can be reached by a specified IPv6 device.

Restricting Web management access by specifying an IPv6 ACL

You can specify an IPv6 ACL that restricts Web management access to management functions on the device that is acting as the IPv6 host.

Example

```
device(config)# access-list 12 deny host 2000:2383:e0bb::2/128 log
device(config)# access-list 12 deny 30ff:3782::ff89/128 log
device(config)# access-list 12 deny 3000:4828::fe19/128 log
device(config)# access-list 12 permit any
device(config)# web access-group ipv6 12
```

Restricting Web management access to an IPv6 host

You can restrict Web management access to the device to the IPv6 host whose IP address you specify. No other device except the one with the specified IPv6 address can access the Web Management Interface.

Example

```
device(config)# web client ipv6 3000:2383:e0bb::2/128
```

The *ipv6-address* you specify must be in hexadecimal format using 16-bit values between colons as documented in RFC 2373.

Configuring name-to-IPv6 address resolution using IPv6 DNS resolver

The Domain Name Server (DNS) resolver feature lets you use a host name to perform Telnet and ping commands. You can also define a DNS domain on a Ruckus device and thereby recognize all hosts within that domain. After you define a domain name, the Ruckus 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 Ruckus 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 ipv6 nyc01
device# ping ipv6 nyc01.newyork.com
```

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. Ruckus 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. They store a complete IPv6 address in each record. AAAA records have a type value of 28.

To define an IPv6 DNS server address, enter command such as the following:

```
device(config)# ipv6 dns server-address 2001:DB8::1
```

As an example, in a configuration where ftp6.companynet.com is a server with an IPv6 protocol stack, when a user pings ftp6.companynet.com, the Ruckus 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.

Pinging an IPv6 address

The **ping** command allows you to verify the connectivity from a Ruckus 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 Ruckus device, enter the following command.

```
device# ping ipv6 2001:DB8:847f:a385:34dd::45
```

Configuring an IPv6 Syslog server

To enable IPv6 logging, specify an IPv6 Syslog server. Enter a command such as the following.

```
device(config)#log host ipv6 2000:2383:e0bb::4/128
```

The IPv6 address must be in hexadecimal using 16-bit values between colons as documented in RFC 2373.

Viewing IPv6 SNMP server addresses

Some of the **show** commands display IPv6 addresses for IPv6 SNMP servers. The following shows an example 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::200
                                162     .....

     3        10.147.202.100
               162     .....

     4                2001:DB8::200
                                162     .....
```

Disabling router advertisement and solicitation messages

Router advertisement and solicitation messages enable a node on a link to discover the routers on the same link. By default, router advertisement and solicitation messages are permitted on the device. To disable these messages, configure an IPv6 access control list that denies them. The following shows an example configuration.

```
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
```

Disabling IPv6 on a Layer 2 switch

IPv6 is enabled by default in the Layer 2 switch code. If desired, you can disable IPv6 on a global basis on a device running the switch code. To disable IPv6 , enter the following command in global configuration mode.

```
device(config)# no ipv6 enable
```

To re-enable IPv6 after it has been disabled, enter **ipv6 enable** .

NOTE
IPv6 is disabled by default in the router code and must be configured on each interface that will support IPv6.

IPv6 ICMP feature configuration

As with the Internet Control Message Protocol (ICMP) for IPv4, ICMP for IPv6 provides error and informational messages. Implementation of the stateless auto configuration, neighbor discovery, and path MTU discovery features use ICMP messages.

This section explains how to configure following IPv6 ICMP features:

- ICMP rate limiting
- ICMP redirects

ICMP rate limiting

You can limit the rate at which IPv6 ICMP error messages are sent out on a network. IPv6 ICMP implements a token bucket algorithm.

To illustrate how this algorithm works, imagine a virtual bucket that contains a number of tokens. Each token represents the ability to send one ICMP error message. Tokens are placed in the bucket at a specified interval until the maximum number of tokens allowed in the bucket is reached. For each error message that ICMP sends, a token is removed from the bucket. If ICMP generates a series of error messages, messages can be sent until the bucket is empty. If the bucket is empty of tokens, 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.

ICMP redirects

You can enable a Layer 3 switch to send an IPv6 ICMP redirect message to a neighboring host to inform it of a better first-hop router on a path to a destination. By default, the sending of IPv6 ICMP redirect messages by a Layer 3 switch is disabled. (For more information about how ICMP redirect messages are implemented for IPv6, refer to IPv6 neighbor discovery configuration on page 120.)

NOTE
This feature is supported on Virtual Ethernet (VE) interfaces only.

Enabling IPv6 ICMP redirects and configuring ICMP rate-limiting

You can enable a Layer 3 switch to send an IPv6 ICMP redirect message to a neighboring host to inform it of a better first-hop router on a path to a destination. and you can limit the rate at which IPv6 ICMP error messages are sent out on a network.

By default, the sending of IPv6 ICMP redirect messages by a Layer 3 switch is disabled. (For more information about how ICMP redirect messages are implemented for IPv6, refer to IPv6 neighbor discovery configuration on page 120.)

NOTE
This feature is supported on Virtual Ethernet (VE) interfaces only.

IPv6 ICMP implements a token bucket algorithm. To illustrate how this algorithm works, imagine a virtual bucket that contains a number of tokens. Each token represents the ability to send one ICMP error message. Tokens are placed in the bucket at a specified interval until the maximum number of tokens allowed in the bucket is reached. For each error message that ICMP sends, a token is removed from the bucket. If ICMP generates a series of error messages, messages can be sent until the bucket is empty. If the bucket is empty of tokens, error messages cannot be sent until a new token is placed in the bucket.

1. Enter global configuration mode.

```
device# configure terminal
```

2. To adjust the transmit interval between ICMP error messages to 1000 milliseconds and the maximum number of ICMP error messages that can be sent to 100, use the **ipv6 icmp error-interval** command.

```
device(config)# ipv6 icmp error-interval 1000 100
```

   ICMP rate limiting is enabled by default. To disable ICMP rate limiting, set the interval to zero.

   NOTE
   If you retain the default interval value or explicitly set the value to 100 milliseconds, output from the **show run** command does not include the setting of the **ipv6 icmp error-interval** command because the setting is the default. Also, if you configure the interval value to a number that does not evenly divide into 100000 (100 milliseconds), the system rounds up the value to a next higher value that does divide evenly into 100000. For example, if you specify an interval value of 150, the system rounds up the value to 200.

3. Enter virtual Ethernet (VE) interface mode for VE 2.

```
device(config)# interface ve2
```

4. To enable the sending of IPv6 ICMP redirect messages on virtual Ethernet (VE) interface 2, use the **ipv6 redirects** command.

```
device(config-vif-2)# ipv6 redirects
```

   To verify that the sending of IPv6 ICMP redirect messages is enabled on a particular interface, use the **show ipv6 interface** command.

The following example enables IPv6 ICMP redirect messages and configures IPv6 ICMP rate-limiting.

```
device# configure terminal
device(config)# ipv6 icmp error-interval 1000 100
device(config)# interface ve2
device(config-vif-2)# ipv6 redirects
```

IPv6 neighbor discovery configuration

The neighbor discovery feature for IPv6 uses IPv6 ICMP messages to perform the following tasks:

- 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 that identify itself:

- 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 or enable the following IPv6 neighbor discovery features:

- Neighbor solicitation messages for duplicate address detection.
- Router advertisement (RA) messages:
  - Interval between RA messages.
  - Value that indicates a router is advertised as a default router (for use by all nodes on a given link).
  - Prefixes advertised in RA messages.
  - Flags for host stateful autoconfiguration.
  - Domain names.
  - Recursive DNS server addresses.
  - IPv6 address advertisement suppression.
- Amount of time during which an IPv6 node considers a remote node reachable (for use by all nodes on a given link).

IPv6 neighbor discovery configuration notes

NOTE
For all solicitation and advertisement messages, Ruckus uses seconds as the unit of measure instead of milliseconds.

- Neighbor discovery is not supported on tunnel interfaces.

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 routers on the same link.

Each configured router 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 router 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 router Ethernet interfaces. You can configure several router advertisement message parameters.

Neighbor redirect messages

After forwarding a packet, by default, a router can send a neighbor redirect message to a host to inform it of a better first-hop router. The host receiving the neighbor redirect message will then readdress the packet to the better router.

A router 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.

Duplicate Address Detection (DAD)

Although the stateless auto configuration 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 auto configuration 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.

IPv6 router advertisement parameters

You can adjust the following parameters for router advertisement messages:

- The "router lifetime" value, which is included in router advertisements sent from a particular interface. The value (in seconds) indicates if the router is advertised as a default router on this interface. If you set the value of this parameter to 0, the router is not advertised as a default router on an interface. If you set this parameter to a value that is not 0, the router is advertised as a default router on this interface. By default, the router lifetime value included in router advertisement messages sent from an interface is 1800 seconds.
- The hop limit to be advertised in the router advertisement.

Prefixes advertised in IPv6 router advertisement messages

By default, router advertisement messages include prefixes configured as addresses on router 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, provided the specified prefix is aggregatable, as specified in RFC 2374.

The following example shows how to configure the prefixes that are advertised in the IPv6 router advertisement messages.

```
device(config)# interface ethernet 1/3/1
device(config-if-e1000-1/3/1)# ipv6 nd prefix-advertisement 2001:DB8:a487:7365::/64 1000 800 onlink
autoconfig
```

The **no** form of the command removes a prefix from the router advertisement messages sent from a particular interface.

Domain Name System Search List

Domain Name System Search List (DNSSL) is an IPv6 router advertisement (RA) option that allows IPv6 devices to advertise domain names of DNS suffixes to IPv6 hosts in a local area network.

The domain names that are advertised by routers are sent through RA messages to IPv6 hosts. The DNSSL option is supported only when IPv6 routing is active on the network.

Configuring DNSSL

Domain Name System Search List (DNSSL) allows IPv6 devices to advertise domain names of DNS suffixes to IPv6 hosts in a local area network. The following task configures two domain names to be advertised for an Ethernet interface, and sets separate lifetime multiplier values for each of the domain names configured.

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
```

3. Enter the **ipv6 nd ra-domain-name** command, specifying a domain name, to advertise the domain name in router advertisement (RA) messages. Use the **lifetime-multiplier** keyword, entering a value, to specify the value of the maximum RA interval.

```
device(config-if-e1000-1/1/3)# ipv6 nd ra-domain-name abc.net lifetime-multiplier 1
```

4. Enter the **ipv6 nd ra-domain-name** command, specifying a domain name, to advertise the domain name in RA messages. Use the **lifetime-multiplier** keyword, entering a value, to specify the value of the maximum RA interval.

```
device(config-if-e1000-1/1/3)# ipv6 nd ra-domain-name xyz.com lifetime-multiplier 2
```

The following example configures the domain name of a DNS suffix as "abc.net" and sets a lifetime multiplier value of 1 for an Ethernet interface. It also configures the domain name of a DNS suffix as "xyz.com" and sets a lifetime multiplier value of 2 for the Ethernet interface.

```
device# configure terminal
device(config)# interface ethernet 1/1/3
device(config-if-e1000-1/1/3)# ipv6 nd ra-domain-name abc.net lifetime-multiplier 1
device(config-if-e1000-1/1/3)# ipv6 nd ra-domain-name xyz.com lifetime-multiplier 2
```

Recursive DNS server addresses

Recursive DNS server (RDNSS) addresses is an IPv6 router advertisement (RA) feature that allows IPv6 devices to advertise recursive DNS server addresses and lifetime values to IPv6 hosts in a local area network.

RDNSS contains the addresses of recursive DNS servers that help in DNS name resolution and can be configured on any network that supports neighbor discovery (ND). The configured recursive server addresses are advertised by devices through RA messages, and are used to translate domain names into IP addresses.

Configuring recursive DNS server addresses

RDNSS addresses allow IPv6 devices to advertise recursive DNS server addresses and lifetime multiplier values to IPv6 hosts in a local area network. The following task configures two DNS server addresses to be advertised for an Ethernet interface in router advertisement (RA) messages, and sets separate lifetime multiplier values for each of the DNS server addresses configured.

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
```

3. Enter the **ipv6 nd ra-dns-server** command, specifying an IPv6 address, to configure a DNS server to be advertised in RA messages. Use the **lifetime-multiplier** keyword, entering a value, to specify the maximum RA interval.

```
device(config-if-e1000-1/1/3)# ipv6 nd ra-dns-server 2001::1 lifetime-multiplier 1
```

4. Enter the **ipv6 nd ra-dns-server** command, specifying an IPv6 address, to configure a DNS server to be advertised in RA messages. Use the **lifetime-multiplier** keyword, entering a value, to specify the value of the maximum RA interval.

```
device(config-if-e1000-1/1/3)# ipv6 nd ra-dns-server 2001::2 lifetime-multiplier 2
```

The following example configures a DNS server with the IPv6 address 2001::1 to be advertised in RA messages, with a lifetime multiplier of 1. It also configures a DNS server with the IPv6 address 2001::2 to be advertised in RA messages, with a lifetime multiplier of 2.

```
device# configure terminal
device(config)# interface ethernet 1/1/3
device(config-if-e1000-1/1/3)# ipv6 nd ra-dns-server 2001::1 lifetime-multiplier 1
device(config-if-e1000-1/1/3)# ipv6 nd ra-dns-server 2001::2 lifetime-multiplier 2
```

IPv6 address advertisement suppression

IPv6 address advertisement suppression is an IPv6 router advertisement (RA) option that suppresses the advertisement of specified or all IPv6 addresses for router advertisement messages on an interface.

By default, prefix information in RA messages includes IPv6 addresses configured on the interface. Specific IPv6 addresses, or all IPv6 addresses configured on the interface, can be excluded from RA message advertisement by using IPv6 address advertisement suppression.

Suppressing the advertisement of all configured IPv6 addresses

By default, prefix information in router advertisement (RA) messages includes IPv6 addresses configured on the interface. The following task suppresses the advertisement of all configured IPv6 addresses in RA messages for an interface.

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
```

3. Enter the **ipv6 nd suppress-ra address** command with the **all** keyword to suppress all IPv6 addresses configured on the interface in RA messages.

```
device(config-if-e1000-1/1/3)# ipv6 nd suppress-ra address all
```

The following example suppresses all IPv6 addresses configured on the interface in RA messages.

```
device# configure terminal
device(config)# interface ethernet 1/1/3
device(config-if-e1000-1/1/3)# ipv6 nd suppress-ra address all
```

Suppressing the advertisement of selected IPv6 addresses

By default, prefix information in router advertisement (RA) messages includes IPv6 addresses configured on the interface. The following task suppresses the advertisement of a specified IPv6 address in RA messages for an interface.

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
```

3. Enter the **ipv6 nd suppress-ra address** command, specifying an IPv6 address, to suppress the advertisement of that IPv6 address in RA messages.

```
device(config-if-e1000-1/1/3)# ipv6 nd suppress-ra address 2001::1
```

The following example suppresses the IPv6 address 2001::1 in RA messages. All other IPv6 addresses configured on the interface are advertised.

```
device# configure terminal
device(config)# interface ethernet 1/1/3
device(config-if-e1000-1/1/3)# ipv6 nd suppress-ra address 2001::1
```

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 nonaddress 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.

Enabling and disabling IPv6 router advertisements

If IPv6 unicast routing is enabled on an Ethernet interface, by default, this interface sends IPv6 router advertisement messages. However, by default, non-LAN interface types, for example, tunnel interfaces, do not send router advertisement messages.

Use the **ipv6 nd suppress-ra** command to disable the sending of router advertisement messages on an Ethernet interface.

```
device# configure terminal
device(config)# interface ethernet 1/3/1
device(config-if-e1000-1/3/1)# ipv6 nd suppress-ra
```

To enable the sending of router advertisement messages on a tunnel interface, enter commands such as the following.

```
device# configure terminal
device(config)# interface tunnel 1
device(config-tnif-1)# no ipv6 nd suppress-ra
```

NOTE
ICX 7150 devices do not support tunnels.

IPv6 router advertisement preference support

IPv6 router advertisement (RA) preference enables IPv6 RA messages to communicate default router preferences from IPv6 routers to IPv6 hosts in network topologies where the host has multiple routers on its Default Router List. This improves the ability of the IPv6 hosts to select an appropriate router for an off-link destination.

Configuring IPv6 RA preference

Configuring IPv6 RA preference

If IPv6 unicast routing is enabled on an Ethernet interface, by default, this interface sends IPv6 router advertisement messages. The IPv6 router sets the preference field based on the configured value on IPv6 RA and sends it periodically to the IPv6 host or as a response to the router solicitations.

1. Enter global configuration mode.

```
device# configuration terminal
```

2. Use the **ipv6 nd router-preference** command to configure IPv6 RA preference for the IPv6 router.

```
device(config)# interface ethernet 1/2/3
device(config-if-e1000-1/2/3)# ipv6 nd router-preference low
```

Reachable time for remote IPv6 nodes

The router advertisement messages sent by a router interface include the duration of time specified so that nodes on a link use the same reachable time duration. By default, the messages include a default value of 0.

Ruckus recommends configuring a longer reachable time duration, because a short duration causes the IPv6 network devices to process the information at a greater frequency.

IPv6 Neighbor Discovery Proxy

IPv6 Neighbor Discovery(ND) Proxy enables the hosts in different broadcast domains or VLANs to communicate with each other.

An IPv6 ND Proxy-enabled interface responds to a neighbor discovery request on behalf of a host connected to another interface. An ND proxy-configured egress interface modifies the source link-layer information of the packet with its own link-layer address and the MAC address in the L2 header with its own mac address. The presence of host neighbor cache entry in the neighbor entry table verifies the reachability state.

ND proxy uses ICMPv6 messages such as neighbor solicitation and neighbor advertisement to determine the link-layer address. Neighbor solicitation messages determine the link-layer address of a neighboring device by using the solicited-node multicast address of the target host. A Neighbor advertisement message is the unicast reply to a neighbor solicitation message.

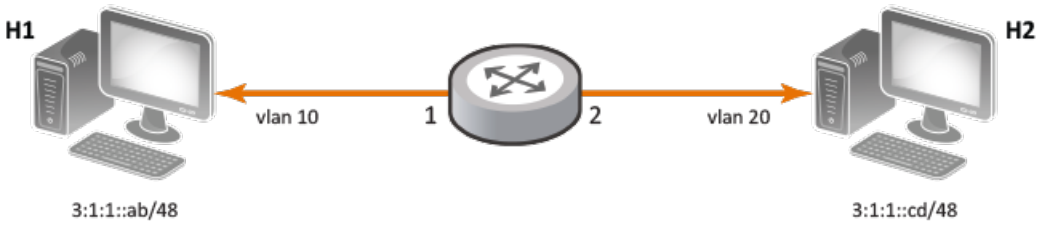
IPv6 ND Proxy includes ND proxy and local ND proxy. ND proxy for IPv6 functions similarly to IPv4 proxy ARP. The Ruckus implementation of IPv6 ND Proxy is based on RFC 4389.

NOTE
IPv6 ND Proxy is supported only on the router and is disabled by default.

IPv6 ND Proxy

IPv6 ND Proxy enables the hosts in different broadcast domains with identical subnet masks to communicate with each other.

FIGURE 11 IPv6 ND Proxy



In the example, host H1 wants to send packets to host H2. However, host H1 does not know the link-layer address of destination host H2 or is in a different broadcast domain which is directly connected to the proxy router. When ND proxy is enabled on the router globally, the hosts on different broadcast domains with identical subnet masks respond with the link-layer address. The source H1 multicasts an ICMPv6 neighbor solicitation message using its MAC address and Ipv6 address to find the link-layer address. The solicited-node multicast MAC address corresponding to the destination host H2 will be the destination MAC address. The target address is the destination IPv6 address. The neighbor solicitation message gets forwarded to the router. The router will update its neighbor cache table with this neighbor solicitation message. The neighbor solicitation packet contains the following information:

- Source MAC address: MAC address of host H1
- Destination MAC address: Solicited-node multicast MAC address of destination host H2
- IPv6 source address: Global unique IPv6 address of host H1
- IPv6 destination address: Solicited-node multicast address of destination host H2
- Target address: Global unique IPv6 address of host H2

The outgoing proxy-enabled interface 2 modifies the received neighbor solicitation packet before sending it to the destination H2. The proxy interface generates a proxy neighbor solicitation packet. The source link-layer address of the neighbor solicitation packet will be changed to link-layer address of the ND proxy-enabled interface 2 in the proxy neighbor solicitation packet. The proxy neighbor solicitation packet contains the following information:

- Source MAC address: MAC address of host H1 will be changed to MAC address of proxy interface 2
- Destination MAC address: Solicited-node multicast MAC address of destination host H2
- IPv6 source address: Global unique IPv6 address of egress interface 2
- IPv6 destination address: Solicited-node multicast address of destination host H2
- Target address: Global unique IPv6 address of host H2

The proxy neighbor solicitation message from proxy interface 2 is forwarded to the destination host H2. Host H2 gives replies to this proxy neighbor solicitation message because the solicited-node multicast address is derived from its own IPv6 address. The destination host H2 replies by sending a neighbor advertisement message that carries its link-layer information. The neighbor

advertisement message reaches the router and updates the neighbor table. The neighbor advertisement packet contains the following information:

- Source MAC address: MAC address of host H2
- Destination MAC address: MAC address of proxy interface 2
- IPv6 source address: Source global IPv6 address of host H2
- IPv6 destination address: The link-local address of proxy interface 2

The neighbor advertisement packet reaches the router. This router initiates a proxy neighbor advertisement packet for the second neighbor solicitation request from the source host H2 because the neighbor cache table has the details of target host H1. The proxy neighbor advertisement packet contains the following information:

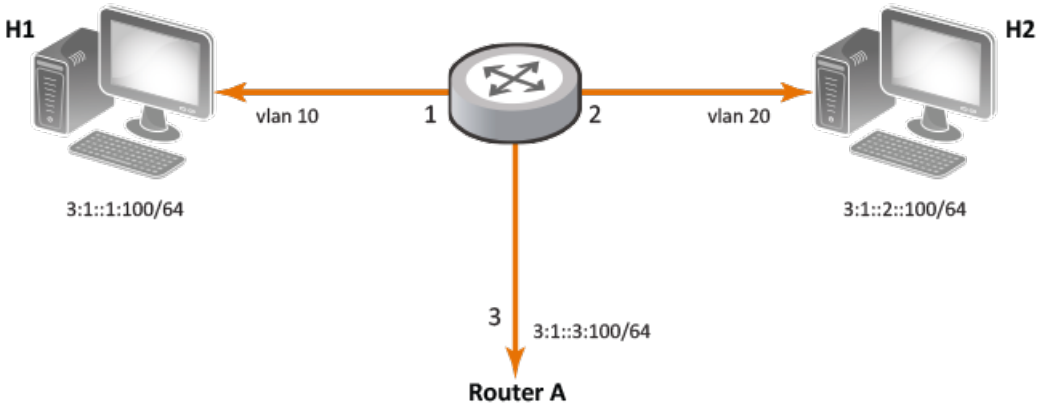
- Source MAC address: MAC address of proxy interface 1 instead of the H2 MAC address
- Destination MAC address: MAC address of host H1
- IPv6 source address: Source global IPv6 address of egress interface 1
- IPv6 destination address: Destination address of host H1

After the host H1 receives the proxy neighbor advertisement packet, both the hosts in different networks communicate with each other.

Local ND Proxy

Local ND proxy allows communication between hosts in different VLANs with identical subnet masks or hosts that are part of private or an isolated VLAN.

FIGURE 12 LOCAL ND PROXY



Local ND proxy can be used to enable ND proxy on specific interfaces in a router. In the example, host 1 and host 2 are on the same IPv6 subnet. However, host 1 resides in VLAN 10 and host 2 resides in VLAN 20. Host 1 sends a neighbor solicitation request directly to host 2 because they are on the same IPv6 subnet. Because host 1 and host 2 are on different VLANs, host 2 will not receive this neighbor solicitation request. Local ND proxy must be enabled on interface 3 of router A to forward messages between host 1 and host 2. The generation of the proxy neighbor advertisement packet works as explained in IPv6 ND Proxy on page 127.

Configuring IPv6 ND Proxy Globally

IPv6 ND proxy in global configuration mode enables all the interfaces on the router.

1. Enter global configuration mode.

```
Router# configure terminal
```

2. Enable the forwarding of IPv6 traffic globally.

```
Router(config)# ipv6 unicast-routing
```

3. Configure IPv6 ND proxy in global configuration mode.

```
Router(config)# ipv6 nd proxy
```

4. Use the **show ipv6** command to verify whether IPv6 ND proxy is enabled on the router.

```
Router(config)# show ipv6

           Global Settings
  unicast-routing disabled, hop-limit 64
  No Inbound Access List Set
  No Outbound Access List Set
  No IPv6 Domain Name Set
  No IPv6 DNS Server Address set
  Prefix-based IPv6 Load-sharing is Enabled, Number of load share paths: 4
  nd proxy enabled
```

The following example configures IPv6 ND proxy on an IPv6 enabled router.

```
Router# configure terminal
Router(config)# ipv6 unicast-routing
Router(config)# ipv6 nd proxy
```

Configuring Local ND Proxy on an interface

IPv6 ND Proxy can be used to configure individual interfaces on a router.

1. Enter global configuration mode.

```
Router# configure terminal
```

2. Specify the interface to be configured in interface configuration mode.

```
Router(config)# interface ethernet 1/1/3
```

3. Enable IPv6 routing for the interface. The **ipv6 enable** command configures a router interface with an automatically computed link-local address.

```
Router(config-if-e1000-1/1/3)# ipv6 enable
```

4. Enable IPv6 ND Proxy using the **ipv6 nd local-proxy** command.

```
Router(config-if-e1000-1/1/3)# ipv6 nd local-proxy
```

5. Use the **show ipv6 neighbor proxy** command to display host entries for which the router acts as the proxy.

```
Router(config)# show ipv6 neighbor proxy

          Neighbor Entries in VRF: default-vrf
IPv6 Address                            LinkLayer-Addr State           Age        Port
vlan  IsR
4:1:1::100                              609c.9f52.241c REACH           31         e 1/1/25
1     1
4:1:1::99                               609c.9f52.f20d REACH           31         e 1/1/25
1     1
```

The following example configures IPv6 ND Proxy on the ethernet interface 1/1/3. The interface is manually configured with a link-local address.

```
Router# configure terminal
Router(config)# interface ethernet 1/1/3
Router(config-if-e1000-1/1/3)# ipv6 address FE80::240:D0FF:FE48:4672 link-local
Router(config-if-e1000-1/1/3)# ipv6 nd proxy
```

Disabling IPv6 ND Proxy

IPv6 ND Proxy can be disabled on an individual port or a range of ports.

1. Enter global configuration mode.

```
Router# configure terminal
```

2. Specify the interface to be disabled in interface configuration mode.

```
Router(config)# interface ethernet 1/2/1
```

3. Disable IPv6 ND proxy on the specified interface

```
Router(config-if-e10000-1/2/1)# ipv6 nd proxy-disable
```

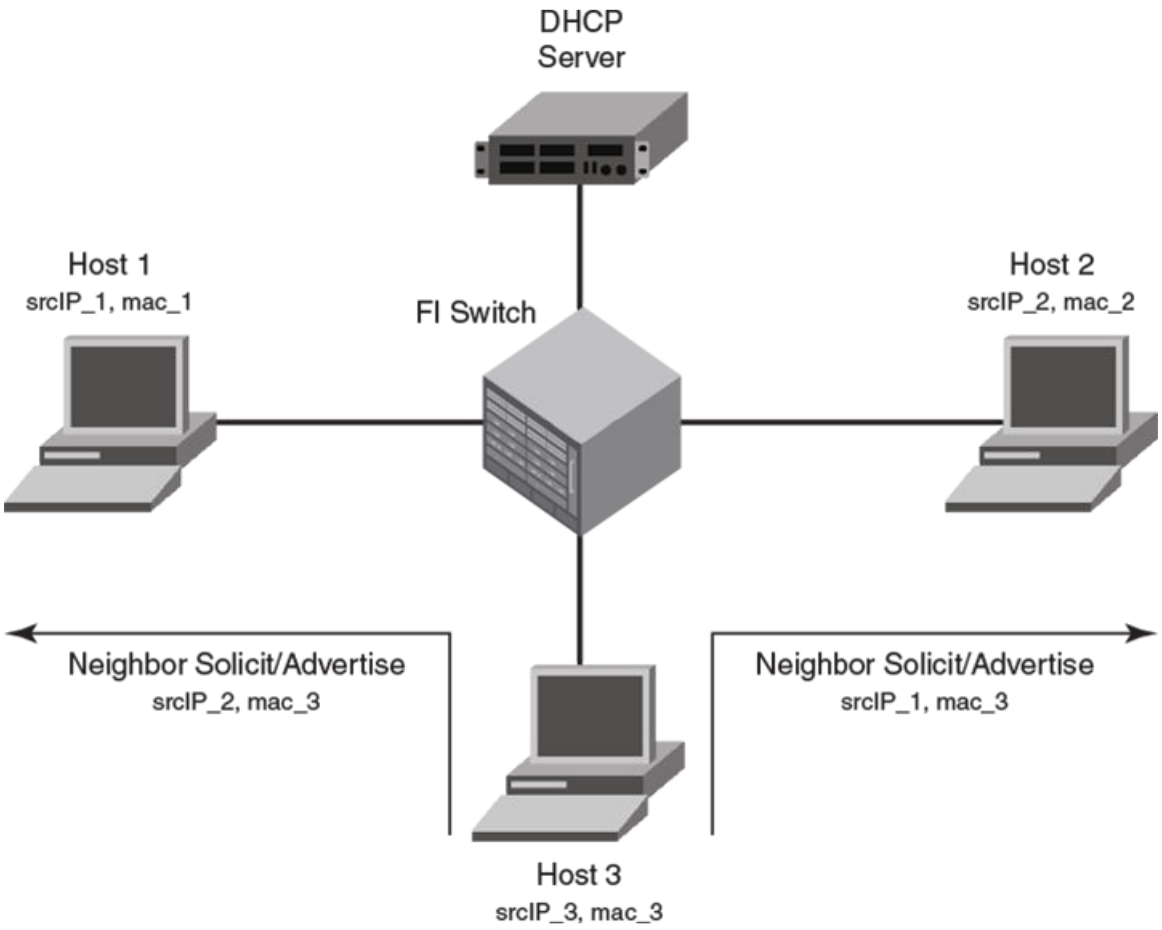
IPv6 neighbor discovery inspection

IPv6 neighbor discovery (ND) inspection is an internal network security system that detects and prevents IPv6 address spoofing at the switch level.

IP communication within a Layer 2 infrastructure is established by mapping an IP address to a MAC address. An invalid host can intercept packet flow between legitimate hosts by sending a neighbor solicitation or neighbor advertisement with a forged IP-to-MAC address binding. The victim host includes an illegitimate entry in the neighbor cache, which is looked up to validate the IP-to-MAC address binding. After a successful attack, all the traffic are redirected through the invalid host and is vulnerable to man-in-the-middle attacks. The neighbor discovery inspection validates all the IPv6 packets carrying neighbor discovery messages by checking the IP-to-MAC address binding of the packets. If there is a discrepancy in the IP-to-MAC address binding, the neighbor discovery message is considered to be from an invalid host and the packets are discarded.

The following figure illustrates the method by which Host 3 performs ND cache poisoning by sending a neighbor solicitation message to Host 1 with the source IP of Host 2, and similarly to Host 2 with the source IP of Host 1, with its own MAC address. By doing this, Host 3 can intercept the packet flow from Host 1 to Host 2.

FIGURE 13 Neighbor discovery cache poisoning



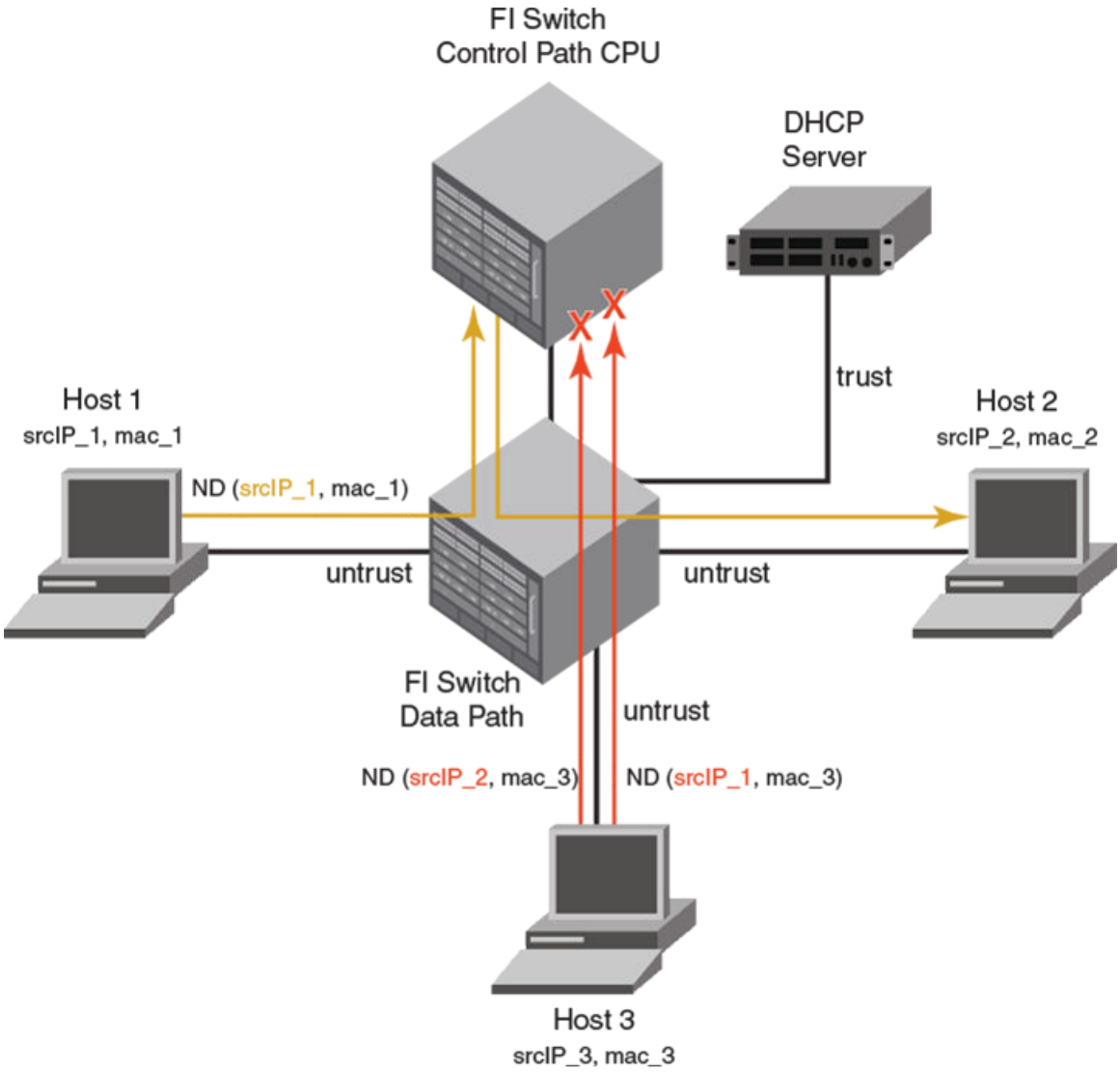
ND inspection, when enabled on a VLAN, checks all the neighbor discovery messages flowing through the switches between the hosts that are part of the VLAN and validates the IP-to-MAC address binding of the packets. All the packets are verified against the trusted binding tables where the preconfigured static ND inspection entries or dynamically learned DHCPv6 snoop entries are stored. DHCPv6 snooping must be enabled for dynamic inspection of ND messages. For more information on dynamically learned DHCPv6 snoop entries, refer to the DHCPv6 section in the *Ruckus FastIron DHCP Configuration Guide*.

To inspect a neighbor discovery message, all the neighbor solicitation and neighbor advertisement messages are directed to a CPU, and the source IP address and source MAC address of each packet are validated against the entries in the trusted tables. Only the valid packets are forwarded and those with invalid IP-to-MAC address bindings are discarded. ND inspection follows CPU-based packet forwarding and thus the neighbor discovery messages in the ND inspection-enabled VLAN may get discarded depending on the CPU load. The neighbor discovery messages are also rate limited to CPU.

The router interface configuration on the ND inspection-enabled VLAN is also subjected to ND inspection. That is, if the interface is a Layer 3 interface, the neighbor solicitation and neighbor advertisement messages addressed to the router are also validated. If there is a discrepancy in the IP-to-MAC address binding, the packets are discarded and the IPv6 neighbor tables will not be updated. Unlike the neighbor solicitation and neighbor advertisement messages, the router solicitation messages are not directed to the CPU, because the hosts are supposed to reject the router solicitation messages by default.

The following figure illustrates unhindered flow of packets from Host 1 to Host 2, while the messages that are sent by Host 3 with invalid IP-to-MAC address bindings are discarded.

FIGURE 14 Neighbor discovery inspection



Though you can configure interfaces in “trust” or “untrust” mode, ND inspection is performed only on untrusted ports that are part of the ND inspection-enabled VLAN. When you enable ND inspection on a VLAN, by default, all the interfaces and member ports are considered as untrusted. When configured, ND inspection protects the directly connected hosts from ND cache poisoning; the hosts connected across the switches are not insulated from any attack.

When configured, ND inspection performs the following functions:

- Intercepts and inspects the IPv6 packets that carry neighbor discovery messages on untrusted ports.
- Validates the source IP addresses and the source MAC addresses of the intercepted packets against the IP-to-MAC address bindings stored in a trusted binding database.
- Forwards the packets which have valid IP-to-MAC address bindings to the destination host and discards the invalid packets. The ICMPv6 packets with auto-generated link-local address (from the MAC address) are also forwarded, provided there is a match between MAC address and the auto-generated link-local address. Hence, there is no need of separate configuration of auto-generated link-local address in the ND inspection database.

NOTE
ND inspection is supported on LAGs and trunk ports and supports Multi-VRF instances. Multiple VRFs can be deployed on a Ruckus Ethernet switch. Each VLAN having a Virtual Interface (VE) is assigned to a VRF.

Configuring neighbor discovery inspection

The ND inspection configuration task includes enabling ND inspection on a VLAN, adding static inspection entries, and enabling trust mode for switch or server ports. To configure neighbor discovery inspection, complete the following steps.

The acl-per-port-per-vlan must be enabled using the **enable acl-per-port-per-vlan** command before configuring ND inspection.

1. Use the **ipv6 neighbor inspection vlan** command to enable ND inspection on a VLAN.

```
device(config)# ipv6 neighbor inspection vlan 10
```

2. Use **ipv6 neighbor inspection** command to add a static ND inspection entry. You can add multiple static ND inspection entries.

```
device(config)# ipv6 neighbor inspection 2001::1 0000.1234.5678
```

3. Use the **interface ethernet** command to enter the interface configuration mode.

```
device(config)# interface ethernet 1/1/1
```

4. Use **ipv6-neighbor inspection trust** command to enable trust mode for the switch or server port. You can enable trust mode for multiple ports.

```
device(config-if-e1000-1/1/1)# ipv6-neighbor inspection trust
```

Configuring neighbor discovery inspection on multiple VLANs

Neighbor discovery (ND) inspection can be enabled on multiple VLANs using one command. The following task configures multiple VLANs and enables ND inspection on most of the configured VLANs using one command.

The ACL per port per VLAN must be enabled using the **enable acl-per-port-per-vlan** command before configuring ND inspection.

1. Enter global configuration mode.

```
device# configure terminal
```

2. Configure the port-based VLANs.

```
device(config)# vlan 100 to 150
```

3. Add port Ethernet 1/1/12 as a tagged port.

```
device(config-mvlan-100-150)# tagged ethernet 1/1/12
```

4. Use the **exit** command to return to global configuration mode.

```
device(config-mvlan-100-150)# exit
```

5. Configure more port-based VLANs.

```
device(config)# vlan 151 to 200
```

6. Add port Ethernet 1/1/12 as a tagged port.

```
device(config-mvlan-151-200)# tagged ethernet 1/1/12
```

7. Use the **exit** command to return to global configuration mode.

```
device(config-mvlan-151-200)# exit
```

8. Use the **ipv6 neighbor inspection vlan** command with the **to** keyword, specifying a VLAN range, to enable ND inspection on multiple VLANs.

```
device(config)# ipv6 neighbor inspection vlan 100 to 150 160 170 to 200
```

   The command enables ND inspection on VLANs 100 through 150, VLAN 160, and VLANs 170 through 200.

9. Use the **ipv6 neighbor inspection** command to add a static ND inspection entry. You can add multiple static ND inspection entries.

```
device(config)# ipv6 neighbor inspection 2001::1 0000.1234.5678
```

10. Use the **interface ethernet** command to enter the interface configuration mode.

```
device(config)# interface ethernet 1/1/1
```

11. Use the **ipv6-neighbor inspection trust** command to enable trust mode for the switch or server port. You can enable trust mode for multiple ports.

```
device(config-if-e1000-1/1/1)# ipv6-neighbor inspection trust
```

The following example configures multiple VLANs and enables ND inspection on VLANS 100 through 150, VLAN 160, and VLANs 170 through 200. It also designates port 1/1/1 as trusted.

```
device# configure terminal
device(config)# vlan 100 to 150
device(config-mvlan-100-150)# tagged ethernet 1/1/12
device(config-mvlan-100-150)# exit
device(config)# vlan 151 to 200
device(config-mvlan-151-200)# tagged ethernet 1/1/12
device(config-mvlan-100-150)# exit
device(config)# ipv6 neighbor inspection vlan 100 to 150 160 170 to 200
device(config)# ipv6 neighbor inspection 2001::1 0000.1234.5678
device(config)# interface ethernet 1/1/1
device(config-if-e1000-1/1/1)# ipv6-neighbor inspection trust
```

Syslog message for ND inspection

The following table lists the syslog message related to ND inspection.

TABLE 11 Syslog message related to ND inspection

Event	Syslog output
Rejected ND	ND Inspect: no static inspect or DHCP6 entry found, packet dropped rx-sip 2001::100 rx-smac 0000.0000.0055 vlan_id 2 vrf_id 0

IPv6 MTU

The IPv6 maximum transmission unit (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 fragments the packet and transmits its contents in multiple packets that are shorter than the configured MTU.

By default, in non-jumbo mode, the default and maximum Ethernet MTU size is 1500 bytes. When jumbo mode is enabled, the default Ethernet MTU size is 9216. The maximum Ethernet MTU size is 10178.

For ICX 7850 devices, the maximum jumbo frame size supported is 9380 bytes. When jumbo mode is enabled, the maximum ethernet MTU size is 9358 bytes.

Configuration notes and feature limitations for IPv6 MTU

- The IPv6 MTU functionality is applicable to VEs and physical IP interfaces. It applies to traffic routed between networks.
- For ICX 7150, ICX 7250, ICX 7450, ICX 7650, and ICX 7750 devices, the IPv4 and IPv6 MTU values are the same. Modifying one also changes the value of the other. For ICX 7850 devices, the maximum mtu is 9380 bytes in jumbo mode.
- For ICX 7150, ICX 7250, ICX 7450, ICX 7650, ICX 7750, and ICX 7850 devices, the minimum IPv4 and IPv6 MTU values for both physical and virtual interfaces are 1280.
- You cannot use IPv6 MTU to set Layer 2 maximum frame sizes per interface. Enabling global jumbo mode causes all interfaces to accept Layer 2 frames.

Changing the IPv6 MTU

You can configure the IPv6 MTU on individual interfaces.

```
device(config)# interface ethernet 1/3/1
device(config-if-e1000-1/3/1)# ipv6 mtu 1280
```

For bytes, specify a value between 1280 - 1500, or 1280 - 10178 if jumbo mode is enabled. If a non-default value is configured for an interface, router advertisements include an MTU option.

NOTE
IPv6 MTU cannot be configured globally. It is supported only on devices running Layer 3 software.

Static neighbor entries configuration

In some special cases, a neighbor cannot be reached using the neighbor discovery feature. 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.)

NOTE
A port that has a statically assigned IPv6 entry cannot be added to a VLAN.

NOTE
Static neighbor configurations will be cleared on secondary ports when a LAG is formed.

Use the **ipv6 neighbor** command to add a static entry for a neighbor .

```
device(config)# ipv6 neighbor 2001:DB8:2678:47b ethernet 1/3/1 0000.002b.8641
```

You configured IPv6 address 2001:DB8:2678:47b and link-layer address 0000.002b.8641 that is reachable through Ethernet interface 1/3/1. 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

You can change the maximum number of hops an IPv6 packet can traverse from the default of 64.

In global configuration mode, you can configure the maximum number of hops for an IPv6 packet to a value between 1 and 255 hops.

1. Enter global configuration mode.

   ```
   device# configure terminal
   ```

2. Configure the maximum number of hops for an IPv6 packet.

   ```
   device(config)# ipv6 hop-limit 70
   ```

   Use the **no** form or use the value of 0 to reset to the default value.

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 might leave a network open to a DoS attack.

A security enhancement disables sending IPv6 source-routed packets to IPv6 devices. (This enhancement conforms to RFC 5095.)

By default, when the router drops a source-routed packet, it sends an ICMP Parameter Problem (type 4), Header Error (code 0) message to the packet's source address, pointing to the unrecognized routing type.

TCAM space configuration

Ruckus devices store routing information for IPv4 and IPv6 routing in the same ternary content-addressable memory (TCAM) table.

Ruckus devices vary in the amount of TCAM space that can be allocated for IPv4 and IPv6 routing. Each IPv6 route entry uses more storage space then IPv4 route entries. The default, maximum, and minimum allocation values for each type of data are shown in the following tables.

NOTE
If you disable IPv6 routing, the TCAM space allocations do not change. If you want to allocate the maximum possible space for IPv4 routing information, you must configure the TCAM space manually.

NOTE
For the ICX 7850, values are used from the forwarding profile and there is no option to change these values. When a new profile is selected, the values from the new profile are applied. Refer to the **forwarding-profile** command in the *Ruckus FastIron Command Reference* and the *Configuration Fundamentals* chapter in the *Ruckus FastIron Management Configuration Guide* for more information.

TABLE 12 TCAM space allocation on ICX 7750 devices

	Default	Maximum	Minimum
IPv4 route entries	98304	131072	98304
IPv6 route entries	5120	7168	5120

TABLE 13 TCAM space allocation on ICX 7650 devices

	Default	Maximum	Minimum
IPv4 route entries	98304	131072	98304
IPv6 route entries	5120	7168	5120

TABLE 14 TCAM space allocation on ICX 7450 devices

	Default	Maximum	Minimum
IPv4 route entries	12000	15168	4096
IPv6 route entries	5120	5120	68

TABLE 15 TCAM space allocation on ICX 7250 devices

	Default	Maximum	Minimum
IPv4 route entries	12000	12000	4096
IPv6 route entries	730	2048	68

TABLE 16 TCAM space allocation on ICX 7150 devices

	Default	Maximum	Minimum
IPv4 route entries	800	1000	20
IPv6 route entries	220	1000	20

Allocating TCAM Space

The amount of TCAM space to allocate for IPv4 routing information can be configured. You must save the running configuration to the startup configuration and reload the device for the changes to take effect. After the reload, the remaining TCAM space is allocated automatically for IPv6 routing information.

TCAM space allocations for IPv4 and IPv6 routes and other entities can be modified by configuring the number of IPv4 route entries. Different devices have different amounts of TCAM space, see the "TCAM space allocation" topic for the default, maximum, and minimum allocations.

NOTE
If you disable IPv6 routing, the TCAM space allocations do not change. If you want to allocate the maximum possible space for IPv4 routing information, you must configure the TCAM space manually.

NOTE
The ICX 7150 and ICX 7250 device only permit manual configuration of IPv4 routes.

NOTE
For the ICX 7850, values are used from the forwarding profile and there is no option to change these values. When a new profile is selected, the values from the new profile are applied. Refer to the **forwarding-profile** command in the *Ruckus FastIron Command Reference* and the *Configuration Fundamentals* chapter in the *Ruckus FastIron Management Configuration Guide* for more information.

1. Enter global configuration mode.

```
device# configure terminal
```

2. To allocate TCAM space to store 6000 IPv4 routes entries, use the following command.

```
device(config)# system-max ip-route 6000
```

3. Copy the running configuration to the startup configuration.

```
device(config)# write memory
```

4. Return to privileged EXEC mode.

```
device(config)# exit
```

5. Reload the device for the new TCAM space allocations to be changed.

```
device# reload
```

The following example configures TCAM space for 6000 IPv4 route entries. After the device reload, you can view the new TCAM allocation numbers for IPv6 entries.

```
device# configure terminal
device(config)# system-max ip-route 6000
device(config)# write memory
device(config)# exit
device# reload
device# show run
.
.
.
ip-route: 6000
ip6-route 365
ip6-cache: 182
```

Displaying IPv6 global information

After configuring and enabling IPv6 in your network, you can use various **show** commands to view IPv6 information.

Before entering the commands in this task, you must have IPv6 enabled and configured in your network.

You can display output for the following global IPv6 parameters:

- IPv6 cache
- IPv6 interfaces
- IPv6 neighbors
- IPv6 route table

- IPv6 TCP connections and the status of individual connections
- IPv6 traffic statistics

NOTE
The following steps can be used in any order. For more details about the fields displayed in the output and other options for some of the commands, see the *Ruckus FastIron Command Reference*.

1. To display general IPv6 interface information, enter the **show ipv6 interface** command.

```
device> show ipv6 interface

Routing Protocols : R - RIP  O - OSPF
Interface           Status        Routing    Global Unicast Address
Ethernet 1/3/3      down/down     R
Ethernet 1/3/5      down/down
Ethernet 1/3/17     up/up                    2017::c017:101/64
Ethernet 1/3/19     up/up                    2019::c019:101/64
VE 4                down/down
VE 14               up/up                    2024::c060:101/64
Loopback 1          up/up                             ::1/128
Loopback 2          up/up                    2005::303:303/128
Loopback 3          up/up
```

2. To display detailed information for a specific interface, use the **show ipv6 interface** with specific interface options.

```
device> show ipv6 interface ethernet 1/3/1

Interface Ethernet 1/3/1 is up, line protocol is up
  IPv6 is enabled, link-local address is fe80::2e0:52ff:fe99:97
  Global unicast address(es):
  Joined group address(es):
    ff02::9
    ff02::1:ff99:9700
    ff02::2
    ff02::1
  MTU is 1500 bytes
  ICMP redirects are enabled
  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 1 seconds
  ND advertised retransmit interval is 0 seconds
  ND router advertisements are sent every 200 seconds
  ND router advertisements live for 1800 seconds
  No Inbound Access List Set
  No Outbound Access List Set
  RIP enabled
```

In this example, detailed information is displayed for the Ethernet interface 1/3/1.

3. To display the IPv6 neighbor table, enter the **show ipv6 neighbor** command.

```
device(config)# show ipv6 neighbor

Total number of Neighbor entries: 3
IPv6 Address              LinkLayer-Addr  State       Age   Port      vlan  IsR
2001:DB8::55              0000.0002.0002  *REACH      0     e 1/3/11  -     0
2000:4::110               0000.0091.bb37  REACH       20    e 1/3/1   5     1
fe80::2e0:52ff:fe91:bb37  0000.0091.bb37  DELAY       1     e 1/3/2   4     1
fe80::2e0:52ff:fe91:bb40  0000.0091.bb40  STALE       5930  e 1/3/3   5     1
```

4. To display IPv6 cache information, use the **show ipv6 cache** command.

```
device> show ipv6 cache

Total number of cache entries: 10
    IPv6 Address                 Next Hop                    Port
1   2001:DB8::2                  LOCAL                       tunnel 2
2   2001:DB8::106                LOCAL                       ethe 1/3/2
3   2001:DB8::110                DIRECT                      ethe 1/3/2
4   2001:DB8:46a::1              LOCAL                       ethe 1/3/2
5   2001:DB8::2e0:52ff:fe99:9737 LOCAL                       ethe 1/3/2
6   2001:DB8::ffff:ffff:feff:ffff LOCAL                      loopback 2
7   2001:DB8::c0a8:46a           LOCAL                       tunnel 2
8   2001:DB8::c0a8:46a           LOCAL                       tunnel 6
9   2001:DB8::1                  LOCAL                       loopback 2
10  2001:DB8::2e0:52ff:fe99:9700 LOCAL                       ethe 1/3/1
```

Other options are available to display detailed IPv6 cache information for a specific interface.

5. To display the IPv6 route table, use the **show ipv6 route** command.

```
device> show ipv6 route

IPv6 Routing Table - 7 entries:
Type Codes:  C - Connected, S - Static, R - RIP, O - OSPF, B - BGP
OSPF Sub Type Codes: O - Intra, Oi - Inter, O1 - Type1 external, O2 - Type2 external
Type  IPv6 Prefix            Next Hop Router           Interface     Dis/Metric
C     2000:4::/64              ::                       ethe 1/3/2    0/0
S     2001:DB8::/16            ::                       tunnel 6      1/1
S     2001:DB8:1234::/32       ::                       tunnel 6      1/1
C     2001:DB8:46a::/64        ::                       ethe 1/3/2    0/0
C     2001:DB8::1/128          ::                       loopback 2    0/0
O     2001:DB8::2/128       fe80::2e0:52ff:fe91:bb37    ethe 1/3/2    110/1
C     2001:DB8::/64            ::                       tunnel 2      0/0
```

Other options are available to display detailed information for a specific IPv6 address, IPv6 prefix, or a specific routing protocol.

6. To display general information about each TCP connection on the router, use the **show ipv6 tcp connections** command.

```
device> show ipv6 tcp connections

Local IP address:port <-> Remote IP address:port TCP state
10.168.182.110:23 <->    10.168.8.186:4933     ESTABLISHED
10.168.182.110:8218 <->  10.168.182.106:179    ESTABLISHED
10.168.182.110:8039 <->  10.168.2.119:179      SYN-SENT
10.168.182.110:8159 <->  10.168.2.102:179      SYN-SENT
2000:4::110:179 <->       2000:4::106:8222       ESTABLISHED (1440)
Total 5 TCP connections
TCP MEMORY USAGE PERCENTAGE
FREE TCP = 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
```

This example displays general information about each TCP connection on the router, including the percentage of free memory for each of the internal TCP buffers. Other options are available to display detailed information for a specific TCP connection.

7. To display IPv6 traffic statistics, use the **show ipv6 traffic** command.

```
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 not forward, 0 redirect sent
  0 frag recv, 0 frag dropped, 0 frag timeout, 0 frag overflow
  0 reassembled, 0 fragmented, 0 ofragments, 0 can not 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 not send error, 0 too freq
Sent Errors:
  0 unreach no route, 0 admin, 0 beyond scope, 0 address, 0 no port
  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
```

Displaying the IPv6 local router information

A device can function as an IPv6 host, instead of an IPv6 router, if you configure IPv6 addresses on its interfaces but do not enable IPv6 routing using the **ipv6 unicast-routing** command.

From the IPv6 host, you can display information about IPv6 routers to which the host is connected. The host learns about the routers through their router advertisement messages. To display information about the IPv6 routers connected to an IPv6 host, use the **show ipv6 router** command.

1. To display information about the IPv6 routers connected to an IPv6 host, use the **show ipv6 router** command.

```
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
```

   Meaningful output for this command is generated for Ruckus devices configured to function as IPv6 hosts only.

2. If you configure a device to function as an IPv6 router (you configure IPv6 addresses on its interfaces and enable IPv6 routing using the **ipv6 unicast-routing command** the **show ipv6 router** command displays the following.

```
No IPv6 router in table
```

Clearing global IPv6 information

You can clear the following global IPv6 information from a Ruckus device:

- Entries from the IPv6 cache
- Entries from the IPv6 neighbor table
- IPv6 routes from the IPv6 route table
- IPv6 traffic statistics

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

To remove entries for IPv6 address 2000:e0ff::1, use the **clear ipv6 cache** command.

```
device# clear ipv6 cache 2000:e0ff::1
```

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.

If you specify an Ethernet interface, also specify the port number associated with the interface. If you specify a VE, VRF, or tunnel interface, also specify the VE, VRF name, or tunnel number, respectively.

Clearing IPv6 neighbor information

You can remove all entries from the IPv6 neighbor table or specify an entry based on the IPV6 prefix, IPv6 address, and interface type.

- IPv6 prefix
- IPv6 address
- Interface type

To remove entries for Ethernet interface 1/3/1, use the **clear ipv6 neighbor** command.

```
device# clear ipv6 neighbor ethernet 1/3/1
```

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.

If you specify an Ethernet interface, also specify the port number associated with the interface. If you specify a VRF or VE, also specify the VRF name or VE number respectively.

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.

To clear IPv6 routes associated with the prefix 2000:7838::/32, use the **clear ipv6 route** command.

```
device# clear ipv6 route 2000:7838::/32
```

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. If you specify a VRF parameter, specify the VRF name.

Clearing IPv6 traffic statistics

To clear all IPv6 traffic statistics (reset all fields to zero), use the **clear ipv6 traffic** command.

```
device(config)# clear ipv6 traffic
```

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 tunnel gateway
- Next-hop network protocol type
- Ethernet interface, typically used for directly attached destination networks
- Virtual interface
- Null interface

NOTE
ICX 7150 devices do not support tunnels.

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.

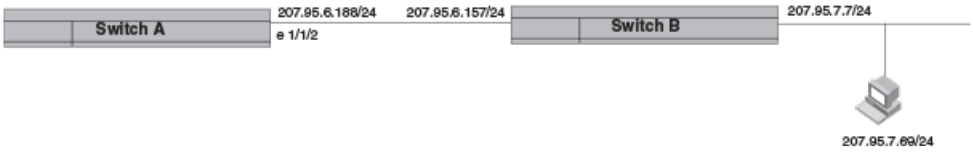
Static route states follow port states

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 15 Example of a 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 1/1/2, 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 1/1/2.

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 1/4/1
```

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 1/4/1 as the next hop.

The following example configures an IP static route to destination network addresses beginning with 10.0.0.0 through the next-hop interface 1/2/1.

```
device# configure terminal
device(config)# ip route 10.0.0.0/24 ethernet 1/2/1
```

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 a tunnel as next hop

NOTE
ICX 7150 devices do not support tunnels.

To configure an IP static route with a tunnel as the next hop, follow these steps.

1. Enter global configuration mode.

```
device# configure terminal
```

2. Configure the destination IP address, followed by the prefix length or address mask. On the same command line, enter the keyword **tunnel** followed by the tunnel ID.

```
device(config)# ip route 10.128.2.71 255.255.255.0 tunnel 4
```

The following example configures an IP static route with a destination address of 10.128.2.71, a network mask of 255.255.255.0, and a tunnel gateway (tunnel 4) as the next hop.

```
device# configure terminal
device(config)# ip route 10.128.2.71 255.255.255.0 tunnel 4
```

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.

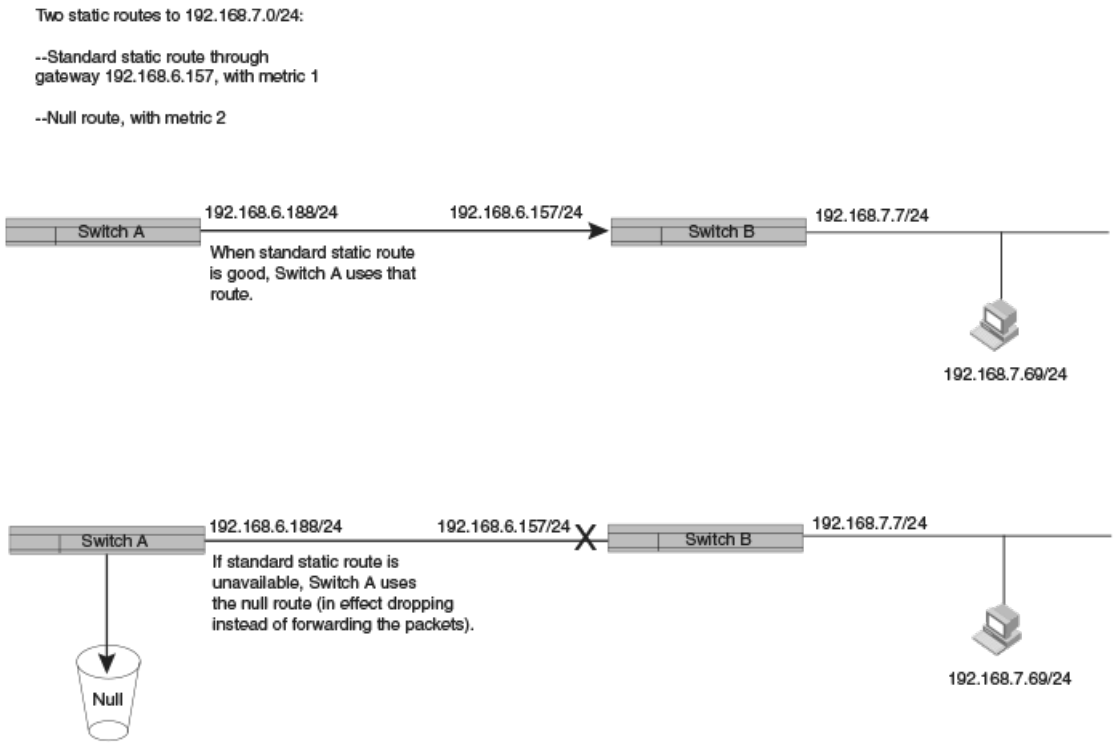
```
device# configure terminal
device(config)# ip route 10.0.0.0/24 10.1.1.1 tag 3
```

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 16 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
```

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
   - 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 rip
```

This example resolves static routes through RIP.

Creating an IP static route in a non-default VRF

You can configure an IP static route in a non-default VRF. If the VRF is not named, the default VRF is used.

The VRF configured must be a valid VRF.

NOTE
ICX 7150 devices do not support tunnels or VRFs.

To create an IP static route with a next hop in a non-default VRF, follow these steps.

1. Enter global configuration mode.

```
device# configure terminal
```

2. Enter the **ip route** command followed by the keyword **vrf** and the VRF name. On the same command line, enter the destination IP address, followed by the prefix-length or the address mask and then the IP address of the next-hop.

```
device(config)# ip route vrf blue 10.0.0.0/24 10.1.1.1
```

This example configures an IP static route through the non-default VRF "blue" with the next-hop address 10.1.1.1.

The following example configures a static route with a destination address 56.1.5.0/24. The route is configured in the non-default VRF "red" and uses tunnel 5 as the next-hop gateway.

```
device# configure terminal
device(config)# ip route vrf red  56.1.5.0/24  tunnel 5
```

NOTE
When a tunnel is designated as the next-hop gateway for a non-default VRF destination, the tunnel must already exist before the static route can be created.

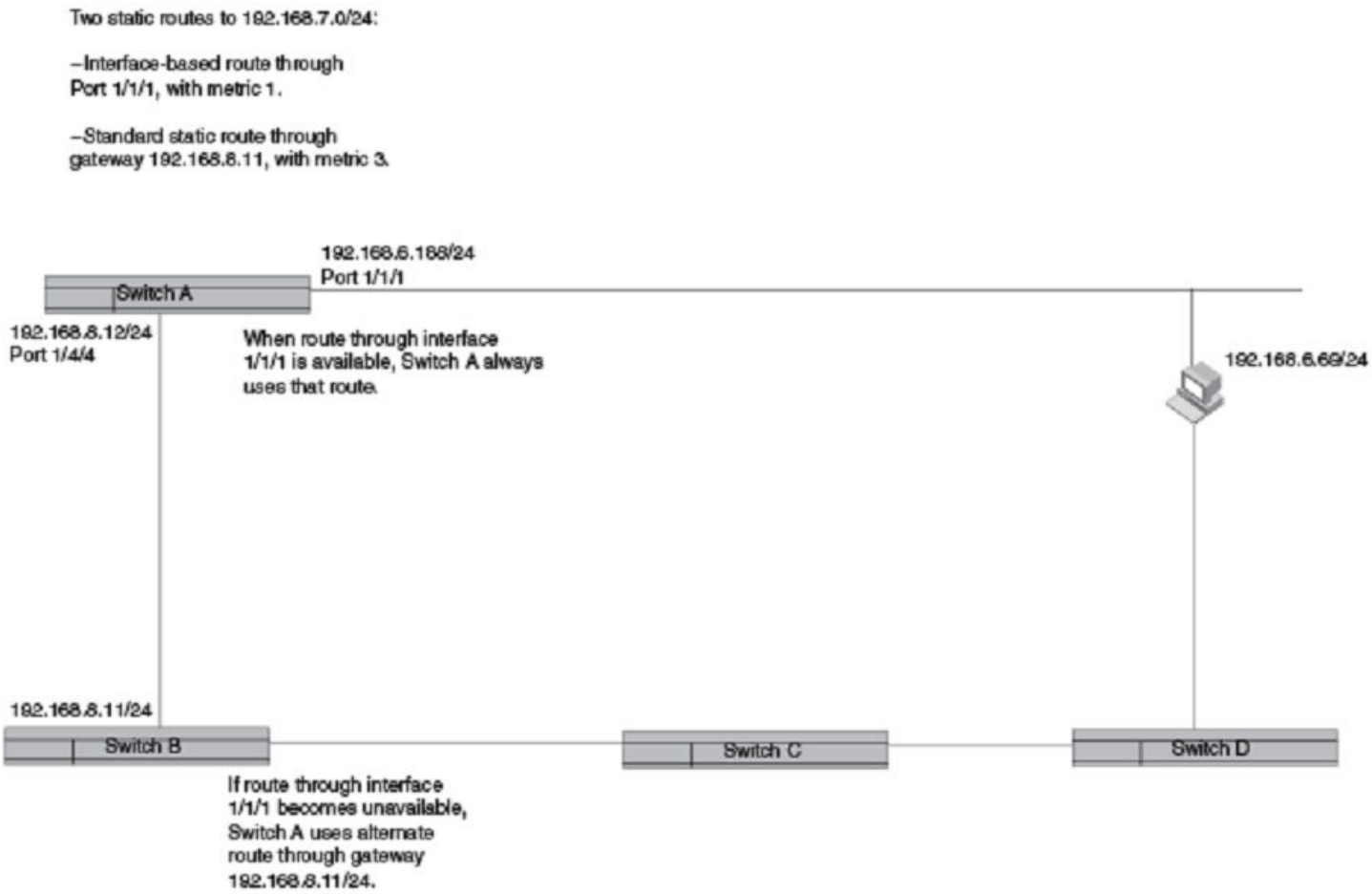
Configuring load sharing and redundancy

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 17 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.

Determining maximum static routes

You can modify the maximum number of static routes.

NOTE
Output examples in this section do not reflect system values for ICX devices. For example, the ICX 7150 default value for IPv4 static routes is 512, the minimum value is 64, and the maximum value is 512.

Perform these steps to check the maximum setting for static routes and to modify the value.

1. Verify the current maximum setting for the device.

```
device# show default values
```

   The maximum number of static IP routes the system can retain is listed under System Parameters in the ip-static-route row as shown in the following example.

```
device# show default values
sys log buffers:50 mac age time:300 sec telnet sessions:5
ip arp age:10 min bootp relay max hops:4 ip ttl:64 hops
ip addr per intf:24
:
:
System Parameters    Default Maximum Current Configured
ip-arp               4000    64000   4000    4000
ip-static-arp        512     6000    512     512
pim-mcache           1024    4096    1024    1024
:
:
ip-route             12000   15168   12000   12000
ip-static-route      64      2048    64      64
:
:
ip-vrf               16      16      16      16
ip-route-default-vrf 12000   15168   12000   12000
ip6-route            5120    5120    5120    5120
ip6-route-default-vr 5120    5120    5120    5120
ip6-route-vrf        100     5120    100     100
device(config)#
```

2. Enter global configuration mode.

```
device# configure terminal
```

3. Specify a new maximum for IP static routes.

```
device(config)# system-max ip-static-route 4096
```

4. Save the change and reload the device.

```
device(config)# write memory
device(config)# reload
```

5. Verify the change with the **show default values** command.

The following example changes the system max value for IP static routes from 2048 to 4096.

```
device# configure terminal
device(config)#
device(config)# show default values
sys log buffers:50 mac age time:300 sec telnet sessions:5
ip arp age:10 min bootp relay max hops:4 ip ttl:64 hops
ip addr per intf:24
:
:
System Parameters    Default Maximum Current Configured
ip-arp               4000    64000   4000    4000
ip-static-arp        512     6000    512     512
pim-mcache           1024    4096    1024    1024
:
:
ip-route             12000   15168   12000   12000
ip-static-route      64      2048    64      64
:
:
ip-vrf               16      16      16      16
ip-route-default-vrf 12000   15168   12000   12000
ip6-route            5120    5120    5120    5120
ip6-route-default-vr 5120    5120    5120    5120
ip6-route-vrf        100     5120    100     100
device(config)#

device(config)# system-max ip-static-route 4096

device(config)# write memory
device(config)# reload

device(config)# show default values
sys log buffers:50 mac age time:300 sec telnet sessions:5
ip arp age:10 min bootp relay max hops:4 ip ttl:64 hops
ip addr per intf:24
:
:
System Parameters    Default Maximum Current Configured
ip-arp               4000    64000   4000    4000
ip-static-arp        512     6000    512     512
pim-mcache           1024    4096    1024    1024
:
:
ip-route             12000   15168   12000   12000
ip-static-route      64      4096    64      64
:
:
ip-vrf               16      16      16      16
ip-route-default-vrf 12000   15168   12000   12000
ip6-route            5120    5120    5120    5120
ip6-route-default-vr 5120    5120    5120    5120
ip6-route-vrf        100     5120    100     100
device(config)#
```

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 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 a distance metric of 254 (beneath the threshold of 255, which would be unreachable) and a metric of 1. The route has been up for over 8 days.

The second route in this example is a directly connected route to all destinations beginning with 10.25.224.0. It has no extra costs associated with it and has been up for 6 hours and 39 minutes. The second route is the preferred route because, unlike the first route, it has no additional cost associated with it.

```
device# show ip route
Total number of IP routes: 2
Type Codes - B:BGP D:Connected O:OSPF R:RIP S:Static; Cost - Dist/Metric
BGP Codes - i:iBGP e:eBGP
OSPF Codes - i:Inter Area 1:External Type 1 2:External Type 2
        Destination       Gateway          Port          Cost   Type Uptime
1       0.0.0.0/0         10.25.224.1      e mgmt1       254/1  S    8d6h
2       10.25.224.0/24    DIRECT           e mgmt1       0/0    D    6h39
```

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

NOTE
ICX 7150 devices do not support tunnels.

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 18 Example of a 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 1/1/2, 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 1/1/2.

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
```

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 a non-default VRF.

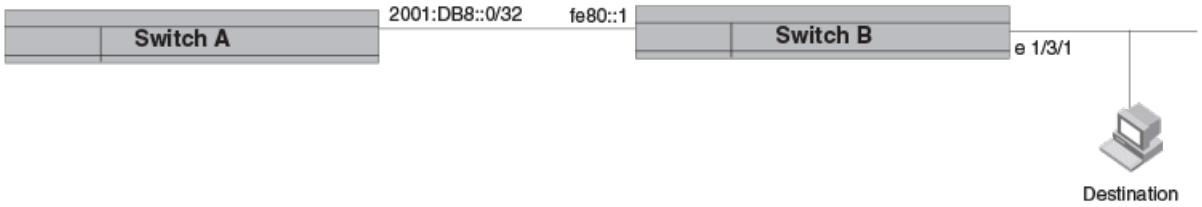
```
device# configure terminal
device(config)# vrf corporate
device(config-vrf-corporate)# rd 20:10
device(config-vrf-corporate)# address-family ipv6
device(config-vrf-corporate-ipv6)# no ipv6 route 2002::/64 ethernet 1/1/1
```

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 19 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 1/3/1 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 1/3/1.

```
device# configure terminal
device(config)# ipv6 route 2001:DB8::0/32 ethernet 1/3/1 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.

```
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.

NOTE
ICX 7150 devices do not support tunnels.

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 **tunnel** and the tunnel ID.

```
device(config)# ipv6 route 2001:DB8::0/32 tunnel 1
```

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 tunnel 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
ICX 7150 devices do not support VRFs.

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 immediately by the keyword **vrf** and the name of the VRF that contains the next-hop router for the route. On the same command line, enter the destination IPv6 address, including the prefix length, and the IPv6 address of the next hop.

```
device(config)# ipv6 route vrf partners 2001:DB8::0/32 2001:DB8:0:ee44::1
```

This example creates an IPv6 static route for the destination network addresses with the prefix 2001:DB8::0/32 through the next-hop VRF named "partners" with the global 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
```

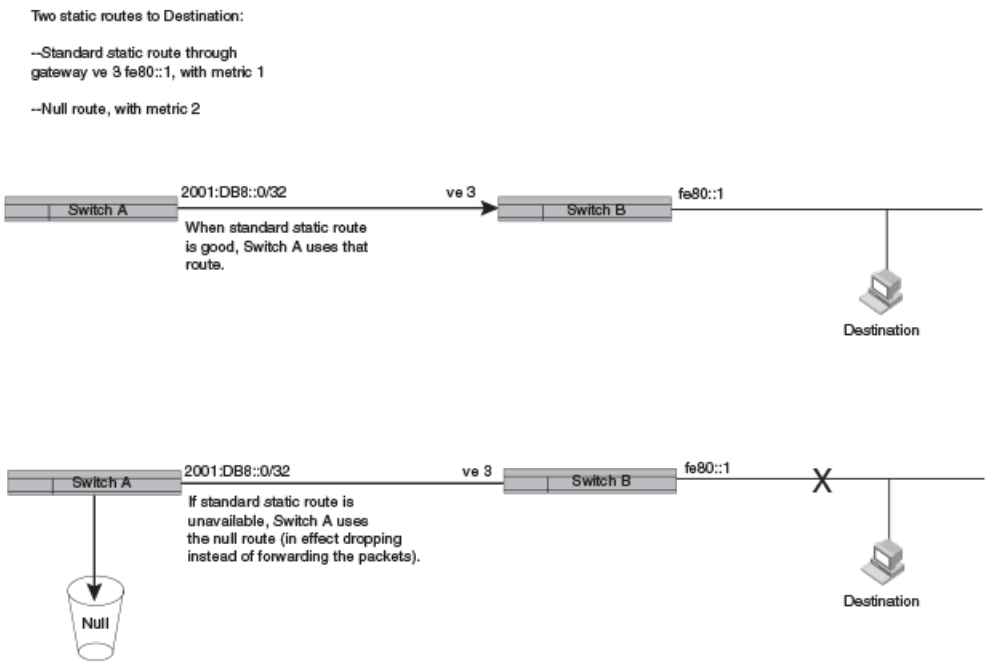
Configuring a null route

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 20 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 a static route using other static routes

You can use other static routes to resolve a static route through recursive lookup in local routing tables, up to 10 hops away.

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. You can enter a value from 1 through 10. If no value is specified, the level of recursion is 3.

```
device(config)# ipv6 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)# ipv6 route next-hop-recursion 5
```

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, or RIP protocol.

NOTE
Connected routes are always used to resolve static routes.

1. Enter global configuration mode.

```
device# configure terminal
```

2. (Optional) Designate the non-default VRF of the destination.
3. Designate next-hop resolution through BGP, RIP, or OSPF protocol.

```
device(config)# ipv6 route next-hop vrf blue bgp
```

This example designates that the VRF named "blue" is to be used and that static routes are to be resolved through BGP.

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
```

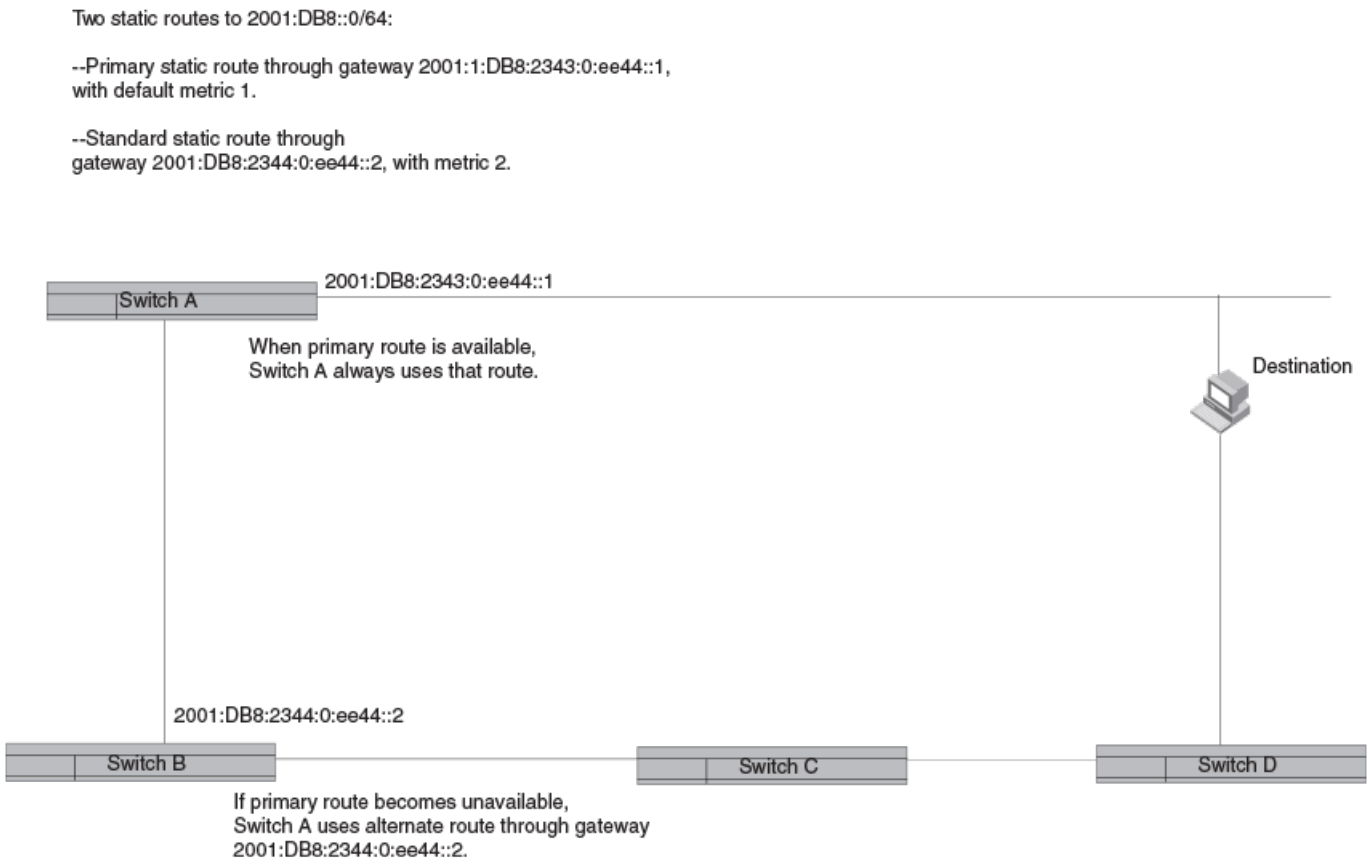
Configuring load sharing and redundancy

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 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)# 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 Ruckus 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 Ruckus 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 1/1/1
   ```

   This example configures an IPv6 static route to IP address 2002::/64 destinations via next-hop interface 1/1/1.

The following example creates an IPv6 multicast route with the RD 20:10 to 2002::/64 IP address via next-hop Ethernet interface 1/1/1. 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 1/1/1 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 the IPv6 route table information, enter the **show ipv6 route** command.

```
device# show ipv6 route
IPv6 Routing Table - 7 entries:
Type Codes: C - Connected, S - Static, R - RIP, O - OSPF, B - BGP
OSPF Sub Type Codes: O - Intra, Oi - Inter, O1 - Type1 external, O2 - Type2 external
Type     IPv6 Prefix         Next Hop Router          Interface           Dis/Metric
C        2000:4::/64         ::                       ethe 1/3/2          0/0
S        2001:DB8::/16       ::                       tunnel 6            1/1
S        2001:DB8:1234::/32  ::                       tunnel 6            1/1
C        2001:DB8:46a::/64   ::                       ethe 1/3/2          0/0
C        2001:DB8::1/128     ::                       loopback 2          0/0
O        2001:DB8::2/128     fe80::2e0:52ff:fe91:bb37 ethe 1/3/2          110/1
C        2001:DB8::/64       ::                       tunnel 2            0/0
```

   As shown in the example, connected, static, RIP, OSPF, and BGP routes are listed, along with the destination address, the next hop router, the interface used toward the destination, and the administrative distance and cost for each route.

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 distance vector used to define cost is often equivalent to the number of hops between the Ruckus device and the destination network. A hop is another router through which packets must travel to reach the destination.

A Ruckus device can receive multiple paths to a destination. The software evaluates the paths, selects the best path, and saves the path in the IP routing table as the route to the destination. Typically, the best path is the path with the fewest hops. If a RIP update is received from another router that contains a path with fewer hops than the path stored in the Ruckus device route table, the older route is replaced with the newer one. The new path is included in the updates sent to other RIP routers, including Ruckus devices.

RIP routers, including Ruckus devices, can modify a route cost, generally by adding to it, to bias the selection of a route for a given destination. In this case, a route may have the same number of hops as other routes, but because it has a higher administrative cost, it is less likely to be used.

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.

Ruckus devices support the following RIP versions:

- Version 1 (v1)
- Version 2 (v2, the default)
- V1 compatible with v2

NOTE
Some Ruckus devices support IPv6 RIP, also known as RIPng. Refer to the chapter "RIPng" for more information.

Overview of RIP route learning and advertising parameters

By default, when RIP is enabled, a Ruckus 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.
- Learning and advertising of RIP default routes - The device can learn and advertise RIP default routes.
- Learning of standard RIP routes - By default, the 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.

The update interval for route advertisements

The update interval specifies how often the device sends route advertisements to its RIP neighbors. The default is 30 seconds.

RIP default routes

The device does not learn default RIP from its neighbors unless you enable learning of RIP default routes. You can enable or disable learning and advertising of default routes on a global or individual interface basis.

RIP neighbor filters

You can create neighbor filters and apply them globally to specify the neighbor routers from which the device can learn RIP routes. RIP neighbor filters are individual commands to permit or deny specified routes. A filter can permit any route, deny any route, or permit or deny a specific IP address.

Each filter includes a filter number as part of the command. The filters are performed in numeric sequence, and it helps to enter filter commands in ascending order.

The following example configures the device so that it does not learn any RIP routes from any RIP neighbors.

```
device# configure terminal
device(config)# router rip
device(config-rip-router)# neighbor 1 deny any
```

In the example, the filter number is 1, which means it is always acted on first. If you want to allow specific routes and deny all others, give the previous command a higher filter number and insert filters with lower numbers that permit routes you want the device to learn. The following example permits learning neighbor routes on one IP address (filter 3) and denies all other routes (filter 32).

```
device# configure terminal
device(config)# router rip
device(config-rip-router)# neighbor 3 permit 10.71.10.102
device(config-rip-router)# neighbor 32 deny any
```

If, instead, you want to allow learning on all routes except for one specific route, you must also include a neighbor filter to permit any route. Be sure you add the filter to permit learning from any neighbors as the filter with the highest number. Otherwise, the software will match on the "permit any" filter and never act on any later filter that denies a specific neighbor. Consequently, routes will be learned from the neighbor that was supposed to be filtered out. The following example blocks route learning from one neighbor (filter 4) and explicitly permits learning routes from all other neighbors (filter 24).

```
device# configure terminal
device(config)# router rip
device(config-rip-router)# neighbor 4 deny 10.70.12.103
device(config-rip-router)# neighbor 24 permit any
```

Redistribution of routes into RIP

When you redistribute routes from another protocol into RIP, the device can use RIP to advertise the routes to its RIP neighbors.

Redistribution filters

You can configure filters to permit or deny redistribution for a route based on its origin (for example, OSPF or BGP4), the destination network address, or the route's metric. You can also configure a filter to set the route metric based on these criteria.

The default redistribution metric

When the 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 set the metric to a value from 1 through 15.

Using prefix lists and route maps to filter RIP routes

You can configure prefix lists to permit or deny specific routes and then apply them globally or to individual interfaces. When you apply a prefix-list, you must specify whether the list 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 prefix lists you can apply later, 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 list4  deny 10.53.7.1 255.255.255.0
```

In this example, list1 is configured to permit an IP address and mask, and list4 is configured to deny another IP address and mask.

Using route maps in RIP

Use route maps to define how you want to permit or deny redistribution through an individual interface. A route map is a named set of match conditions that the device can use to modify route attributes or to control redistribution of certain routes into other protocols.

A route map consists of a sequence of up to 50 instances. The 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 device stops evaluating the route against the remaining route map instances.

Route maps contain match statements. In RIP, match statements are based on prefix lists and access control lists. A route map can be applied to learned routes (in) our advertised routes (out). Each route is checked against match statements. When a match is found, the route may be permitted, denied, or modified, depending on the contents of the route map.

The following rules apply to route maps:

- If there is no match statement at all in the route map, the route is considered to be a match.
- If a match statement contains a permit action, a matching route is permitted, and no additional route map instances are checked for that route.
- If a match statement contains a deny action, a matching route is denied, and no additional route map instances are checked for that route.
- 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".
- 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.
- For a virtual routing interface, the default redistribution action is permit, even after you configure and apply redistribution filters. 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.

The following example shows the configuration of a route map that permits routes to two networks and denies routes to one network.

In the following example, an access-list (ACL) named 21 is created. The first ACL entry denies IP addresses that match a particular network mask. The second ACL entry permits any other IP addresses. A route map is configured with the name routemap1 to permit routes that are defined in routemap configuration sub-mode, and a sequence number of 21 is assigned. In the routemap, a match statement is defined to match addresses filtered using ACL 21. Any routes that match the IP address and mask of 10.16.0.0 0.0.255.255 will be denied. All other routers are permitted.

```
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
```

NOTE
You can configure a route map to match on all RIP routes as shown in the following match statement. This example allows any RIP route.

```
device(config-routemap test)# match protocol rip permit any
```

Enabling RIP and configuring global parameters

RIP is disabled by default. You must enable RIP globally and on individual interfaces on which you want to advertise RIP routes. You can enable RIP on physical interfaces as well as virtual routing interfaces.

Once enabled, RIP operates with parameters at default settings. Default settings can be modified at the global level. Interface-level settings can be modified to override global settings on individual interfaces.

1. Enter global configuration mode.

   ```
   device# configure terminal
   ```

2. To enable RIP globally, enter the **router rip** command.

   ```
   device(config)# router rip
   device(config-rip-router)#
   ```

   This example places the device in RIP router configuration mode.

3. (Optional) Change the route loop prevention method.

   ```
   device(config-rip-router)# poison-reverse
   ```

   This example disables split horizon (the default) and enables poison-reverse route loop prevention. To re-enable split horizon, use the **no** form of the command.

4. (Optional) Configure the 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
   ```

5. (Optional) Modify the administrative distance.

```
device(config-rip-router)# distance 140
```

This example increases the administrative distance to 140.

NOTE
The default distance is 120.

6. (Optional) Modify RIP timer settings. You must enter a value for each of the timers, even for those you are not changing.

```
device(config-rip-router)# timer 15 115 115 120
```

This example sets the update timer to 15 seconds, the timeout timer to 115 seconds, the hold-down timer to 115 seconds, and leaves the garbage collection timer at its default setting of 120 seconds.

NOTE
To reset the timers to their defaults, enter the **no timer** command with the current value of all timer parameters.

NOTE
If you only want to modify the value of the update timer, use the **update-time** command.

7. (Optional) Enable learning of default RIP routes from RIP neighbors.

```
device(config-rip-router)# learn-default
```

8. (Optional) Apply global filters for learning and advertising specific routes from neighbors.

```
device(config-rip-router)# neighbor 4 deny 10.70.12.103
device(config-rip-router)# neighbor 24 permit any
```

This example denies route learning from IP address 10.70.12.103 and permits learning routes from all other neighbors.

9. (Optional) Modify the default redistribution metric.

```
device(config-rip-router)# default-metric 10
```

10. **NOTE**
    Do not enable redistribution until you configure other parameters related to redistribution. For example, set the default redistribution metric and configure any prefix lists or route-maps to be used beforehand.

    (Optional) Enable redistribution of routes from other protocols into RIP using available parameters:

    - **connected** - applies redistribution to connected routes
    - **bgp** - applies redistribution to BGP4 routes
    - **ospf** - applies redistribution to OSPF routes
    - **static** - applies redistribution to IP static routes
    - **metric** *value* - sets the RIP metric value from 1 through 15 for any routes imported into RIP
    - **route-map** *name* - indicates the name of a pre-configured route map to be used in filtering specified routes

    ```
    device(config-rip-router)# redistribute ospf metric 3
    ```

    This example redistributes OSPF into RIP and sets the metric for OSPF routes to 3.

    ```
    device(config-rip-router)# redistribute ospf route-map routemap1
    ```

    This example applies a previously configured route map (routemap1) to OSPF route redistribution.

    NOTE
    To stop redistributing routes into RIP, use the **no** form of the redistribute command, including the full command syntax of the active command.

The following example enables RIP, increases the administrative distance, modifies timer values for all but the garbage-collection timer, sets the metric for all distributed routes to 10, denies route learning from IP address 10.70.12.203, and applies a pre-configured route map to redistributed OSPF routes.

```
device# configure terminal
device(config)# router rip
device(config-rip-router)# distance 140
device(config-rip-router)# timer 15 115 115 120
device(config-rip-router)# learn-default
device(config-rip-router)# default-metric 10
device(config-rip-router)# neighbor 4 deny 10.70.12.103
device(config-rip-router)# neighbor 24 permit any
device(config-rip-router)# redistribute ospf route-map routemap1
```

Configuring RIP interfaces

1. Configure the Ethernet interface link.

   ```
   device# configure terminal
   device(config)# interface ethernet 1/1/1
   ```

2. To enable RIP on the interface, enter the **ip rip** command and, if necessary, specify the version of RIP.

   NOTE
   RIP version 2 is the default.

   ```
   device(config-if-e1000-1/1/1)# ip rip v1-only
   ```

   This example enables RIP version 1 on port 1/1/1.

3. (Optional) Change the route loop prevention method used on the interface.

```
device(config-if-e10000-1/1/1)# no ip rip poison-reverse
```

   This example disables poison-reverse on the interface and enables split horizon loop prevention (the default).

4. (Optional) To increase the metric for routes learned on the interface, enter the **ip rip metric-offset** command followed by the desired value and the keyword **in**.

```
device(config-if-e1000-1/1/1)# ip rip metric-offset 5 in
```

   This example configures the port to add 5 to the cost of each RIP route it learns.

   NOTE
   The metric-offset can be any value from 1 through 16. A value of 16 prevents a learned route from being used.

5. (Optional) To increase the metric for RIP routes the interface advertises to neighbors, enter the **ip rip metric-offset** command followed by the desired value and the keyword **out**.

```
device(config-if-e1000-1/1/1)# ip rip metric-offset 5 out
```

   This example configures the port to add 5 to the cost of each route it advertises, using the keyword **out**.

   NOTE
   The metric-offset can be any value from 1 through 16. A value of 16 prevents an advertised route from being used.

6. (Optional) Enable or disable learning RIP default routes on the interface.

```
device(config-if-e10000-1/1/1)# ip rip learn-default
```

   This example configures Port 1/1/1 to learn default RIP routes.

7. (Optional) Apply filters to learn and advertise specific routes.

```
device(config-if-e1000-1/1/1)# ip rip prefix-list list2 in
device(config-if-e1000-1/1/1)# ip rip prefix-list list3 out
```

   This example applies a prefix list (list2) to learned routes and another prefix list (list3) to advertised routes.

The following example configures port 1/1/1 to use RIP version 1 with split horizon loop prevention. It increases the cost of learned and advertised routes by 5, enables learning default RIP routes, and applies prefix lists to learned and advertised routes.

```
device(config)# interface ethernet 1/1/1
device(config-if-e1000-1/1/1)# ip rip v1-only
device(config-if-e10000-1/1/1)# no ip rip poison-reverse
device(config-if-e1000-1/1/1)# ip rip metric-offset 5 in
device(config-if-e1000-1/1/1)# ip rip metric-offset 5 out
device(config-if-e10000-1/1/1)# ip rip learn-default
device(config-if-e1000-1/1/1)# ip rip prefix-list list2 in
device(config-if-e1000-1/1/1)# ip rip prefix-list list3 out
```

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  29, Next Update 27
      Need trigger update  0, Next trigger broadcast 1
      Minimum update  interval 25, Max update Offset 5
      Split horizon is on; poison reverse is off
      Import metric 1
      Prefix List, Inbound : block_223
      Prefix List, Outbound : block_223
      Route-map, Inbound : Not set
      Route-map, Outbound : Not set
      Redistribute: CONNECTED  Metric : 0  Routemap : Not Set

  No Neighbors are configured in RIP Neighbor Filter Table
```

To display RIP filters for a specific interface, enter the following command.

```
device# show ip rip interface ethernet 1/1/1
Interface e 1/1/1
RIP Mode : Version2 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
Route-map, Outbound : Not set
RIP Sent/Receive packet statistics:
    Sent : Request 2 Response 34047
    Received : Total 123473 Request 1 Response 123472 UnRecognised 0
RIP Error packet statistics:
    Rejected 0  Version 0 RespFormat 0 AddrFamily 0
    Metric 0 ReqFormat 0
```

To display RIP route information, enter the following command.

```
device# show ip rip route
RIP Routing Table - 474 entries:
1.1.1.1/32, from 169.254.30.1, e 1/1/23   (820)
          RIP, metric 4, tag 0, timers: aging 13
1.1.2.1/32, from 169.254.50.1, e 1/3/1   (482)
          RIP, metric 3, tag 0, timers: aging 42
1.1.6.1/32, from 169.254.100.1, ve 101    (413)
          RIP, metric 2, tag 0, timers: aging 42
169.254.40.0/24, from 192.168.1.2, e 1/1/1   (1894)
          RIP, metric 3, tag 0, timers: aging 14
169.254.50.0/24, from 192.168.1.2, e 1/1/1   (1895)
          RIP, metric 4, tag 0, timers: aging 14
169.254.100.0/24, from 192.168.1.2, e 1/1/1   (2040)
          RIP, metric 2, tag 0, timers: aging 14
169.254.101.0/30, from 192.168.1.2, e 1/1/1   (2105)
223.229.32.0/31, from 169.254.50.1, e 1/3/1   (818)
          RIP, metric 2, tag 0, timers: aging 21
```

To display current running configuration for interface 1/1/1, enter the following command.

```
device# show running-config interface ethernet 1/1/1
interface ethernet 1/1/1
 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
!
```

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.

RIPng configuration overview

To configure RIPng, you must enable RIPng globally on the device and on individual device interfaces.

The following RIPng configuration tasks are optional:

- Change the default settings of RIPng timers
- Configure poison reverse parameters
- Configure how the device learns and advertises routes
- Configure which routes are redistributed into RIPng from other sources
- Configure how the device distributes routes through RIPng

RIPng timers

You can adjust timers for RIPng. Before doing so, keep the following caveats in mind:

- If you adjust RIPng timers, Ruckus 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.
- Ruckus recommends setting the timeout timer value to at least three times the value of the update timer.
- Ruckus recommends a shorter hold-down timer interval, because a longer interval can cause delays in RIPng convergence.

RIPng route loop prevention

By default, split horizon is enabled as the route loop prevention method. Split horizon prevents advertising a route on the same interface over which the route is learned. 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. Enabling poison reverse on the RIPng device disables split-horizon and vice versa.

By default, if a RIPng interface goes down, the device does not send a triggered update for the interface's IPv6 networks. You can use the **poison-local-routes** command to configure a RIPng 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.

RIPng route learning and advertisement

You can configure the following learning and advertising parameters:

- **Learning and advertising of RIPng default routes** - By default, the device does not learn IPv6 default routes (::/0). You can originate default routes into RIPng, which causes individual device interfaces to include the default routes in their updates. When configuring the origination of the default routes, you can include only the default routes, or you can include default routes and all other routes.
- **Advertising of 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 instead of advertising the original route. If a route's prefix length matches the value specified in the **ipv6 rip summary-address** command, RIPng advertises the prefix specified in the command instead of the original route.
- **Metric of routes learned and advertised on a device interface** - You can change the metric offset an individual interface adds to a route it learns or advertises.

Route redistribution into RIPng

You can configure the device to redistribute routes from the following sources into RIPng:

- IPv6 static routes
- Directly connected IPv6 networks
- BGP4+
- OSPFv3

When you redistribute a route from BGP4+ or OSPFv3 into RIPng, the device can use RIPng to advertise the route to its RIPng neighbors.

When configuring the 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.

Applying filters to RIPng route redistribution

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 create and apply a prefix list that permits only routes with the prefix 2001:db8::/32 in RIPng routing updates sent to RIPng neighbor routers, 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 create and apply a prefix list to deny prefix lengths greater than 64 bits in routes that have the prefix 2001:db8::/64 and allow all other routes received from RIPng neighbor routers, 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
```

Enabling RIPng and configuring global parameters

By default, RIPng is disabled. To enable RIPng, you must enable it globally on the device and also on individual device interfaces.

Before configuring the device to run RIPng, you must do the following:

- Enable 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 enabling IPv6 with the **ipv6 enable** command on that interface.

1. Enter global configuration mode.

   ```
   device# configure terminal
   ```

2. Enter RIP router configuration mode.

   ```
   device(config)# router rip
   device(config-rip-router)#
   ```

3. Enter the following command to enable RIPng.

   ```
   device(config-rip-router)# ipv6 router rip
   device(config-ripng-router)#
   ```

   This example enables RIPng and places the device in RIPng router configuration mode.

   NOTE
   To disable RIPng globally, use the **no** form of this command.

4. (Optional) Change the route loop prevention method.

   ```
   device(config-ripng-router)# poison-reverse
   ```

   This example disables split horizon and enables poison-reverse route loop prevention. To re-enable split horizon, use the **no** form of the command.

5. (Optional) Configure the device to avoid routing loops by advertising local RIPng routes with a cost of 16 ("infinite" or "unreachable") when these routes go down.

   ```
   device(config-ripng-router)# poison-local-routes
   ```

6. (Optional) Modify the administrative distance.

   ```
   device(config-ripng-router)# distance 140
   ```

   This example increases the administrative distance to 140.

7. (Optional) Modify RIPng timer settings. Set the four timers in this order: update timer, timeout timer, hold-down timer, and garbage-collection timer. You must enter a value for each timer, even when you are not changing the value of an individual timer.

```
device(config)# ipv6 router rip
device(config-ripng-router)# timers 45 135 10 20
```

   This 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.

   To return to the default values of the RIPng timers, use the **no** form of the **timers** command.

8. **NOTE**
   Do not enable redistribution until you configure the other redistribution parameters.

   (Optional) Enable redistribution of routes from other protocols into RIPng using available parameters:

   - **connected** - applies redistribution to connected types
   - **bgp** - applies redistribution to BGP4 routes
   - **ospf** - applies redistribution to OSPF routes
   - **static** - applies redistribution to IP static routes

```
device# configure terminal
device(config)# router rip
device(config-rip-router)# ipv6 router rip
device(config-ripng-router)# redistribute ospf metric 3
```

   This example redistributes OSPF into RIPng and sets the metric for OSPF routes to 3.

   NOTE
   To stop redistributing routes into RIPng, use the **no** form of the redistribute command, including the full command syntax of the active command.

9. (Optional) Apply a pre-configured prefix list to control route distribution through RIPng.

```
device(config-ripng-router)# distribute-list prefix-list routesfor2001 out
```

   This example applies a pre-configured prefix list (2001routes) to routes advertised.

The following example enables RIPng globally, sets poison-reverse as the loop prevention method, enables blocking of local routes for interfaces that are down, redistributes OSPF routes into RIPng with an added cost of 3, and applies a pre-configured prefix list (routesfor2001) to advertised routes.

```
device# configure terminal
device(config)# router rip
device(config-rip-router)# ipv6 router rip
device(config-ripng-router)# poison-reverse
device(config-ripng-router)# poison-local-routes
device(config-ripng-router)# redistribute ospf metric 3
device(config-ripng-router)# distribute-list prefix-list routesfor2001 out
```

Enabling and configuring RIPng interfaces

After enabling RIPng globally, you must enable it on individual device interfaces.

1. You can enable RIPng on physical as well as virtual routing interfaces. For example, to enable RIPng on Ethernet interface 1/3/1, enter the following commands.

```
device# configure terminal
device(config)# interface ethernet 1/3/1
device(config-if-e1000-1/3/1)# ipv6 rip enable
```

   NOTE
   To disable RIPng on an individual device interface, use the **no** form of this command.

2. (Optional) Enable default route learning.

```
device(config-if-e1000-1/3/1)# ipv6 rip default-information originate
```

   This example originates IPv6 default routes and includes all other routes in updates sent from Ethernet interface 1/3/1.

3. (Optional) Configure the interface so that it advertises IPv6 address summaries instead of the original route.

```
device(config-if-e1000-1/3/1)# ipv6 address 2001:db8:0:adff:8935:e838:78:
e0ff /64
device(config-if-e1000-1/3/1)# ipv6 rip summary-address 2001:db8::/36
```

   This example advertises the summarized prefix 2001:db8::/36 instead of the original IPv6 address from Ethernet interface 1/3/1.

4. (Optional) Change the metric offset that the interface adds for learned or advertised routes.

```
device(config-if-e1000-1/3/1)# ipv6 rip metric-offset 2
device(config-if-e1000-1/3/1)# ipv6 rip metric-offset out 3
```

   In this example, if Ethernet interface 1/3/1 learns about an incoming route, it will increase the incoming metric by two. if the interface 1/3/1 advertises an outgoing route, it will increase the metric offset by 3.

5. (Optional) Apply a prefix-list to the interface for learned and/or advertised routes.

```
device(config-if-e1000-1/3/1)# ipv6 rip prefix-list test1 in
device(config-if-e1000-1/3/1)# ipv6 rip prefix-list test2 out
```

   In this example, the prefix-list test1 is applied to learned routes, and the prefix-list test2 is applied to advertised routes.

The following example enables RIPng on port 1/3/1. It enables learning of default RIPng routes and all other routes. It modifies the interface to send route summaries. It changes the metric for learned routes to 2 and the metric for advertised routes to 3. Finally, it applies a pre-configured prefix list to filter incoming (learned) routes.

```
device# configure terminal
device(config)# interface ethernet 1/3/1
device(config-if-e1000-1/3/1)# ipv6 rip enable
device(config-if-e1000-1/3/1)# ipv6 rip default-information originate
device(config-if-e1000-1/3/1)# ipv6 address 2001:db8:0:adff:8935:e838:78:e0ff/64
device(config-if-e1000-1/3/1)# ipv6 rip summary-address 2001:db8::/36
device(config-if-e1000-1/3/1)# ipv6 rip metric-offset 2
device(config-if-e1000-1/3/1)# ipv6 rip metric-offset out 3
device(config-if-e1000-1/3/1)# ipv6 rip prefix-list test1 in
```

Clearing RIPng routes from the 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
```

Displaying RIPng information

1. 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
```

2. 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:
ada::1:1:1:2/128, from fe80::224:38ff:fe8f:3000, e 1/3/4
          RIP, metric 2, tag 0, timers: aging 17
 2001:db8::/64, from fe80::224:38ff:fe8f:3000, e 1/3/4
          RIP, metric 3, tag 0, timers: aging 17
 bebe::1:1:1:4/128, from ::, null   (0)
          CONNECTED, metric 1, tag 0, timers: none
cccc::1:1:1:3/128, from fe80::768e:f8ff:fe94:2da, e 2/1/23
          RIP, metric 2, tag 0, timers: aging 50
```

OSPFv2

OSPFv2 overview

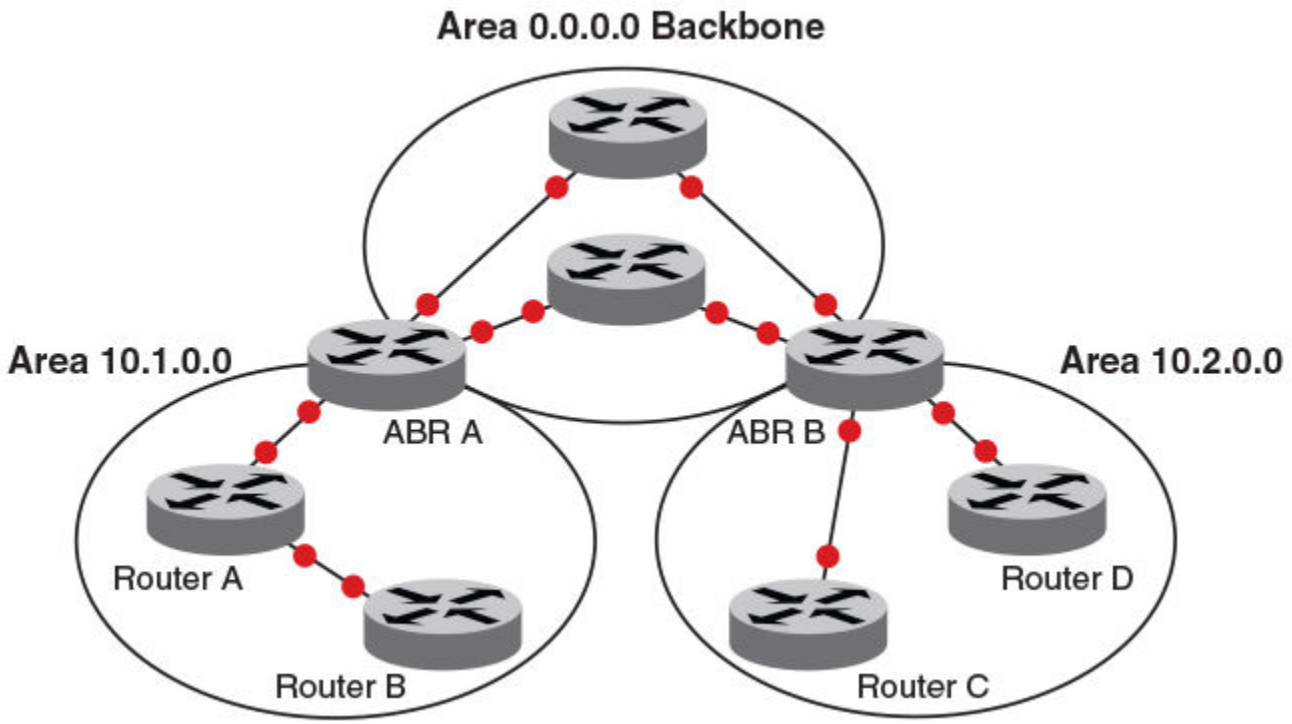
Open Shortest Path First Version 2 (OSPFv2) is a link-state routing protocol that uses link-state advertisements (LSAs) to update neighboring routers about a router's interfaces. Each router maintains an identical area-topology database to determine the shortest path to any neighboring router.

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 22 OSPF operating in a network



NOTE
For details of components and virtual links, refer to OSPFv2 components and roles on page 191 and Virtual links on page 201, 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.

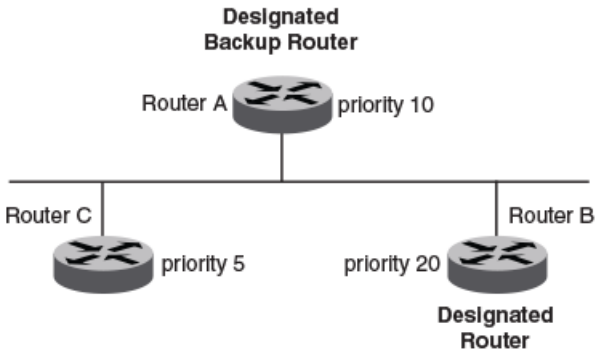
On broadcast networks such as LAN links, all routers on the LAN other than the DR and BDR form full adjacencies with the DR and BDR and pass LSAs only to them. The DR forwards updates received from one neighbor on the LAN to all other neighbors on that same LAN. One of the main functions of a DR is to 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 FastIron Command Reference.

FIGURE 23 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.

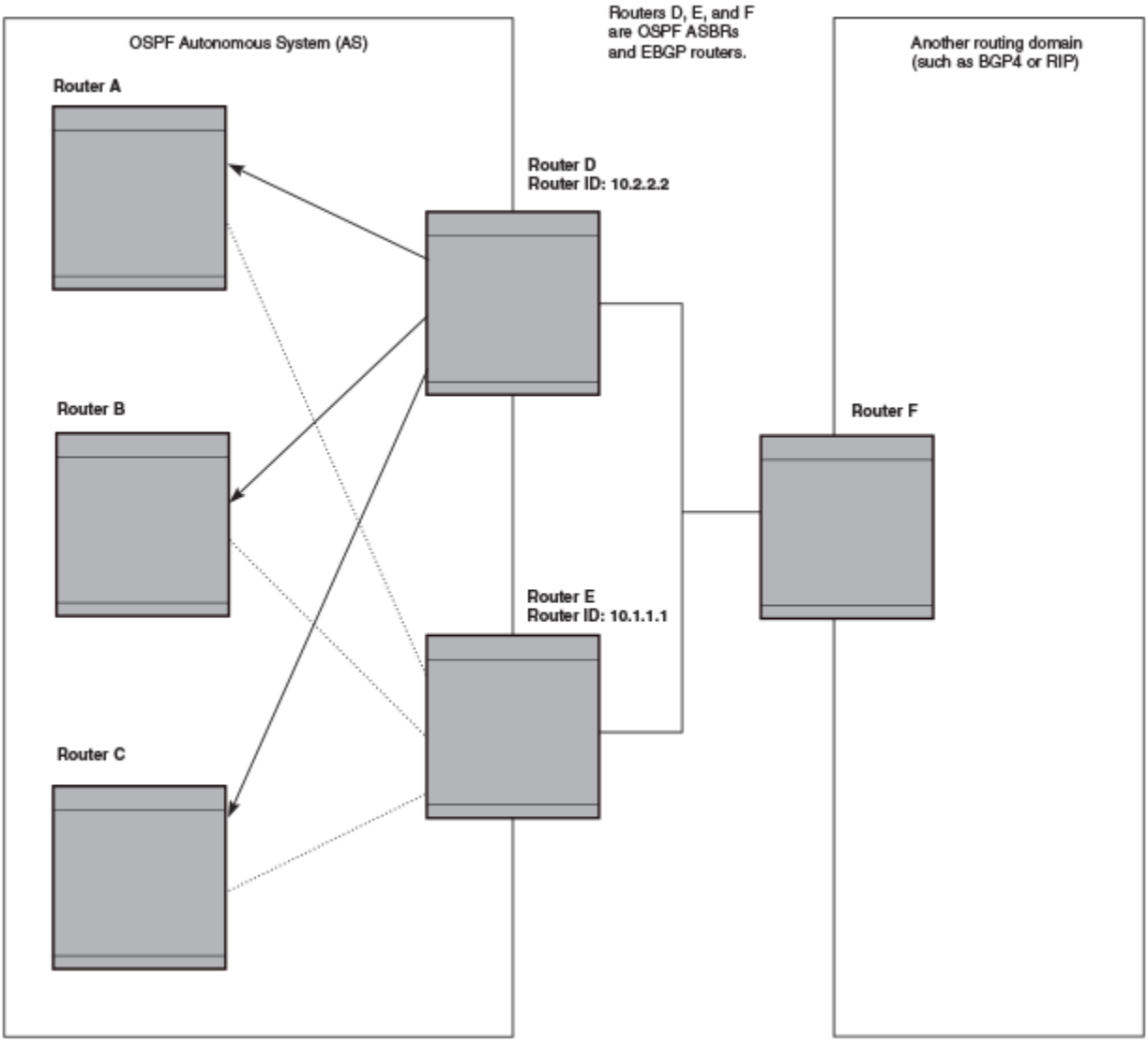
Reduction of equivalent AS external LSAs

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 24 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.

Maximum limit overload processing

Maximum limit overload processing automatically begins if a link-state database (LSDB) overflow condition occurs, bringing down the adjacency and clearing the LSDB database. When this LSDB overflow occurs, a syslog is automatically generated. OSPF remains in this condition for 600 seconds before resuming processing.

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 228.

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.

NOTE
For the ICX 7150, a maximum of 4 OSPF areas is supported for each OSPF instance.

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
```

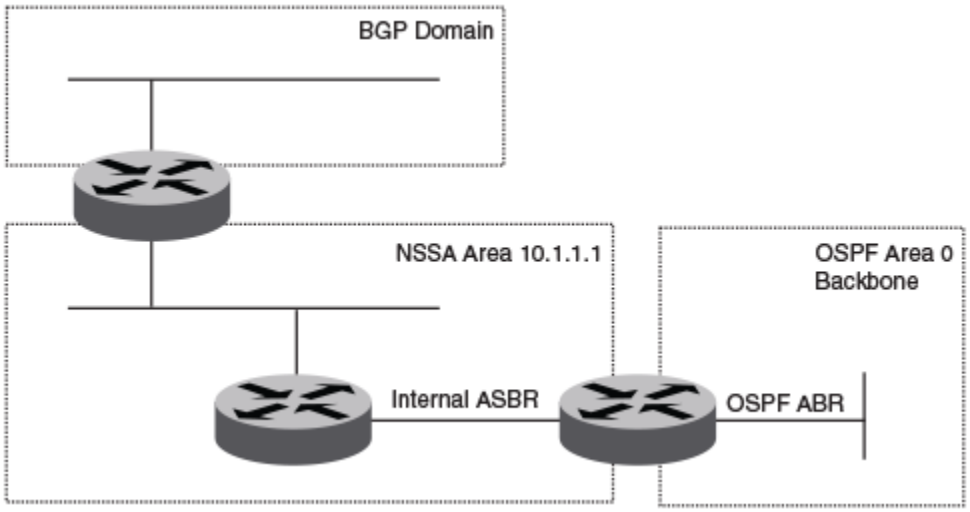
Not-so-stubby area (NSSA)

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 25 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 *Ruckus FastIron 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

Ruckus 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:

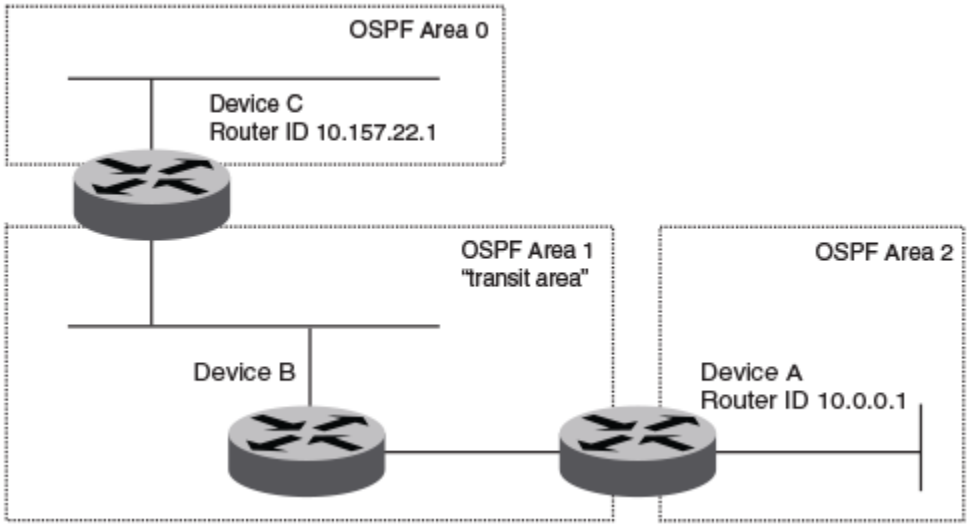
- The transit area ID represents the shared area of the two ABRs and serves as the connection point between the two routers. This number should match the area ID value.
- The neighbor router field is the router ID (IP address) of the router that is physically connected to the backbone, when assigned 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 26 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 Ruckus 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
```

Disabling the Opaque LSA Capability

The opaque link-state advertisement (LSA) capability is enabled by default and can be disabled. When the opaque LSA capability is disabled, the device does not accept opaque LSAs from a peer. Therefore, these LSAs are not added to the OSPF link state database and are not flooded to the opaque capable peers. When the default opaque capability is enabled for a device, the device does not originate any opaque LSAs from itself.

The following task disables the opaque LSA capability so that opaque LSAs are not added to the OSPF link state database and are not flooded to the opaque capable peers.

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 **opaque-capability** command with the **disable** parameter to disable the opaque LSA capability.

```
device(config-ospf-router)# opaque-capability disable
```

The following example disables the opaque LSA capability.

```
device# configure terminal
device(config)# router ospf
device(config-ospf-router)# opaque-capability disable
```

The following example re-enables the opaque LSA capability.

```
device# configure terminal
device(config)# router ospf
device(config-ospf-router)# opaque-capability enable
```

Support for OSPF RFC 2328 Appendix E

Ruckus 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.

The table below shows GR support for OSPFv2.

TABLE 17 Graceful restart support for OSPFv2

GR restarting router	GR helper	NSR (no neighbor support needed)
Yes	Yes	Yes

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.
- NSR does not support virtual links.
- 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.

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 10.0.0.2/32
device(config)# ip prefix-list test1 seq 10 permit 10.102.1.0/24
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 O:OSPF R:RIP S:Static; Cost - Dist/Metric
BGP  Codes - i:iBGP e:eBGP
OSPF Codes - i:Inter Area 1:External Type 1 2:External Type 2
        Destination        Gateway         Port          Cost          Type Uptime
1       10.0.0.2/32        10.1.1.2        ve 100        200/501       O    1h3m
2       10.102.1.0/24      10.1.1.2        ve 100        200/2         O    1h3m
3       10.102.6.0/24      10.1.1.2        ve 100        110/2         O    1h3m
4       10.102.8.0/30      DIRECT          ve 100        0/0           D    1h4m
```

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, OSPF, RIP, and BGP. OSPF redistribution supports the import of static, RIP, and BGP routes into OSPF routes.

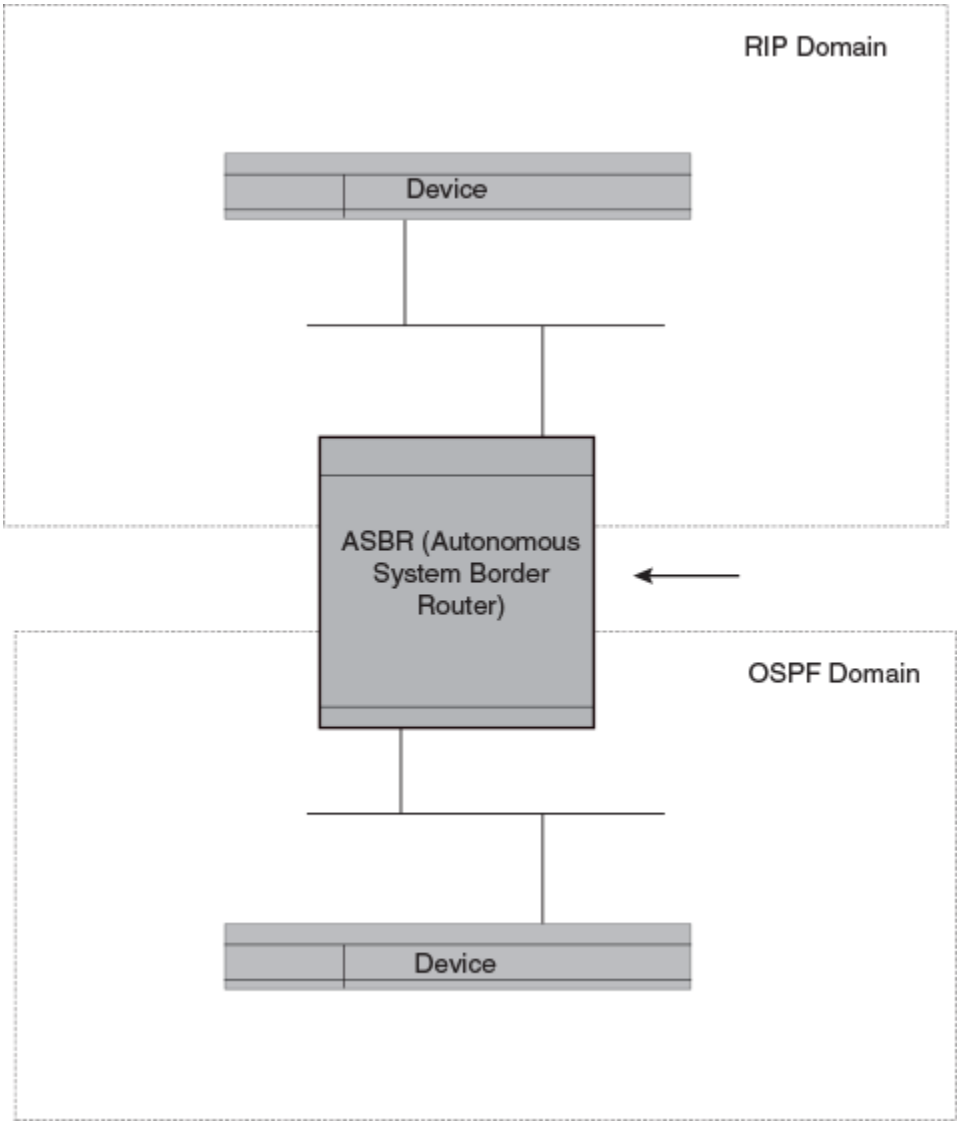
NOTE
The device advertises the default route into OSPF even if redistribution is not enabled, and even if the default route is learned through an IBGP neighbor. IBGP routes (including the default route) are not redistributed into OSPF by OSPF redistribution (for example, by the OSPF **redistribute** 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 27 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
```

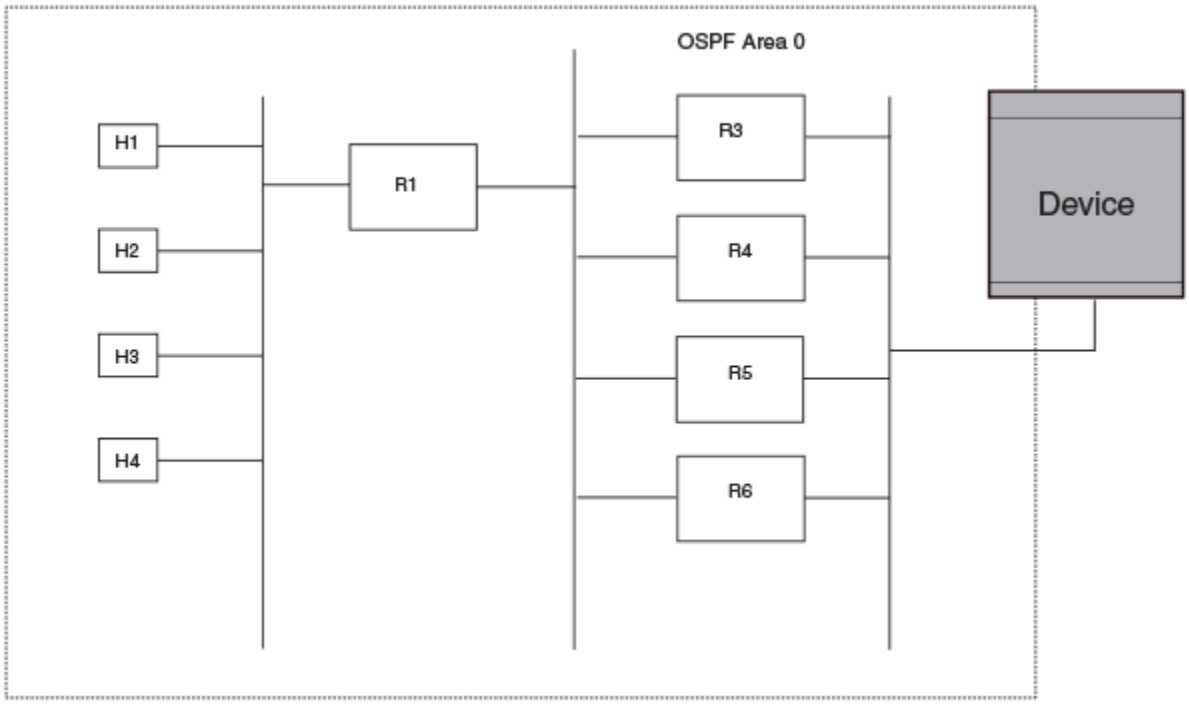
Load sharing

Ruckus 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.

On ICX 7650, ICX 7750, and ICX 7850 devices, the value range for the maximum number of load-sharing paths is from 2 through 32 ,which is controlled by the **system-max max-ecmp** command.

The device software can use the route information it learns through OSPF to determine the paths and costs.

FIGURE 28 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.

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.

NOTE
ICX 7150 devices do not support VRFs.

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.

NOTE
ICX 7150 devices do not support VRFs.

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)#
```

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
```

Re-enabling OSPFv2 compatibility with RFC 1583

OSPFv2 is compatible with RFC 1583 and maintains a single best route to an autonomous system (AS) boundary router in the OSPF routing table. Disabling this compatibility causes the OSPF routing table to maintain multiple intra-AS paths, which helps prevent routing loops. You can re-enable OSPFv2 compatibility with RFC 1583 if it has been disabled.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **ip router ospf** command to enter OSPF router configuration mode and enable OSPFv2 on the device.

```
device(config)# router ospf
```

3. Enter the **rfc1583-compatibility** command to re-enable OSPFv2 compatibility with RFC 1583.

```
device(config-ospf-router)# rfc1583-compatibility
```

The following example re-enables OSPFv2 compatibility with RFC 1583.

```
device# configure terminal
device(config)# router ospf
device(config-ospf-router)# rfc1583-compatibility
```

OSPFv2 authentication

OSFPv2 can be configured to authenticate packets using one of the following authentication algorithms:

- Plain text
- Message Digest 5 (MD5)
- Hashed Message Authentication Code-Secure Hash Algorithm 1 (HMAC-SHA-1)
- Hashed Message Authentication Code-Secure Hash Algorithm 256 (HMAC-SHA-256)

The authentication algorithms provide varying levels of security and must be configured depending on your security requirements, including any regulatory requirements such as FIPS compliance.

Authentication is implemented as detailed in RFC 2328 and RFC 5709.

Algorithms HMAC-SHA-1 and HMAC-SHA-256 are supported in FIPS-compliant deployments. MD5 and plain text are not supported in FIPS deployments.

NOTE
OSPFv2 packets are not authenticated by default. You must configure OSPFv2 authentication as required.

OSPFv2 authentication can be enabled on each interface of virtual link.

In addition to the other authentication methods, you can configure keychain authentication. For more information regarding the keychain authentication module and configuration of keychains, refer to the Keychain module section in the *Ruckus FastIron Security Configuration Guide*.

NOTE
If multiple OSPFv2 are stacked and authentication is enabled, the OSPFv2 non-stop routing (NSR) feature must be enabled explicitly. NSR is not enabled by default. If NSR is not enabled, there may be disruption to service upon stack switchover.

OSPFv2 keychain authentication

OSPFv2 can be configured to authenticate packets using the keychain authentication module. The keychain authentication module provides hitless authentication key rollover, which allows OSPFv2 to overcome the limitation of the static configuration in authentication methods that require manual intervention to change the key periodically. For each OSPFv2 protocol packet, a key is used to generate and verify a message digest. The key is valid for the entire duration of the protocol without any option to change the key string or authentication algorithm automatically. The keychain authentication module that functions as a container of keys with different attributes such as the authentication algorithm, password, and different lifetimes provides OSPFv2 with an option to choose the key that best suits its criteria and automatically change the key ID, password, and cryptographic algorithm without manual intervention.

For more information regarding the keychain authentication module and configuration of keychains, refer to the Keychain module section in the *Ruckus FastIron Security Configuration Guide*.

NOTE
If multiple OSPFv2 interfaces are deployed within a VRF or Multi-VRF deployment and keychain authentication is implemented, it is necessary to ensure the **ip ospf hello-interval** and **ip ospf dead-interval** values are configured appropriately. If these values are set too low, for example hello-interval (1 second) and dead-interval (4 seconds), it may cause performance issues and disruption to service during key rollover.

OSPFv2 authentication configuration

Authentication must be configured on each interface or virtual link, as required. The same authentication method must be enabled on peer and neighbor routers to ensure a connection is established and the packets are transmitted successfully.

Configuring plain text authentication on an OSPFv2 interface

To configure plain text authentication on an OSPFv2 interface, complete the following steps.

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 ve 1
```

3. Enter the **ip ospf authentication plain-text** command with the required plain text key string.

```
device(config-vif-1)# ip ospf authentication plain-text mystring
```

The following example enables plain text authentication using the key string "mystring" on the specified interface.

```
device# configure terminal
device(config)# interface ve 1
device(config-vif-1)# ip ospf authentication plain-text mystring
```

Configuring plain text authentication on an OSPFv2 virtual link

To configure plain text authentication on an OSPFv2 virtual link, complete the following steps.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **ip router-id** command to specify the required router.

```
device(config)# ip router-id 10.1.1.1
```

3. Enter the **router ospf** command to enter OSPFv2 configuration mode and enable OSPFv2 on the device.

```
device(config)# router ospf
```

4. Enter **area** *area-ID* **virtual-link** *virtual-link-address* **authentication plain-text** command, specifying the required plain text key string.

```
device(config-ospf-router)# area 1 virtual-link 20.1.1.1 authentication plain-text mystring
```

The following example enables plain text authentication using the key string "mystring" on the specified virtual link.

```
device# configure terminal
device(config)# ip router-id 10.1.1.1
device(config)# router ospf
device(config-ospf-router)# area 1 virtual-link 20.1.1.1 authentication plain-text mystring
```

Configuring MD5 authentication on an OSPFv2 interface

To configure MD5 authentication on an OSPFv2 interface, complete the following steps.

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 ve 1
```

3. Enter the **ip ospf authentication md5** command with the required parameters. The following example enables MD5 authentication with key ID 10 and key string "mymd5passwordkey".

```
device(config-vif-1)# ip ospf authentication md5 key-id 10 key mymd5passwordkey
```

The following example enables MD5 authentication using key ID 10 and key string "mymd5passwordkey" on the specified interface.

```
device# configure terminal
device(config)# interface ve 1
device(config-vif-1)# ip ospf authentication md5 key-id 10 key mymd5passwordkey
```

Configuring MD5 authentication on an OSPFv2 virtual link

To configure MD5 authentication on an OSPFv2 virtual link, complete the following steps.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **ip router-id** command to specify the required router.

```
device(config)# ip router-id 10.1.1.1
```

3. Enter the **router ospf** command to enter OSPFv2 configuration mode and enable OSPFv2 on the device.

```
device(config)# router ospf
```

4. Enter **area** *area-ID* **virtual-link** *virtual-link-address* **authentication md5** command, with the required parameters. The following example enables MD5 authentication with key ID 10 and key string "mymd5passwordkey".

```
device(config-ospf-router)# area 1 virtual-link 20.1.1.1 authentication md5 key-id 10 key
mymd5passwordkey
```

The following example enables MD5 authentication using the key ID 10 and key string 'mymd5passwordkey'.

```
device# configure terminal
device(config)# ip router-id 10.1.1.1
device(config)# router ospf
device(config-ospf-router)# area 1 virtual-link 20.1.1.1 authentication md5 key-id 10 key mymd5passwordkey
```

Configuring HMAC-SHA-1 or HMAC-SHA-256 authentication on an OSPFv2 interface

To configure HMAC-SHA-1 or HMAC-SHA-256 authentication on an OSPFv2 interface, complete the following steps.

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 ve 1
```

3. Enter the **ip ospf authentication** command with the required authentication option and parameters. The following example enables HMAC-SHA-1 authentication with key ID 10 and key string "mypasswordkey".

```
device(config-vif-1)# ip ospf authentication hmac-sha-1 key-id 10 key mypasswordkey
```

The following example enables HMAC-SHA-1 authentication using the key ID 10 and key string "mypasswordkey" on the specified interface.

```
device# configure terminal
device(config)# interface ve 1
device(config-vif-1)# ip ospf authentication hmac-sha-1 key-id 10 key mypasswordkey
```

Configuring HMAC-SHA-1 or HMAC-SHA-256 on an OSPFv2 virtual link

To configure HMAC-SHA-1 or HMAC-SHA-256 authentication on an OSPFv2 virtual link, complete the following steps.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **ip router-id** command to specify the required router.

```
device(config)# ip router-id 10.1.1.1
```

3. Enter the **router ospf** command to enter OSPFv2 configuration mode and enable OSPFv2 on the device.

```
device(config)# router ospf
```

4. Enter **area** *area-ID* **virtual-link** *virtual-link-address* **authentication** command, with the required authentication option and parameters. The following example enables HMAC-SHA-1 authentication with the key ID 10 and key string "mypasswordkey".

```
device(config-ospf-router)# area 1 virtual-link 20.1.1.1 authentication hmac-sha-1 key-id 10 key
mypasswordkey
```

The following example enables HMAC-SHA-1 authentication using the key ID 10 and key string "mypasswordkey" on the specified virtual link.

```
device# configure terminal
device(config)# ip router-id 10.1.1.1
device(config)# router ospf
device(config-ospf-router)# area 1 virtual-link 20.1.1.1 authentication hmac-sha-1 key-id 10 key
mypasswordkey
```

Configuring keychain authentication on an OSPFv2 interface

To configure keychain authentication on an OSPFv2 interface, complete the following steps.

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 ve 1
```

3. Enter the **ip ospf authentication keychain** command with the required keychain name. The following example enables keychain authentication with the keychain "mykeychain".

```
device(config-vif-1)# ip ospf authentication keychain mykeychain
```

The following example enables keychain authentication using the keychain name "mykeychain" on the specified interface.

```
device# configure terminal
device(config)# interface ve 1
device(config-vif-1)# ip ospf authentication keychain mykeychain
```

Configuring keychain authentication on an OSPFv2 virtual link

To configure keychain authentication on an OSPFv2 virtual link, complete the following steps.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **ip router-id** command to specify the required router.

```
device(config)# ip router-id 10.1.1.1
```

3. Enter the **router ospf** command to enter OSPFv2 configuration mode and enable OSPFv2 on the device.

```
device(config)# router ospf
```

4. Enter **area** *area-ID* **virtual-link** *virtual-link-address* **authentication keychain** command, with the required keychain name. The following example enables keychain authentication with the keychain name "mykeychain".

```
device(config-ospf-router)# area 1 virtual-link 20.1.1.1 authentication keychain mykeychain
```

The following example enables keychain authentication using the keychain name "mykeychain" on the specified virtual link.

```
device# configure terminal
device(config)# ip router-id 10.1.1.1
device(config)# router ospf
device(config-ospf-router)# area 1 virtual-link 20.1.1.1 authentication keychain mykeychain
```

Configuring authentication key activation wait time on an OSPFv2 interface

The key activation wait time configures the time before an authentication key change is activated for an OSPFv2 interface. This allows you to coordinate a key change across devices to ensure there is no disruption to service.

To configure the authentication key activation wait time on an OSPFv2 interface, complete the following steps.

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 ve 1
```

3. Enter the **ip ospf authentication key-activation-wait-time** command with the required wait time value. The wait time can be set from 0 through 14400 seconds. The following example configures a wait time of 600 seconds.

```
device(config-vif-1)# ip ospf authentication key-activation-wait-time 600
```

The following example enables an authentication key activation wait time of 600 seconds on the specified interface.

```
device# configure terminal
device(config)# interface ve 1
device(config-vif-1)# ip ospf authentication key-activation-wait-time 600
```

Configuring authentication key activation wait time on an OSPFv2 virtual link

The key activation wait time configures the time before an authentication key change is activated for an OSPFv2 virtual link. This allows you to coordinate a key change across devices to ensure there is no disruption to service.

To configure the authentication key activation wait time on an OSPFv2 virtual link, complete the following steps.

1. Enter the **configure terminal** command to access global configuration mode.

   ```
   device# configure terminal
   ```

2. Enter the **ip router-id** command to specify the required router.

   ```
   device(config)# ip router-id 10.1.1.1
   ```

3. Enter the **router ospf** command to enter OSPFv2 configuration mode and enable OSPFv2 on the device.

   ```
   device(config)# router ospf
   ```

4. Enter **area** *area-ID* **virtual-link** *virtual-link-address* **authentication** command, with the required wait time value. The wait time can be set from 0 through 14400 seconds. The following example configures a wait time of 600 seconds.

   ```
   device(config-ospf-router)# area 1 virtual-link 20.1.1.1 authentication key-activation-wait-time 600
   ```

The following example enables an authentication key activation wait time of 600 seconds on the specified virtual link.

```
device# configure terminal
device(config)# ip router-id 10.1.1.1
device(config)# router ospf
device(config-ospf-router)# area 1 virtual-link 20.1.1.1 authentication key-activation-wait-time 600
```

Displaying OSPFv2 results

To view the OSPFv2 interface details and authentication settings, use the **show ip ospf interface** command. This is command is optional and allows you to verify the OSPFv2 details.

To view the OSPFv2 interface details and authentication settings, complete the following steps.

Enter the **show ip ospf interface** command to display general OSPFv2 information.

```
device# show ip ospf interface ve 500
ve 500  admin up, oper up, ospf enabled, state up
     IP Address 50.1.1.2, Area 1
     Database Filter: Not Configured
     State DR, Pri 1, Cost 1, Options 2, Type broadcast Events 3
     Timers(sec): Transmit 1, Retrans 5, Hello 10, Dead 40
     DR:  Router ID 71.50.71.50       Interface Address 50.1.1.2
     BDR: Router ID 0.0.0.0           Interface Address 0.0.0.0
                    Packets Received        Packets Sent
     Hello                         127                 125
     Database                        3                   3
     LSA Req                         1                   0
     LSA Upd                         8                   2
     LSA Ack                         1                   5
     Packet Errors: Ospf Auth Key 4,
     Neighbor Count = 0, Adjacent Neighbor Count= 0
     In-Use Authentication: hmac-sha-256, Key: ********, Key-Id:    1
```

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, Ruckus 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.

NOTE
For the ICX 7150, a maximum of 4 OSPF areas is supported for each OSPF instance.

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.

NOTE
For the ICX 7150, a maximum of 4 OSPF areas is supported for each OSPF instance.

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 ve 1
```

3. Enter the **ipv6 address** command to add an IPv6 address to the interface.

```
device(config-vif-1)# ipv6 address 2001:db8:93e8:cc00::1
```

4. Enter the **ipv6 ospf area** command.

```
device(config-vif-1)# ipv6 ospf area 0
```

Area 0 is assigned to the specified interface with the IPv6 address of 2001:db8:93e8:cc00::1.

5. Enter the **exit** command to return to global configuration mode.

```
device(config-vif-1)# exit
```

6. Enter the **interface** command and specify an interface.

```
device(config)# interface ve 2
```

7. Enter the **ipv6 address** command to add an IPv6 address to the interface.

```
device(config-vif-2)# ipv6 address 2001:db8:93e8:cc00::2
```

8. Enter the **ipv6 ospf area** command.

```
device(config-vif-2)# ipv6 ospf area 1
```

Area 1 is assigned to the specified interface with the IPv6 address of 2001:db8:93e8:cc00::1.

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 ve 1
device(config-vif-1)# ipv6 address 2001:db8:93e8:cc00::1
device(config-vif-1)# ipv6 ospf area 0
device(config-vif-1)# exit
device(config)# interface ve 2
device(config-vif-2)# ipv6 address 2001:db8:93e8:cc00::2
device(config-vif-2)# 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 2740) 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.

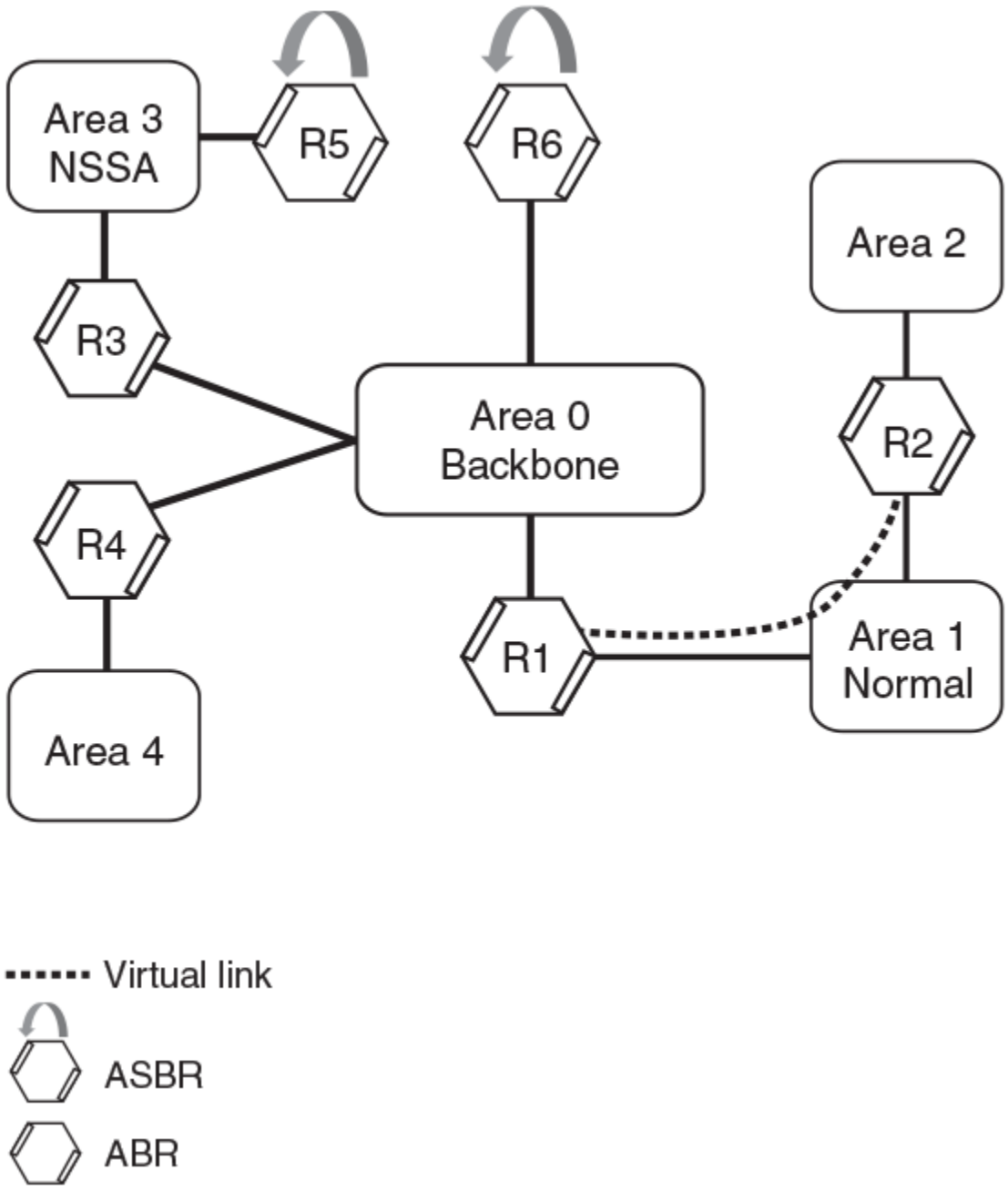
Virtual links

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 29 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+
- 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.

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 Ruckus 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.

NOTE
ICX 7150 devices do not support VRFs.

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.

NOTE
ICX 7150 devices do not support VRFs.

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.

NOTE
ICX 7150 devices do not support VRFs.

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.

The table below shows GR support for OSPFv3.

TABLE 18 Graceful restart support for OSPFv3

GR restarting router	GR helper	NSR (no neighbor support needed)
No	Yes	Yes

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 authentication

OSFPv3 can be configured to authenticate packets using one of the following authentication methods:

- Authentication trailer

  Authentication trailer allows authentication of OSPFv3 packets using Hashed Message Authentication Code-Secure Hash Algorithm 1 (HMAC-SHA-1) or Hashed Message Authentication Code-Secure Hash Algorithm 256 (HMAC-SHA-256)
- IP Security (IPsec)

  IP Security (IPsec) secures OSPFv3 communications by authenticating and encrypting each IP packet of a communication session.

  NOTE
  OSPFv3 packets are not authenticated by default. You must configure OSPFv3 authentication as required.

OSPFv3 authentication can be enabled on each interface, virtual link or area.

OSPFv3 authentication trailer

OSFPv3 can be configured to authenticate packets using one of the following authentication algorithms:

- Hashed Message Authentication Code-Secure Hash Algorithm 1 (HMAC-SHA-1)
- Hashed Message Authentication Code-Secure Hash Algorithm 256 (HMAC-SHA-256)

The authentication algorithms provide varying levels of security and must be configured depending on your security requirements, including any regulatory requirements such as FIPS compliance.

Algorithms HMAC-SHA-1 and HMAC-SHA-256 are supported in FIPS-compliant deployments.

Authentication is implemented as detailed in RFC 2328 and RFC 5709.

NOTE
OSPFv3 packets are not authenticated by default. You must configure OSPFv3 authentication as required.

OSPFv3 authentication can be enabled on each interface, virtual link or area.

In addition to the other authentication methods, you can configure keychain authentication. For more information regarding the keychain authentication module and configuration of keychains, refer to the Keychain module section in the *Ruckus FastIron Security Configuration Guide*.

OSPFv3 keychain authentication

OSPFv3 can be configured to authenticate packets using the keychain authentication module. The keychain authentication module provides hitless authentication key rollover, which allows OSPFv3 to overcome the limitation of the static configuration in authentication methods that require manual intervention to change the key periodically. For each OSPFv3 protocol packet, a key is used to generate and verify a message digest. The key is valid for the entire duration of the protocol without any option to change the key string or authentication algorithm automatically. The keychain authentication module that functions as a container of keys with different attributes such as authentication algorithm, password, and different lifetimes provides OSPFv3 with an option to choose the key that best suits its criteria and automatically change the key ID, password, and cryptographic algorithm without manual intervention.

For more information regarding the keychain authentication module and configuration of keychains, refer to the Keychain module section in the *Ruckus FastIron Security Configuration Guide*.

OSPFv3 authentication trailer configuration

Authentication must be configured on each interface, virtual link, or area as required. The same authentication method must be enabled on peer and neighbor routers to ensure a connection is established and the packets transmitted successfully.

OSPFv3 authentication trailer is defined in RFC 7166 and RFC 6506. RFC 7166 supersedes RFC 6506 and by default FastIron implements authentication trailer in accordance with RFC 7166. However, some vendor equipment still support RFC 6506. If your deployment includes vendor equipment that support RFC 6506, you can configure authentication for the required interfaces or virtual links, as required, using the **ospf authentication rfc6506** command.

Configuring HMAC-SHA-1 or HMAC-SHA-256 authentication on an OSPFv3 interface

To configure HMAC-SHA-1 or HMAC-SHA-256 authentication on an OSPFv3 interface, complete the following steps.

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 ve 1
```

3. Enter the **ipv6 ospf authentication** command with the required authentication option and parameters. The following example enables HMAC-SHA-1 authentication with key ID 10 and key string "mypasswordkey".

```
device(config-vif-1)# ipv6 ospf authentication hmac-sha-1 key-id 10 key mypasswordkey
```

The following example enables HMAC-SHA-1 authentication using the key ID 10 and key string "mypasswordkey'" on the specified interface.

```
device# configure terminal
device(config)# interface ve 1
device(config-vif-1)# ipv6 ospf authentication hmac-sha-1 key-id 10 key mypasswordkey
```

Configuring HMAC-SHA-1 or HMAC-SHA-256 authentication on an OSPFv3 virtual link

To configure HMAC-SHA-1 or HMAC-SHA-256 authentication on an OSPFv3 virtual link, complete the following steps.

1. Enter the **configure terminal** command to access global configuration mode.

   ```
   device# configure terminal
   ```

2. Enter the **ip router-id** command to specify the required router.

   ```
   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** *area-ID* **virtual-link** *virtual-link-address* **authentication** command, with the required authentication option and parameters. The example enables HMAC-SHA-1 authentication with key ID 10 and key string "mypasswordkey".

   ```
   device(config-ospf6-router)# area 1 virtual-link 20.1.1.1 authentication hmac-sha-1 key-id 10 key
   mypasswordkey
   ```

The following example enables HMAC-SHA-1 authentication using key ID 10 and key string "mypasswordkey" on the specified virtual link.

```
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 20.1.1.1 authentication hmac-sha-1 key-id 10 key
mypasswordkey
```

Configuring HMAC-SHA-1 or HMAC-SHA-256 authentication on an OSPFv3 area

To configure an OSPFv3 area so all devices within the area use HMAC-SHA-1 or HMAC-SHA-256 authentication, complete the following steps.

1. Enter the **configure terminal** command to access global configuration mode.

   ```
   device# configure terminal
   ```

2. Enter the **ip router-id** command to specify the required router.

   ```
   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** *area-ID* **authentication** command, with the required authentication option and parameters. The following example enables HMAC-SHA-1 authentication with key ID 10 and key string "mypasswordkey".

   ```
   device(config-ospf6-router)# area 1 authentication hmac-sha-1 key-id 10 key mypasswordkey
   ```

NOTE
This configuration instructs all interfaces within the area to use HMAC-SHA-1 or HMAC-SHA-256 authentication. It is possible to remove this configuration from individual interfaces using the **ipv6 ospf authentication disable** command on the required interface.

The following example enables HMAC-SHA-1 authentication using the key ID 10 and key string "mypasswordkey" on the specified area.

```
device# configure terminal
device(config)# ip router-id 10.1.1.1
device(config)# ipv6 router ospf
device(config-ospf6-router)# area 1 authentication hmac-sha-1 key-id 10 key mypasswordkey
```

Configuring keychain authentication on an OSPFv3 interface

To configure keychain authentication on an OSPFv3 interface, complete the following steps.

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 ve 1
```

3. Enter the **ipv6 ospf authentication keychain** command with the required keychain. The following example enables keychain authentication with the keychain "mykeychain".

```
device(config-vif-1)# ipv6 ospf authentication keychain mykeychain
```

The following example enables keychain authentication using the keychain "mykeychain" on the specified interface.

```
device# configure terminal
device(config)# interface ve 1
device(config-vif-1)# ipv6 ospf authentication keychain mykeychain
```

Configuring keychain authentication on an OSPFv3 virtual link

To configure keychain authentication on an OSPFv3 virtual link, complete the following steps.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **ip router-id** command to specify the required router.

```
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** *area-ID* **virtual-link** *virtual-link-address* **authentication** command, with the required keychain. The following example enables keychain authentication with the keychain "mykeychain".

```
device(config-ospf6-router)# area 1 virtual-link 20.1.1.1 authentication keychain mykeychain
```

The following example enables keychain authentication using the keychain "mykeychain" on the specified virtual link.

```
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 20.1.1.1 authentication keychain mykeychain
```

Configuring keychain authentication on an OSPFv3 area

To configure an OSPFv3 area so all devices within the area use keychain authentication, complete the following steps.

1. Enter the **configure terminal** command to access global configuration mode.

   ```
   device# configure terminal
   ```

2. Enter the **ip router-id** command to specify the required router.

   ```
   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** *area-ID* **authentication** command, with the required keychain. The following example enables keychain authentication with the keychain "mykeychain".

   ```
   device(config-ospf6-router)# area 1 authentication keychain mykeychain
   ```

   NOTE
   This configuration instructs all interfaces within the area to use keychain authentication. It is possible to remove this configuration from individual interfaces using the **ipv6 ospf authentication disable** command on the required interface.

The following example enables keychain authentication using the keychain "mykeychain" on the specified area.

```
device# configure terminal
device(config)# ip router-id 10.1.1.1
device(config)# ipv6 router ospf
device(config-ospf6-router)# area 1 authentication keychain mykeychain
```

Configuring authentication key activation wait time on an OSPFv3 interface

The key activation wait time configures the time before an authentication key change is activated for an OSPFv3 interface. This allows you to coordinate a key change across devices to ensure there is no disruption to service.

To configure the authentication key activation wait time on an OSPFv3 interface, complete the following steps.

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 ve 1
   ```

3. Enter the **ipv6 ospf authentication key-activation-wait-time** command with the required wait time value. The wait time can be set from 0 through 14400 seconds. The following example configures a wait time of 600 seconds.

   ```
   device(config-vif-1)# ipv6 ospf authentication key-activation-wait-time 600
   ```

The following example enables an authentication key activation wait time of 600 seconds on the specified interface.

```
device# configure terminal
device(config)# interface ve 1
device(config-vif-1)# ipv6 ospf authentication key-activation-wait-time 600
```

Configuring authentication key activation wait time on an OSPFv3 virtual link

The key activation wait time configures the time before an authentication key change is activated for an OSPFv3 virtual link. This allows you to coordinate a key change across devices to ensure there is no disruption to service.

To configure the authentication key activation wait time on an OSPFv3 virtual link, complete the following steps.

1. Enter the **configure terminal** command to access global configuration mode.

   ```
   device# configure terminal
   ```

2. Enter the **ip router-id** command to specify the required router.

   ```
   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** *area-ID* **virtual-link** *virtual-link-address* **authentication** command, with the required wait time value. The wait time can be set from 0 through 14400 seconds. The following example configures a wait time of 600 seconds.

   ```
   device(config-ospf6-router)# area 1 virtual-link 20.1.1.1 authentication key-activation-wait-time 600
   ```

The following example enables an authentication key activation wait time of 600 seconds on the specified virtual link.

```
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 20.1.1.1 authentication key-activation-wait-time 600
```

Configuring RFC 6506 authentication on an OSPFv3 interface

This may be required for interoperability purposes. By default, OSPFv3 authentication is implemented in accordance with RFC 7166, which supercedes RFC 6506. However, some vendors still support RFC 6506. To configure authentication in accordance with RFC 6506 on an OSPFv3 interface, complete the following steps.

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 ve 1
   ```

3. Enter the **ipv6 ospf authentication rfc6506** command.

   ```
   device(config-vif-1)# ipv6 ospf authentication hmac-sha-1 key-id 10 key mypasswordkey
   ```

4. Enter the **ipv6 ospf authentication** command with the required authentication option and parameters. The following example enables HMAC-SHA-1 authentication with key ID 10 and key string "mypasswordkey".

   ```
   device(config-vif-1)# ipv6 ospf authentication hmac-sha-1 key-id 10 key mypasswordkey
   ```

The following example enables HMAC-SHA-1 authentication using the key ID 10 and key string "mypasswordkey'" on the specified interface.

```
device# configure terminal
device(config)# interface ve 1
device(config-vif-1)# ipv6 ospf authentication rfc6505
device(config-vif-1)# ipv6 ospf authentication hmac-sha-1 key-id 10 key mypasswordkey
```

Configuring RFC 6506 authentication on an OSPFv3 virtual link

This may be required for interoperability purposes. By default, OSPFv3 authentication is implemented in accordance with RFC 7166, which supercedes RFC 6506. However, some vendors still support RFC 6506. To configure authentication in accordance with RFC 6506 on an OSPFv3 virtual link, complete the following steps.

1. Enter the **configure terminal** command to access global configuration mode.

   ```
   device# configure terminal
   ```

2. Enter the **ip router-id** command to specify the required router.

   ```
   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** *area-ID* **virtual-link** *virtual-link-address* **authentication rfc6506** command.

   ```
   device(config-ospf6-router)# area 1 virtual-link 20.1.1.1 authentication rfc6506
   ```

5. Enter **area** *area-ID* **virtual-link** *virtual-link-address* **authentication** command, with the required authentication option and parameters. The example enables HMAC-SHA-1 authentication with key ID 10 and key string "mypasswordkey".

   ```
   device(config-ospf6-router)# area 1 virtual-link 20.1.1.1 authentication hmac-sha-1 key-id 10 key
   mypasswordkey
   ```

The following example enables HMAC-SHA-1 authentication using key ID 10 and key string "mypasswordkey" on the specified virtual link.

```
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 20.1.1.1 authentication rfc6506
device(config-ospf6-router)# area 1 virtual-link 20.1.1.1 authentication hmac-sha-1 key-id 10 key
mypasswordkey
```

Displaying OSPFv3 results

To view the OSPFv3 interface details and authentication settings, use the **show ipv6 ospf interface** command. This command is optional and allows you to verify the OSPFv3 details.

To view the OSPFv3 interface details and authentication settings, complete the following steps.

Enter the **show ipv6 ospf interface** command to display general OSPFv3 information.

```
device# show ipv6 ospf interface

ve 500  admin up, oper up, IPv6 enabled
  IPv6 Address:
      fe80::1
  Instance ID 0, Router ID 71.50.71.50
```

```
Area ID 1, Cost 1, Type BROADCAST
MTU: 1500
State BDR, Transmit Delay 1 sec, Priority 1, Link-LSA Tx not suppressed
Timer intervals :
  Hello 10, Hello Jitter 10  Dead 40, Retransmit 5
IPSec Authentication: Enabled
 KeyRolloverTime(sec): Configured: 300 Current: 0
 KeyRolloverState: NotActive
 Outbound: None
 Inbound: None
Authentication-trailer: Configured
 In-Use: hmac-sha-256, Key: ********, Key-id: 1
 Higher 32-bitSequence Number 4(0x4), Lower 32-bitSequence Number 133(0x85)
DR:50.1.1.1 BDR:71.50.71.50  Number of I/F scoped LSAs is 2
DRElection:     1 times, DelayedLSAck:     1 times
Neighbor Count = 1,   Adjacent Neighbor Count= 1
   Neighbor:
    50.1.1.1 (DR)
  Statistics of interface ve 500 :
    Type     tx         rx         tx-byte    rx-byte
    Unknown  0          0          0          0
    Hello    117        122        4668       4880
    DbDesc   2          5          116        240
    LSReq    6          0          456        0
    LSUpdate 5          8          432        1552
    LSAck    3          2          148        192
    OSPF messages dropped:
        Authentication Failure(IPSec)       : 0
        Authentication Failure(Auth-Trailer): 2
```

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.

Ruckus 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

  NOTE
  ICX 7150 devices do not support VRFs.

IPsec for OSPFv3 considerations

IPsec generates security associations and security policies based on certain user-specified parameters. Refer to the *FastIron 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 RFC 4552. 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 1/1/1 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 ve 1
```

3. Enter the **ipv6 ospf area** command to assign a specified area to the interface.

```
device(config-vif-1)# ipv6 ospf area 0
```

4. Enter **ipv6 ospf authentication ipsec spi** *value* **esp sha1** and specify a 40-character hexadecimal key.

```
device(config-vif-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 virtual Ethernet (VE) interface.

```
device# configure terminal
device(config)# interface ve 1
device(config-vif-1)# ipv6 ospf area 0
device(config-vif-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 1134567890223456789012345678901234567890
```

   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 timer

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.

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-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 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-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 0xc0a862d5(10.168.98.213)
 Running 0 days 2 hours 55 minutes 36 seconds
 Number of AS scoped LSAs is 4
 Sum of AS scoped LSAs Checksum is 18565
 External LSA Limit is 250000
 Database Overflow Interval is 10
 Database Overflow State is NOT OVERFLOWED
 Route calculation executed 15 times
 Pending outgoing LSA count 0
 Authentication key rollover interval 300 seconds
 Number of areas in this router is 3
 Router is operating as ABR
 Router is operating as ASBR, Redistribute: CONNECTED RIP
 High Priority Message Queue Full count: 0
 Graceful restart helper is enabled, strict lsa checking is disabled
 Nonstop-routing is ENABLED
```

3. The following example of the **show ipv6 ospf area** command shows detailed output for assigned OSPFv3 Area 1.

```
device> show ipv6 ospf area 1

Area 1:
  Authentication: Not Configured
  Active interface(s)attached to this area: None
  Inactive interface(s)attached to this area: ve 20  ve 30
  Number of Area scoped LSAs is 311
  Sum of Area LSAs Checksum is 9e8fff
  Statistics of Area 1:
    SPF algorithm executed 10 times
    SPF last updated: 5920 sec ago
    Current SPF node count: 1
      Router: 1 Network: 0
      Maximum of Hop count to nodes: 0
```

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    0     up      BCST 1     DROther   1/1
loopback 1   0     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

Total number of neighbors in all states: 2
Number of neighbors in state Full      : 2
RouterID        Pri State    DR              BDR             Interface  [State]
192.168.98.111    1 Full     192.168.98.111  192.168.98.213  e 4/3/1    [BDR]
192.168.98.111    1 Full     192.168.98.111  192.168.98.213  ve 17      [BDR]
```

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 1/1/8
                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
0.0.0.200       Link 897        192.168.98.213  80000007 1277 9044  64    Yes
0.0.0.200       Link 136        192.168.98.111  80000007 582  fb0b  64    Yes
0.0.0.200       Link 2049       192.168.98.213  80000006 1277 381a  64    Yes
0.0.0.200       Link 1156       192.168.98.111  80000007 582  cf38  64    Yes
0.0.0.200       Link 2052       192.168.98.213  80000004 799  5b06  64    Yes
0.0.0.200       Rtr  0          192.168.98.111  800002ea 823  cb7b  56    Yes
0.0.0.200       Rtr  0          192.168.98.213  800001c7 799  8402  56    Yes
0.0.0.200       Net  1156       192.168.98.111  80000004 823  b2d2  32    Yes
0.0.0.200       Net  136        192.168.98.111  80000008 823  aed2  32    Yes
N/A             Extn 0000021d   10.223.223.223     800000a8 1319 441e  32       Yes
```

8. The following example of the **show ipv6 ospf routes** command shows output for OSPFv3 routes.

```
device> show ipv6 ospf routes

Current Route count: 309
    Intra: 304 Inter: 4 External: 1 (Type1 0/Type2 1)
    Equal-cost multi-path: 56
    OSPF Type: IA- Intra, OA - Inter, E1 - External Type1, E2 - External Type2
   Destination                    Cost       E2Cost     Tag        Flags    Dis
E2 ::/0                           2          1          0          00000003 110
   Next_Hop_Router                Outgoing_Interface Adv_Router
   fe80::768e:f8ff:fe3e:1800      e 4/3/1            192.168.98.111
   fe80::768e:f8ff:fe3e:1800      ve 17              192.168.98.111
   Destination                    Cost       E2Cost     Tag        Flags    Dis
IA 5100::192:61:1001:0/112        3          0          0          00000007 110
   Next_Hop_Router                Outgoing_Interface Adv_Router
   fe80::768e:f8ff:fe3e:1800      e 4/3/1            192.168.98.111
   fe80::768e:f8ff:fe3e:1800      ve 17              192.168.98.111
   Destination                    Cost       E2Cost     Tag        Flags    Dis
IA 5100::192:111:2:111/128        1          0          0          00000007 110
   Next_Hop_Router                Outgoing_Interface Adv_Router
   fe80::768e:f8ff:fe3e:1800      e 4/3/1            192.168.98.111
   fe80::768e:f8ff:fe3e:1800      ve 17              192.168.98.111
   Destination                    Cost       E2Cost     Tag        Flags    Dis
IA 5100::192:111:3:111/128        1          0          0          00000007 110
   Next_Hop_Router                Outgoing_Interface Adv_Router
   fe80::768e:f8ff:fe3e:1800      e 4/3/1            192.168.98.111
--More--, next page: Space, next line: Return key, quit: Control-c
```

9. The following example of the **show ipv6 ospf database as-external** command shows information about external LSAs.

```
device> show ipv6 ospf database as-external

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
N/A             Extn 2          192.168.98.213  80000004 895  6e5e  44    Yes
     Bits: E--
     Metric: 0
     Prefix Options:
     Referenced LSType: 0
     Prefix: 5100:213:213:0:192:213:1:0/112
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
N/A             Extn 1          192.168.98.190  80001394 643  1cc9  28    Yes
     Bits: E--
     Metric: 1
     Prefix Options:
     Referenced LSType: 0
     Prefix: ::/0
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
N/A             Extn 2          192.168.98.71   80000258 132  a3ff  32    Yes
     Bits: E-T
     Metric: 1
     Prefix Options:
     Referenced LSType: 0
     Prefix: ::/0
     Tag: 1
```

10. 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
```

11. 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 192.168.98.111        --V-B V6E---R-   1  fe80::768e:f8ff:fe3e:1800 e 4/3/1
R 192.168.98.111        --V-B V6E---R-   1  fe80::768e:f8ff:fe3e:1800 ve 17
N 192.168.98.111[136]   ----- V6E---R-   1  ::                        e 4/3/1
N 192.168.98.111[1156   ----- V6E---R-   1  ::                        ve 17
```

12. 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      5100::192:213:163:0/112             Connect   Type-2       0
2      5100:213:213:0:192:213:1:0/112      Connect   Type-2       0
```

13. The following example of the **show ipv6 ospf routes** command shows information about a specified OSPFv3 route.

```
device> show ipv6 ospf routes 2001::192:111:42:111
Destination                    Cost       E2Cost     Tag        Flags    Dis
IA 2001::192:111:42:111/128       1          0         0       00000007 110
   Next_Hop_Router               Outgoing_Interface Adv_Router
   fe80::768e:f8ff:fe3e:1800     e 4/3/1            10.168.98.111
   fe80::768e:f8ff:fe3e:1800     ve 17              10.168.98.111
```

BGP4

BGP4 overview

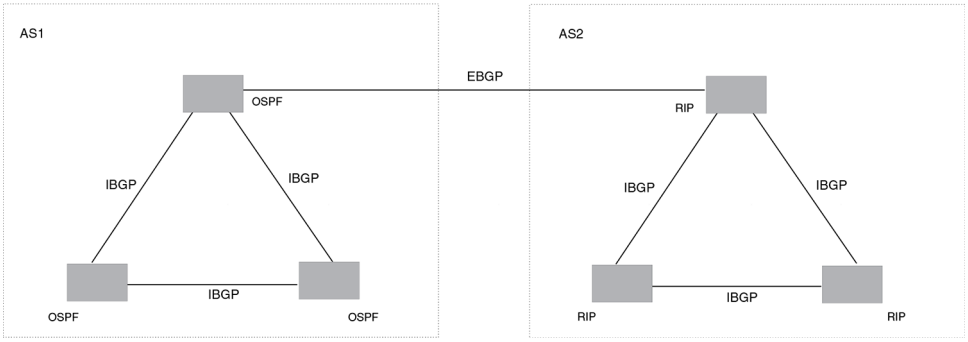
Border Gateway Protocol version 4 (BGP4) is an exterior gateway protocol that performs inter-autonomous system (AS) or inter-domain routing. It peers to other BGP-speaking systems over TCP to exchange network reachability and routing information. BGP primarily performs two types of routing: inter-AS 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 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 Ruckus 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, RIP, and OSPF. They also can redistribute static routes.

FIGURE 30 BGP4 autonomous systems



NOTE
ICX 7150 devices do not support BGP4.

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 *Ruckus FastIron 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 *Ruckus FastIron 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 Ruckus BGP4 implementation fully complies with RFC 1771. Ruckus 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)
- RFC 4893 BGP Support for Four-octet AS Number Space
- RFC 3682 Generalized TTL Security Mechanism, for eBGP Session Protection

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:

  address-family                Enter Address Family command mode
  address-filter                Configure IP address filters
  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-filter                Configure autonomous system path filters
  as-path-ignore                Ignore AS_PATH length info for best route
                                selection
  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
  community-filter              Configure community list filters
  compare-routerid              Compare router-id for identical BGP paths
  confederation                 Configure AS confederation parameters
  dampening                     Enable route-flap dampening
  default-information-originate
  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
  exit-address-family           Exit Address Family command mode
  fast-external-fallover        Reset session if link to EBGP peer goes down
  graceful-restart              Enables the BGP graceful restart capability
  local-as                      Configure local AS number
  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-recursion            Perform next-hop recursive lookup for BGP route
readvertise                   Allow readvertisement of best BGP routes not in
                              IP forwarding table
redistribute                  Redistribute information from another routing
                              protocol
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-router)# local-as 1000
```

The following example configures the local ASN for a device.

```
device# configure terminal
device(config)# router bgp
device(config-bgp-router)# local-as 1000
```

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-router)# neighbor 10.1.1.1 ?

  activate                Allow exchange of route in the current family mode
```

```
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
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
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
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-router)# 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-router)# 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-router)# local-as 1000
device(config-bgp-router)# neighbor 10.1.1.1 remote-as 1001
```

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-router)# local-as 1000
```

4. Enter the **neighbor** *peer-group-name* **peer-group** command to create a peer group.

```
device(config-bgp-router)# 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-router)# 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-router)# 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-router)# 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-router)# local-as 1000
device(config-bgp-router)# neighbor mypeergroup1 peer-group
device(config-bgp-router)# neighbor mypeergroup1 remote-as 11
device(config-bgp-router)# neighbor 10.2.2.2 peer-group mypeergroup1
device(config-bgp-router)# 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-router)# 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-router)# 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, 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-router)# redistribute connected
```

The following example redistributes connected routes into BGP4.

```
device# configure terminal
device(config)# router bgp
device(config-bgp-router)# redistribute connected
```

Advertised networks

As previously described, you can advertise routes into BGP by redistributing static, connected, RIP, or OSPF routes.

However, you can explicitly specify routes to be advertised by BGP4 by using the **network** command in BGP global 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 *Ruckus FastIron 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 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-router)# 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-router)# 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-router)# neighbor 10.2.2.2 remote-as 1001
device(config-bgp-router)# network 10.1.1.1/32
```

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-router)# 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-router)# 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-router)# 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-router)# 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-router)# 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-router)# local-as 1000
device(config-bgp-router)# 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-router)# 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-router)# 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-router)# 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-router)# 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-router)# 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-router)# 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-router)# local-as 1000
device(config-bgp-router)# 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-router)# 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-router)# 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-router)# local-as 65520
device(config-bgp-router)# 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-router)# 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-router)# 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-router)# dampening
```

The following example changes all dampening values.

```
device# configure terminal
device(config)# router bgp
device(config-bgp-router)# 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-router)# next-hop-recursion
   ```

The following example enables recursive next hop lookups.

```
device# configure terminal
device(config)# router bgp
device(config-bgp-router)# 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 19 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.

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-router)# 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-router)# 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-router)# 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-router)# 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-router)# 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-router)# neighbor 10.1.2.3 prefix-list myprefixlist in
device(config-bgp-router)# neighbor 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-router)# neighbor 10.1.2.3 prefix-list myprefixlist in
device(config-bgp-router)# 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
```

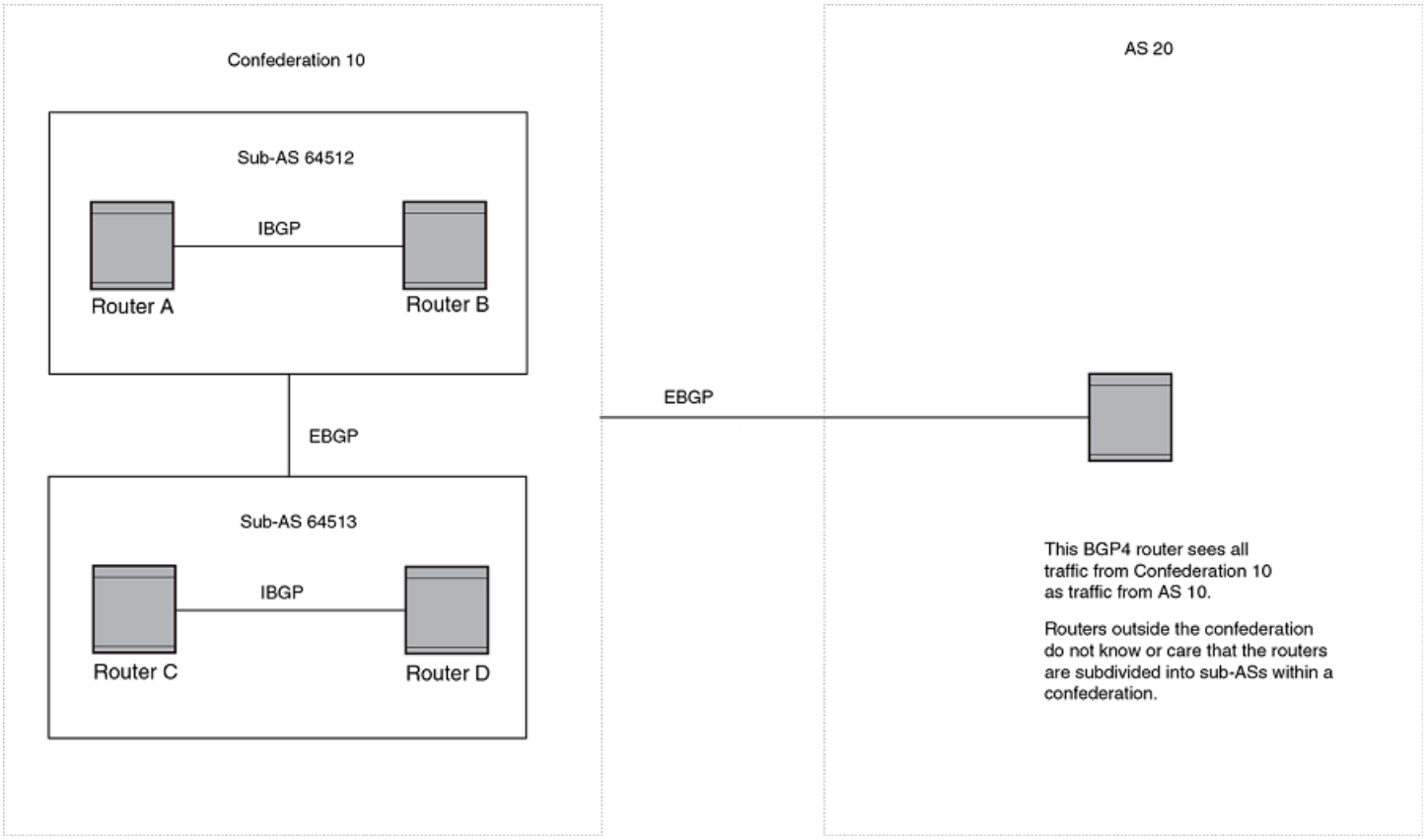
BGP4 confederations

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-router)# local-as 65520
```

4. Enter the **confederation identifier** command and specify an ASN to configure a BGP confederation identifier.

```
device(config-bgp-router)# 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-router)# 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-router)# local-as 65520
device(config-bgp-router)# confederation identifier 100
device(config-bgp-router)# 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.

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 enabled by default for both IPv4 and IPv6 address families.

NOTE
BGP4 GR can be configured for a global routing instance or for a specified VRF instance.

NOTE
BGP4 GR is supported in ICX switches in a stack.

Disabling BGP4 graceful restart

The BGP4 graceful restart (GR) feature 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 bgp** command to enable BGP routing.

```
device(config)# router bgp
```

3. Enter the **no graceful restart** command to disable GR on the device.

```
device(config-bgp-router)# no graceful-restart
```

The following example disables the GR feature.

```
device# configure terminal
device(config)# router bgp
device(config-bgp-router)# no graceful-restart
```

Re-enabling BGP4 graceful restart in bgp global configuration mode

If you disable the BGP4 graceful restart (GR) feature on a routing device you can re-enable it, providing it with the capability to inform its neighbors and peers when it is performing a restart. BGP IPv4 GR can be re-enabled in BGP 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-router)# local-as 1000
```

4. Enter the **neighbor remote-as** command to add a neighbor.

```
device(config-bgp-router)# neighbor 10.10.10.1 remote-as 2
```

5. Enter the **graceful-restart** command to re-enable the graceful restart feature.

```
device(config-bgp-router)# graceful-restart
```

6. 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-router)# 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-router)# 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-router)# graceful-restart stale-routes-time 100
     ```

The following example re-enables the GR feature.

```
device# configure terminal
device(config)# router bgp
device(config-bgp-router)# local-as 1000
device(config-bgp-router)# neighbor 10.10.10.1 remote-as 2
device(config-bgp-router)# graceful-restart
```

The following example re-enables the GR feature and sets the purge time to 120 seconds, over-writing the default value.

```
device# configure terminal
device(config)# router bgp
device(config-bgp-router)# local-as 1000
device(config-bgp-router)# neighbor 10.10.10.1 remote-as 2
device(config-bgp-router)# graceful-restart purge-time 120
```

The following example re-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-router)# local-as 1000
device(config-bgp-router)# neighbor 10.10.10.1 remote-as 2
device(config-bgp-router)# graceful-restart restart-time 180
```

The following example re-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-router)# local-as 1000
device(config-bgp-router)# neighbor 10.10.10.1 remote-as 2
device(config-bgp-router)# 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.

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-router)# local-as 65520
```

4. Enter the **neighbor remote-as** command to add a neighbor.

```
device(config-bgp-router)# 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-router)# 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-router)# local-as 65520
device(config-bgp-router)# neighbor 10.10.10.1 remote-as 2
device(config-bgp-router)# 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-router)# 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-router)# neighbor 10.1.1.1 allowas-in 3
```

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 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
```

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.

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 Ruckus 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 *Ruckus FastIron 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: 97371
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      10.3.0.0/8          192.168.4.106              100        0      BE
         AS_PATH: 65001 4355 701 80
2      10.4.0.0/8          192.168.4.106              100        0      BE
         AS_PATH: 65001 4355 1
3      10.60.212.0/22      192.168.4.106              100        0      BE
         AS_PATH: 65001 4355 701 1 189
4      10.6.0.0/8          192.168.4.106              100        0      BE
         AS_PATH: 65001 4355 3356 7170 1455
5      10.8.1.0/24         192.168.4.106   0          100        0      BE
         AS_PATH: 65001
```

   This example shows general BGP4 route information.

3. Enter the **show ip bgp** command.

```
device> show ip bgp

Total number of BGP Routes: 1
Status codes: s suppressed, d damped, h history, * valid, > best, i internal, S stale
Origin codes: i - IGP, e - EGP, ? - incomplete
    Network        Next Hop      Metric  LocPrf   Weight Path
*> 10.1.1.0/24    192.168.1.5    1       100      0      90000 100 200 65535
65536 65537 65538 65539 75000
```

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
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
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 pg1

1   BGP peer-group is pg
    Description: peer group abc
       SendCommunity: yes
       NextHopSelf: yes
       DefaultOriginate: yes
    Members:
       IP Address: 10.168.10.10, AS: 65111
```

This example shows output for one BGP peer group, called “pg1”.

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

  neighbors              Details on TCP and BGP neighbor connections
    Total number of BGP Neighbors: 1
1   IP Address: 192.168.1.1, AS: 7701000 (IBGP), RouterID: 192.168.1.1, VRF: default-vrf
    State: ESTABLISHED, Time: 0h3m33s, KeepAliveTime: 60, HoldTime: 180
       KeepAliveTimer Expire in 49 seconds, HoldTimer Expire in 177 seconds
    Minimal Route Advertisement Interval: 0 seconds
       RefreshCapability: Received
    Messages:    Open    Update  KeepAlive Notification Refresh-Req
       Sent    : 1       0       5         0            0
       Received: 1       1       5         0            0
    Last Update Time: NLRI       Withdraw          NLRI       Withdraw
                  Tx: ---        ---           Rx: 0h3m33s    ---
    Last Connection Reset Reason:Unknown
    Notification Sent:     Unspecified
    Notification Received: Unspecified
    Neighbor NLRI Negotiation:
      Peer Negotiated IPV4  unicast  capability
      Peer configured for IPV4 unicast  Routes
    Neighbor AS4 Capability Negotiation:
      Peer Negotiated AS4  capability
      Peer configured for AS4  capability

 As-path attribute count: 1
    Outbound Policy Group:
       ID: 1, Use Count: 1
    TCP Connection state: ESTABLISHED, flags:00000044 (0,0)
     Maximum segment size: 1460
    TTL check: 0, value: 0, rcvd: 64
       Byte Sent:   148, Received: 203
       Local host:  192.168.1.2, Local  Port: 179
       Remote host: 192.168.1.1, Remote Port: 8041
       ISentSeq:    1656867  SendNext:    1657016  TotUnAck:          0
       TotSent:         149  ReTrans:          19  UnAckSeq:    1657016
       IRcvSeq:     1984547  RcvNext:     1984751  SendWnd:       64981
       TotalRcv:        204  DupliRcv:        313  RcvWnd:        65000
       SendQue:           0  RcvQue:            0  CngstWnd:       5840
```

   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 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
       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 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        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)
- Next hop attributes that use IPv6 addresses

NOTE
The implementation of BGP4+ supports the advertising of routes among different address families. However, it supports BGP4+ unicast routes only; it does not currently support BGP4+ multicast routes.

NOTE
ICX 7150 devices do not support BGP4+.

BGP global mode

Configurations that are not specific to address family configuration are available in the BGP global configuration mode.

```
device(config-bgp-router)# ?

Possible completions:

  address-family                Enter Address Family command mode
  address-filter                Configure IP address filters
  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-filter                Configure autonomous system path filters
  as-path-ignore                Ignore AS_PATH length info for best route
                                selection
  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
  community-filter              Configure community list filters
  compare-routerid              Compare router-id for identical BGP paths
  confederation                 Configure AS confederation parameters
  dampening                     Enable route-flap dampening
  default-information-originate
  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
  local-as                      Configure local AS number
  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-recursion            Perform next-hop recursive lookup for BGP route
  readvertise                   Allow readvertisement of best BGP routes not in
                                IP forwarding table
  redistribute                  Redistribute information from another routing
                                protocol
  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-router)# 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
  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
  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
  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.

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
  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-originate   Originate Default Information
  default-metric                  Set metric of redistributed routes
  graceful-restart                Enables the BGP graceful restart capability
  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
  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
  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
  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
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
```

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-router)# 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-router)# 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-router)# 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-router)# local-as 1000
device(config-bgp-router)# neighbor 2001:db8:93e8:cc00::1 remote-as 1001
device(config-bgp-router)# address-family ipv6 unicast
device(config-bgp-ipv6u)# neighbor 2001:db8:93e8:cc00::1 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-router)# 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-router)# neighbor fe80:4398:ab30:45de::1 remote-as 1001
```

5. Enter the **neighbor update-source** command to specify an interface.

```
device(config-bgp-router)# neighbor fe80:4398:ab30:45de::1 update-source ethernet 1/3/1
```

6. Enter the **address-family ipv6 unicast** command to enter IPv6 address family configuration mode.

```
device(config-bgp-router)# 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-router)# local-as 1000
device(config-bgp-router)# neighbor fe80:4398:ab30:45de::1 remote-as 1001
device(config-bgp-router)# neighbor fe80:4398:ab30:45de::1 update-source ethernet 1/3/1
device(config-bgp-router)# 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-route-mapmyroutemap)# 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-router)# local-as 1000
```

4. Enter the **neighbor peer-group** command to create a peer group.

```
device(config-bgp-router)# 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-router)# neighbor mypeergroup1 remote-as 11
```

6. Enter the **neighbor peer-group** command to associate a neighbor with the peer group.

```
device(config-bgp-router)# 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-router)# 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-router)# 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-router)# local-as 1000
device(config-bgp-router)# neighbor mypeergroup1 peer-group
device(config-bgp-router)# neighbor mypeergroup1 remote-as 11
device(config-bgp-router)# neighbor 2001:2018:8192::125 peer-group mypeergroup1
device(config-bgp-router)# neighbor 2001:2018:8192::124 peer-group mypeergroup1
device(config-bgp-router)# 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-router)# local-as 1000
```

4. Enter the **neighbor peer-group** command, specifying a peer group, to create a peer group.

```
device(config-bgp-router)# neighbor p1 peer-group
```

5. Enter the **neighbor remote-as** command to specify the ASN of the peer group.

```
device(config-bgp-router)# 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-router)# 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-router)# 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-router)# 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-router)# local-as 1000
device(config-bgp-router)# neighbor p1 peer-group
device(config-bgp-router)# neighbor p1 remote-as 11
device(config-bgp-router)# neighbor 2001:2018:8192::124 peer-group p1
device(config-bgp-router)# neighbor 10.0.0.1 peer-group p1
device(config-bgp-router)# 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 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-router)# 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-router)# 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-router)# neighbor fe80:4398:ab30:45de::1 remote-as 1001
device(config-bgp-router)# 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-router)# 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-router)# 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-router)# local-as 1000
```

4. Enter the **address-family ipv6 unicast** command to enter IPv6 address family configuration mode.

```
device(config-bgp-router)# 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-router)# local-as 1000
device(config-bgp-router)# 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-router)# 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-router)# 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-router)# 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-router)# address-family ipv6 unicast
device(config-bgp-ipv6u)# 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-router)# 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-router)# 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-router)# 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-router)# local-as 1000
```

4. Enter the **address-family ipv6 unicast** command to enter IPv6 address family configuration mode.

```
device(config-bgp-router)# 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-router)# local-as 1000
device(config-bgp-router)# 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-router)# 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-router)# 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-router)# 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-router)# 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-router)# 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-router)# 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-router)# 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-router)# 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-router)# local-as 1000
device(config-bgp-router)# neighbor fe80:4398:ab30:45de::1 remote-as 1001
device(config-bgp-router)# 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, 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-router)# 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-router)# 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-router)# 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-router)# 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-router)# local-as 65520
```

4. Enter the **address-family ipv6 unicast** command to enter address family IPv6 unicast configuration mode.

```
device(config-bgp-router)# 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-router)# local-as 65520
device(config-bgp-router)# address-family ipv6 unicast
device(config-bgp-ipv6u)# neighbor 2001:db8:e0ff:783a::4 weight 200
```

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-router)# 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-router)# 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-router)# 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-router)# 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 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-router)# local-as 65520
```

4. Enter the **confederation identifier** command and specify an ASN to configure a BGP confederation identifier.

```
device(config-bgp-router)# 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-router)# 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-router)# local-as 65520
device(config-bgp-router)# confederation identifier 100
device(config-bgp-router)# 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 a community ACL

A BGP community ACL can be configured, and BGP community attributes set in a route map instance.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **ip community-list extended** command using the **permit** keyword to configure a BGP community ACL.

```
device(config)# ip community-list extended 1 permit ^[1-2]23
```

3. Enter the **route-map** *name* command to create and define a route map and enter route map configuration mode.

```
device(config)# route-map ComRmap permit 10
```

4. Enter the **match community** command and specify a community list name.

```
device(config-route-map-ComRmap)# match community 1
```

5. Enter the **set community** command to set the BGP community attributes.

```
device(config-route-map-ComRmap)# set community 323:1 additive
```

6. Enter the **exit** command to return to global configuration mode.

```
device(config-route-map-ComRmap)# exit
```

7. Enter the **route-map** *name* command to define a route map and enter route map configuration mode.

```
device(config)# route-map sendComRmap permit 10
```

8. Enter the **set community** command to set the BGP community attributes.

```
device(config-route-map-sendComRmap)# set community  3:3
```

The following example configures a BGP community ACL and sets the BGP community attributes in a route map instance.

```
device# configure terminal
device(config)# ip community-list extended 1 permit ^[1-2]23
device(config)# route-map ComRmap permit 10
device(config-route-map-ComRmap)# match community 1
device(config-route-map-ComRmap)# set community 323:1 additive
device(config-route-map-ComRmap)# exit
device(config)# route-map sendComRmap permit 10
device(config-route-map-sendComRmap)# set community  3:3
```

Applying a BGP extended community filter

A BGP extended community filter can be applied.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **ip community-list extended** command using the **permit** keyword to configure a BGP community ACL.

```
device(config)# ip community-list extended 1 permit ^[1-2]23
```

3. Enter the **route-map** *name* command to create and define a route map and enter route map configuration mode.

```
device(config)# route-map ComRmap permit 10
```

4. Enter the **match community** command and specify a community list name.

```
device(config-route-map-ComRmap)# match community 1
```

5. Enter the **set local-preference** command and specify a value to set a BGP local-preference path attribute.

```
device(config-route-map-ComRmap)# set local-preference 200
```

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-router)# local-as 1000
```

8. Enter the **neighbor** *ipv6-address* **remote-as** command to specify the ASN in which the remote neighbor resides.

```
device(config-bgp-router)# neighbor fe80:4398:ab30:45de::1 remote-as 1001
```

9. Enter the **neighbor** *ipv6-address* **update-source** command to specify an interface.

```
device(config-bgp-router)# neighbor fe80:4398:ab30:45de::1 update-source ve 1000
```

10. Enter the **address-family** command and specify the **ipv6** and **unicast** keywords to enter IPv6 address family configuration mode.

```
device(config-bgp-router)# address-family ipv6 unicast
```

11. Enter the **neighbor** *ipv6-address* **activate** command to enable the exchange of information with the neighbor.

```
device(config-bgp-ipv6u)# neighbor fe80:4398:ab30:45de::1 activate
```

12. Enter the **neighbor** *ipv6-address* **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 ComRmap
```

13. Enter the **neighbor** *ipv6-address* **send-community** command 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
```

The following example applies a BGP extended community filter.

```
device# configure terminal
device(config)# ip community-list extended 1 permit ^[1-2]23
device(config)# route-map ComRmap permit 10
device(config-route-map-ComRmap)# match community 1
device(config-route-map-ComRmap)# set local-preference 200
device(config)# router bgp
device(config-bgp-router)# local-as 1000
device(config-bgp-router)# neighbor fe80:4398:ab30:45de::1 remote-as 1001
device(config-bgp-router)# neighbor fe80:4398:ab30:45de::1 update-source ve 1000
device(config-bgp-router)# 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 ComRmap
device(config-bgp-ipv6u)# neighbor fe80:4398:ab30:45de::1 send-community
```

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 enabled by default in both IPv4 and IPv6 address families.

NOTE
BGP4+ GR can be configured for a global routing instance or for a specified VRF instance.

NOTE
BGP4+ GR is supported in ICX switches in a stack.

Disabling BGP4+ graceful restart

The BGP4+ graceful restart (GR) feature 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 bgp** command to enable BGP routing.

```
device(config)# router bgp
```

3. (Optional) Enter the **address-family** command and specify the **ipv6** and **unicast** keywords to enter IPv6 address family configuration mode.

```
device(config-bgp-router)# address-family ipv6 unicast
```

4. Enter the **no graceful restart** command to disable graceful restart at the IPv6 address family configuration level.

```
device(config-bgp-ipv6u))# no graceful-restart
```

In the following example, the graceful restart feature is disabled at the IPv6 address family configuration level.

```
device# configure terminal
device(config)# router bgp
device(config-bgp-router)# address-family ipv4 unicast
device(config-bgp-ipv6u))# no graceful-restart
```

Re-enabling BGP4+ graceful restart

If the BGP4+ graceful restart (GR) feature is disabled on a routing device, it can be re-enabled, providing it with the capability to inform its neighbors and peers when it is performing a restart.

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-router)# local-as 1000
```

4. Enter the **neighbor** *ipv6-address* **remote-as** command to specify the autonomous system ASN in which the remote neighbor resides.

```
device(config-bgp-router)# neighbor 1000::1 remote-as 2
```

5. Enter the **address-family** command and specify the **ipv6** and **unicast** keywords to enter IPv6 address family configuration mode.

```
device(config-bgp-router)# address-family ipv6 unicast
```

6. Enter the **neighbor** *ipv6-address* **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 re-enables the graceful restart feature.

```
device# configure terminal
device(config)# router bgp
device(config-bgp-router)# local-as 1
device(config-bgp-router)# neighbor 1000::1 remote-as 2
device(config-bgp-router)# address-family ipv6 unicast
device(config-bgp-ipv6u)# neighbor 1000::1 activate
device(config-bgp-ipv6u)# graceful-restart
```

The following example re-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-router)# local-as 1
device(config-bgp-router)# neighbor 1000::1 remote-as 2
device(config-bgp-router)# address-family ipv6 unicast
device(config-bgp-ipv6u)# neighbor 1000::1 activate
device(config-bgp-ipv6u)# graceful-restart purge-time 300
```

The following example re-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-router)# local-as 1
device(config-bgp-router)# neighbor 1000::1 remote-as 2
device(config-bgp-router)# address-family ipv6 unicast
device(config-bgp-ipv6u)# neighbor 1000::1 activate
device(config-bgp-ipv6u)# graceful-restart restart-time 180
```

The following example re-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-router)# local-as 1
device(config-bgp-router)# neighbor 1000::1 remote-as 2
device(config-bgp-router)# 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.

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-router)# local-as 65520
```

4. Enter the **address-family ipv6 unicast** command to enter IPv6 address family unicast configuration mode.

```
device(config-bgp-router)# 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-router)# local-as 65520
device(config-bgp-router)# 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-router)# 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-router)# 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: 113.1.1.1   Local AS Number: 65020
 Confederation Identifier: not configured
 Confederation Peers:
 Maximum Number of IP ECMP Paths Supported for Load Sharing: 1
 Number of Neighbors Configured: 2, UP: 1
 Number of Routes Installed: 5, Uses 430 bytes
 Number of Routes Advertising to All Neighbors: 7 (7 entries), Uses 336 bytes
 Number of Attribute Entries Installed: 4, Uses 360 bytes
 Neighbor Address     AS#   State  Time    Rt:Accepted Filtered Sent ToSend
 2001:db8:113:113::2  65001 CONN   1d14h32m       0      0     0      4
 2001:db8:400:400::2  65020 ESTAB  3h59m24s       2      0     3      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: 4
1       Next Hop  : ::                                          MED :1
        Origin:IGP
        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   : (length 0)
        AsPathLen: 0  AsNum: 0,      SegmentNum: 0, Neighboring As: 1, Source As 0
        Address: 0x2a8bd092  Hash:364 (0x1000000)
        Links: 0x0, 0x0
        Reference Counts: 2:0:4, Magic: 3
```

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 peer_group1
      Address family : IPV4 Unicast
        no activate
      Address family : IPV4 Multicast
        no activate
      Address family : IPV6 Unicast
        activate
      Address family : IPV6 Multicast
        no activate
      Address family : VPNV4 Unicast
        no activate
      Address family : L2VPN VPLS
        no activate
    Members:
       IP Address: 2000:400:400:400::3, AS: 65020
```

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:db8:10:10::/64 ::         1                        100      32768  BL
         AS_PATH:
2      2001:db8:113:113::/64  ::
                                      1                        100      32768  BL
         AS_PATH:
3      2001::db8:400::/64  ::
                                      0        100      32768  BL
         AS_PATH:
4      2001:db8:400:400::/64  2001:db8:400:400::2
                                      0         400      0      I
         AS_PATH: 65005 65010
```

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:db8:10:10::/64,  Status: BL,  Age: 8h31m39s
       NEXT_HOP: ::, Learned from Peer: Local Router
       LOCAL_PREF: 100,  MED: 0,  ORIGIN: incomplete,  Weight: 32768
       AS_PATH:
       Adj_RIB_out count: 3,  Admin distance 1
2 Prefix: 2001:db8:113:113::/64,  Status: BL,  Age: 6h58m35s
       NEXT_HOP: ::, Learned from Peer: Local Router
       LOCAL_PREF: 100,  MED: 0,  ORIGIN: igp,  Weight: 32768
       AS_PATH:
       Adj_RIB_out count: 3,  Admin distance 1
3 Prefix: 2001:db8:202:202::/64,  Status: BI,  Age: 5h42m36s
       NEXT_HOP: 2001:db8:400:400::2, Metric: 0,  Learned from Peer: 2001:db8:400:400::2 (65020)
       LOCAL_PREF: 400,  MED: 0,  ORIGIN: incomplete,  Weight: 0
       AS_PATH: 65005 65010
       Adj_RIB_out count: 1,  Admin distance 200
4 Prefix: 2001:db8:400:400::/64,  Status: BL,  Age: 5h43m14s
       NEXT_HOP: ::, Learned from Peer: Local Router
       LOCAL_PREF: 100,  MED: 0,  ORIGIN: igp,  Weight: 32768
       AS_PATH:
       Adj_RIB_out count: 3,  Admin distance 1
```

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: 2
1   IP Address: 2001:1001::1, AS: 63753 (IBGP), RouterID: 1.0.0.1, VRF: default-vrf
       Description: SWD-2
    State: ESTABLISHED, Time: 0h47m50s, KeepAliveTime: 60, HoldTime: 180
       KeepAliveTimer Expire in 26 seconds, HoldTimer Expire in 168 seconds
    Minimal Route Advertisement Interval: 0 seconds
       MD5 Password: $Qj0tZHMlXC1vbjYt
       UpdateSource: Loopback 1
       NextHopSelf: yes
       RefreshCapability: Received
       GracefulRestartCapability: Received
           Restart Time 120 sec, Restart bit 0
           afi/safi 2/1, Forwarding bit 0
       GracefulRestartCapability: Sent
           Restart Time 120 sec, Restart bit 0
           afi/safi 2/1, Forwarding bit 0
    Messages:    Open    Update  KeepAlive Notification Refresh-Req
…
```

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

There are 2 routes advertised to neighbor 2001:db8::110
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::/32    ::         1            32768  BL
         AS_PATH:
2      2001:db8::/16    ::         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:db8:400:400::2 received-routes

       There are 4 received routes from neighbor 2001:db8:400:400::2
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 m:NOT-INSTALLED-MULTIPATH
       S:SUPPRESSED F:FILTERED s:STALE
       Prefix                Next Hop             MED LocPrf Weight  Status
1      2001:db8:202:202::/64  2001:db8:400:400::2  0   400    0       BI

         AS_PATH: 65005 65010
2      2001:db8:400:400::/64  2001:db8:400:400::2  0   400    0       I

         AS_PATH: 65005 65010
```

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::/32                 ::        1          100        32768  BL
         AS_PATH:
2      2001:db8::/16                       ::        1          100        32768  BL
         AS_PATH:
```

This example shows information about BGP4+ outbound RIB routes.

BFD

Bidirectional Forwarding Detection Overview

Bidirectional Forwarding Detection (BFD) is a lightweight hello protocol, with little system overhead, used to rapidly detect link faults without overloading the system. BFD improves network performance by using hardware acceleration for sending packets, resulting in less overheard on the control CPU. BFD works by checking that a device is alive, thus providing rapid detection of the failure of a forwarding path and minimizing traffic loss. BFD can detect the failure of the forwarding plane in a sub-second time interval that is user-configurable.

When BFD is not enabled, packet loss can occur if a delay occurs in detecting that a neighboring device is not operational. Failure in the traffic's forwarding path leads to a traffic black hole, impacting the user experience for applications such as VOIP and video over IP.

Even if multiple protocols are running between the same interfaces, BFD has a single mechanism for detection. Applications that are supported for BFD, such as OSPF, BGP, and static routes, can use BFD to check the liveliness of the next-hop for their respective routes installed in the route table manager (RTM). Without BFD, the system can easily become overloaded because such applications need to individually send and receive their hello packets. With BFD, the control protocols between the same endpoints can use a single BFD session, eliminating the requirement to configure aggressive protocol-specific timers. BFD can track the liveliness of a single-hop or multihop forwarding engine. Once a BFD session is established, BFD transmits control packets to, and receives control packets from, the session endpoint at a given interval. If BFD packets are not received from the session endpoint for a specified configurable time interval, the session is brought down and the parent application or protocol is notified of the session failure. Upon receipt of such a notification, the respective application looks for an alternate next-hop for the route and updates the RTM accordingly.

BFD can be configured for use with the following protocols or applications:

- OSPFv2
- OSPFv3
- BGP4
- BGP4+
- IP static routes

BFD is supported for the following FastIron platforms:

- ICX 7750

Micro-BFD is supported. Micro-BFD sessions are BFD sessions running on member links of the LAG. Micro-BFD sessions provide BFD with the ability to verify link continuity for every LAG member link. Micro-BFD helps to guarantee the detection of physical member link failures, as well as the overall LAG interface failure.

Multihop BFD is supported, providing subsecond forwarding failure detection for a destination with more than one hop and up to 256 hops.

BFD Session States

The following are states through which a BFD session can move:

- Admin down
- Down
- Init
- Up

The following are states through which a BFD session normally moves:

- Down
- Init
- Up

Each system communicates its session state in the State field in the BFD control packet. This remotely received session state, in combination with the local session state, updates the state machine for that session.

Once BFD session information is passed to the micro-controller, the micro-controller can establish the BFD session and maintain the state machines on its own, tracking the session for any state changes and notifying BFD when necessary.

BFD State Change Notifications

When RX packets are missed for failure detect multiplier times, the BFD state machine detects session failure and BFD registers a callback. BFD then does a session lookup using the session ID, and marks the session state as down. All user applications associated with this BFD session are notified about the state change through an IPC message. If the holdover timer is configured for an application or protocol, it waits for the configured interval to check if the BFD session comes back up. If the BFD session comes back up before the configured time interval per application times out, the session state is updated and the holdover timer is stopped. If the BFD session is not back up before the configured interval times out, each respective application is notified about the session failure. Refer to Holdover Timer on page 350 for more information.

BFD Session Removal

When an application or protocol notifies BFD to remove a session that it is no longer using, the session is searched based on the parameters provided by the application. The application or protocol is then removed from the BFD session. If no other application or protocol is using the BFD session, the remote peer is notified about the admin down state and the admin down state is transmitted to the peer using the control packet. The BFD session is then removed from the micro-controller. The remote peer receives the control packet and brings the session down, notifying the application or protocol.

BFD in a Stacking System

The configuration of a BFD session occurs on the unit where the TX port, or next-hop, resides. The TX port is known once the ARP, or the next-hop in the case of a multihop BFD session, is resolved.

BFD operates in a stacking system in the following manner:

1. An application or protocol notifies BFD about the initiation of a session.
2. The appropriate MAC address and physical port information is gathered.
3. An IPC message is sent from the active unit to the remote unit to configure the BFD session.
4. Upon receipt of the IPC message, the standby unit stores the session locally.
5. If the TX port is a local unit port, the source host IP programming is moved from the ALPM to the L3_Entry table. The BFD session is then configured in the micro-controller.
6. The BFD enable bit is then set in the respective L3_Entry table and the Layer 2 entry pointing to the session my_discriminator value in the local unit is also programmed.

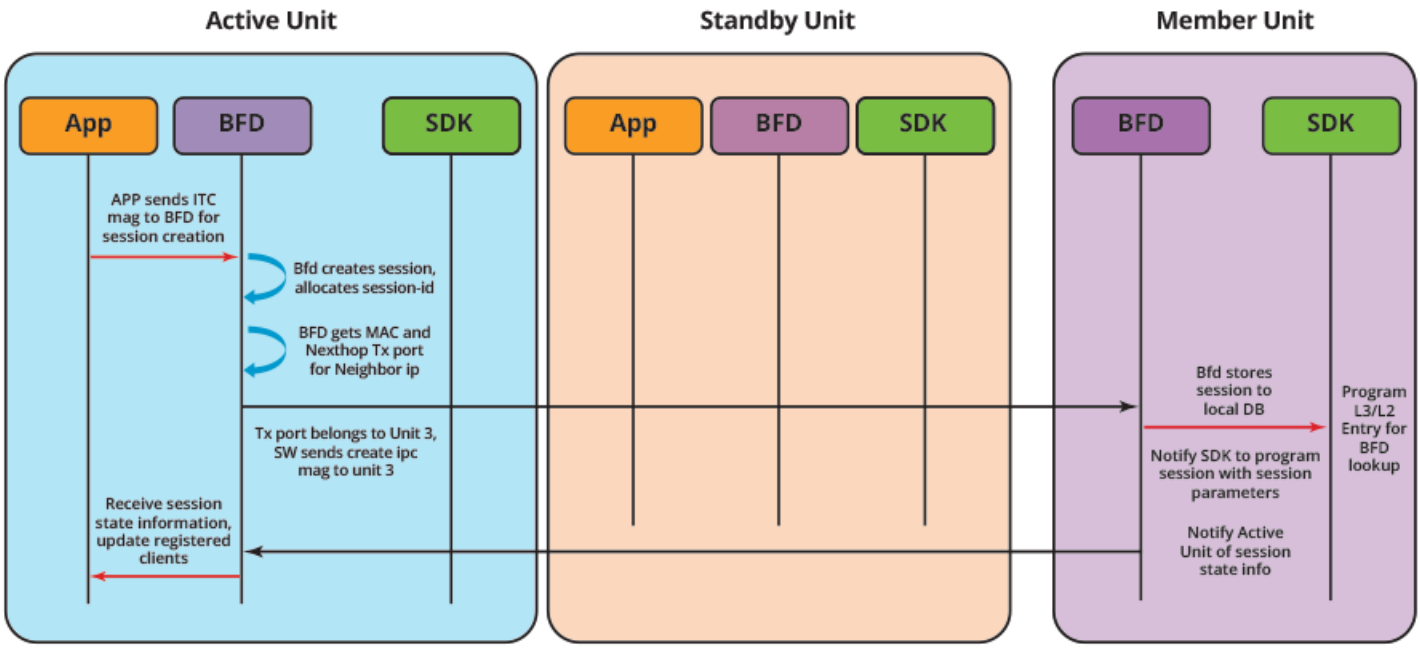
The active unit can program the BFD sessions on any remote unit. A BFD session is programmed only on the micro-controller of the remote unit ,and BFD packet exchange with the peer occurs from the micro-controller of the remote unit.

Any state change for the BFD session in the micro-controller invokes the softwarae callback within the unit. If this unit is not the active unit, the standby unit relays the event to the active unit through IPC messages. On receipt of such a notification, the active unit updates the state locally and notifies all applications.

BFD High Availability Support

BFD is deployed in the stacking setup where units take up active, standby, and member roles, as illustrated in the following figure. When a failover occurs and the active unit fails or goes through a switchover, the standby unit takes over as active. Therefore, BFD-enabled routes are available for forwarding both during and post switchover.

FIGURE 32 BFD HA Support



NOTE
BFD session packets are generated in the member unit and not on the active unit.

BFD backs up and syncs session information from the active unit to the standby unit in the following cases:

- An application initiates a new BFD session.
- An existing BFD session is changed or updated.
- A new application is configured for an existing BFD session.
- An existing BFD session is unconfigured or removed.

During a switchover, BFD sessions are changed to an admin-down state, and are newly booted up post switchover. After a switchover or failover, the BFD sessions are reconciled to the new active unit. This can result in the removal of stale sessions or the moving of existing sessions to another port. BFD applications registered with BFD at the peer level will are not impacted and no re-convergence is needed.

Micro-BFD

Micro-BFD sessions are BFD sessions running on member links of the LAG. Micro-BFD sessions provide BFD with the ability to verify link continuity for every LAG member link. Micro-BFD helps to guarantee the detection of physical member link failures, as

well as the overall LAG interface failure. A single micro-BFD session runs on each member link of the LAG for every enabled address family.

Each micro-BFD session uses its own unique local discriminator values, maintains its own set of state variables, and has its own state machine.

NOTE
The timer values remain the same for all micro-BFD sessions for a LAG interface.

Micro-BFD is capable of detecting the overall LAG interface failure and performs the following tasks:

- If a micro-BFD session state for all LAG member links is down, the BFD state of the overall LAG interface is marked as down and all applications and protocols participating in the BFD session are notified.
- If a micro-BFD session comes up, the BFD state of the overall LAG interface is marked as up and all applications and protocols participating in the BFD session are notified.

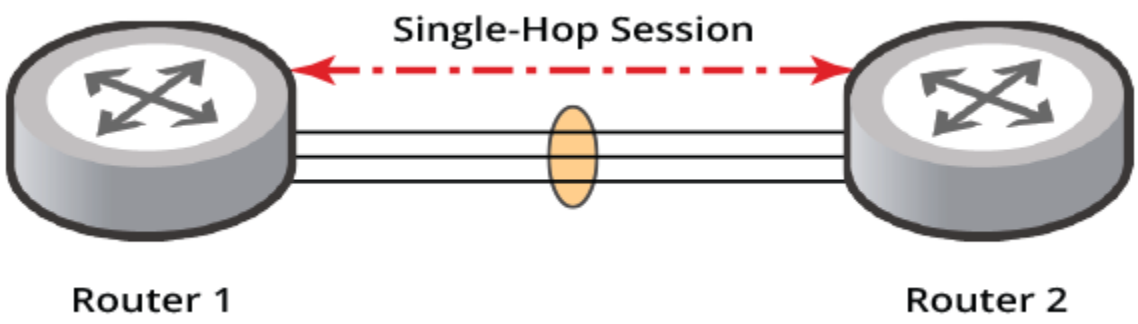
BFD session parameters, configured at the global level for a protocol or application, apply for micro-BFD sessions.

NOTE
A maximum number of 100 micro-BFD sessions are supported.

NOTE
Micro-BFD and Multi-hop-BFD cannot be enabled at the same time. Enabling micro-BFD automatically disables multihop for BFD. Switching from one mode to another requires a system reboot.

To configure micro-BFD, all LAG member links must be directly connected between two session endpoints, as illustrated in the following figure.

FIGURE 33 Micro-BFD Session



For detailed configuration examples that include micro-BFD configurations, refer to BFD Configuration Examples on page 363.

Enabling Micro-BFD Globally

Micro-BFD can be enabled globally. The following task enables micro-BFD globally.

To configure micro-BFD, all LAG member links must be directly connected between two session endpoints.

NOTE
Enabling micro- bfd causes multihop BFD support to be disabled.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **micro-bfd-enable** command to enable micro-BFD at a global level.

```
device(config)# micro-bfd-enable
```

The following example enables micro-BFD at a global level.

```
device# configure terminal
device(config)# micro-bfd-enable
```

For detailed configuration examples that include micro-BFD configurations, refer to BFD Configuration Examples on page 363.

After performing this task, you must restart the device for micro-BFD to become effective at a global level.

Configuring Micro-BFD for the Member Links of a LAG Interface

Micro-BFD can be enabled on each member link of a LAG interface so that link continuity for every LAG member link can be verified. The following task configures a static LAG, adds ports to the static LAG, configures a LAG virtual interface for the static LAG, and enables micro-BFD for each member link of the LAG interface.

To configure micro-BFD, all LAG member links must be directly connected between two session endpoints.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **lag** command, specifying a LAG, with the **static** and **id** keywords and specifying a LAG ID, to configure a static LAG.

```
device(config)# lag red static id 55
```

3. Enter the **ports** command with the **ethernet** and **to** keywords, specifying a range of Ethernet interfaces and specific Ethernet interfaces, to add ports to the LAG.

```
device(config-lag-red)# ports ethernet 1/1/1 to 1/1/4 ethernet 1/1/5 ethernet 1/1/6
```

4. Enter the **exit** command to return to global configuration mode.

```
device(config-lag-red)# exit
```

5. Enter the **interface** command with the **lag** keyword and specify an interface to configure a LAG virtual interface for the static LAG.

```
device(config)# interface lag 55
```

6. Enter the **bfd per-link** command to enable micro-BFD for each member link of the LAG interface.

```
device(config-lag-if-lg55)# bfd per-link
```

The following example configures a static LAG, adds ports to the static LAG, configures a LAG virtual interface for the static LAG, and enables micro-BFD for each member link of the LAG interface.

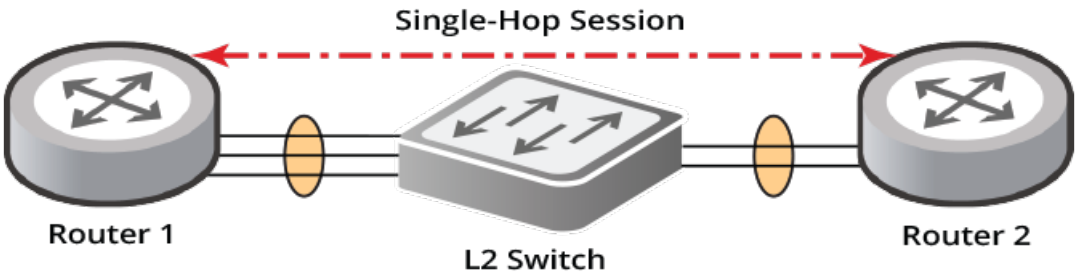
```
device# configure terminal
device(config)# lag red static id 55
device(config-lag-red)# ports ethernet 1/1/1 to 1/1/4 ethernet 1/1/5 ethernet 1/1/6
device(config-lag-red)# exit
device(config)# interface lag 55
device(config-lag-if-lg55)# bfd per-link
```

For detailed configuration examples that include micro-BFD configurations, refer to BFD Configuration Examples on page 363.

BFD Configuration on One LAG Member Port

The following figure shows two session endpoints that are connected through a Layer 2 switch using two separate LAGs. Because the LAG member links from R1 are not directly connected to R2, micro-BFD sessions cannot be configured on either of the LAG interfaces on R1 or R2.

FIGURE 34 Session Endpoints Connected Using Separate LAGS



If all LAG member links are not directly connected between two session endpoints, the following situations apply:

- The BFD session can be configured on one of the member ports of the LAG. Applications are then notified based on the status of the BFD session for this member port alone. If the BFD session running on the member port goes down, applications are notified to act on the link failure even though the LAG remains operational. To avoid this, the holdover timer can be set to a higher time interval so that the session can attempt to transition to another operational LAG member port.

For a detailed configuration example of a single-hop static BFD session for one member link for a LAG interface, refer to Configuration Example: Single-Hop Session for One Member Link for a LAG on page 364.

BFD Session Configuration Where Next-Hop is a VE Interface

BFD session configurations in the micro controller use only a local port number as the TX port parameter. When the TX port, or nexthop, for the BFD session is part of a VE interface with a physical port, the session is configured on one port for the VLAN. ARP resolution for the peer allocates the actual outgoing port to use.

If a change occurs to the Layer 2 state of the VLAN, for example a spanning tree state change, the next-hop port can be brought down, triggering a session down notification to BFD. Applications are then notified to act on the session failure even though the VE is still operational. To delay this, the holdover timer can be set to a higher time interval during which the session can come back up due to the successful movement to another VE member port. Refer to Holdover Timer on page 350, and to the *Ruckus FastIron Command Reference*, for more information.

NOTE
Applications running at the remote endpoint of the session also require the exact configuration of the holdover timer.

To view a sample configuration for a BFD single-hop session that is configured for OSPFv2 for a Virtual Ethernet (VE) interface on a LAG, refer to Configuration Example: BFD for OSPF Single-Hop Session for a VE Interface on page 367.

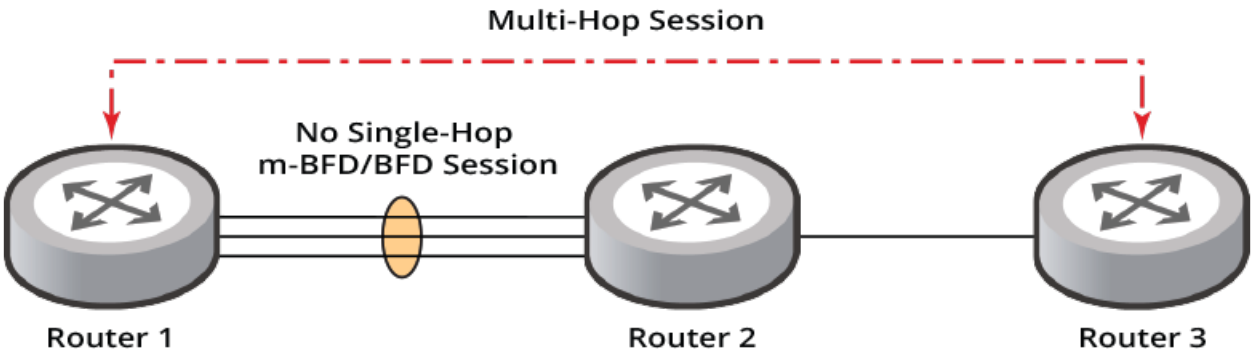
BFD Session Configuration Where Next-Hop is a LAG Interface

When the TX port, or next-hop, on which a BFD session is to be configured is a LAG member, the BFD session is configured on one of the member ports of the LAG. Applications are notified about the session based on the session status of this member port alone. If the BFD session on this member port goes down, applications are then notified to act on the link failure even though the LAG remains operational. To postpone this event, the holdover timer can be set to a higher time interval during which the session can come back up due to the successful movement to another operational LAG member port. Refer to Holdover Timer on page 350, and to the *Ruckus FastIron Command Reference*, for more information. With this configuration, if a micro-BFD session down notification for a LAG member port is received, the multihop BFD session is moved to the next available LAG member port.

For more information on micro-BFD refer to Micro-BFD on page 344.

The following figure shows a multihop BFD session running over a LAG interface with no underlying single-hop micro-BFD session also running on the LAG interface. With such a configuration, when a multihop session down notification or a LAG member port down notification is received, the multihop BFD session is moved to the next available LAG member port.

FIGURE 35 Multihop BFD session over a LAG interface



BFD Considerations and Limitations

Consider the following when configuring BFD:

- BFD asynchronous mode, which is based on the sending of BFD control packets between two systems to activate and maintain BFD neighbor sessions between devices, is supported. This means that for a BFD session to be created, BFD must be configured on both BFD peers.
- BFD is not supported if the router is running an Enabling Unicast Reverse Path Forwarding (uRPF) configuration.
- If a LAG port goes down, an attempt is made to keep the BFD session active by moving the BFD session to the next available LAG member port so that the control path is not affected.
- A maximum of 256 BFD sessions are supported.
- A maximum number of 100 micro-BFD sessions are supported.
- BFD supports both single-hop and multihop sessions.
- BFD is not supported for tunnel and management interfaces.
- Micro-BFD and Multi-hop-BFD cannot be enabled at the same time. Enabling micro-BFD automatically disables multihop for BFD. Switching from one mode to another requires a system reboot.
- BFD sessions cannot be formed with overlapping addresses across VRFs.
- BFD runs on data ports.
- If ARP is not resolved for a neighbor when a BFD session is being set up, the ARP resolution process is initiated. Once ARP is resolved, the micro-controller is notified and the process of initiating a BFD session begins. If there is any change in the ARP, the BFD session is modified. When any ARP entry in use by a BFD session ages out, all respective BFD sessions are torn down. Therefore, the ARP application continually refreshes the ARP entries without aging them out as long as the next-hop is reachable.
- Multi-VRF is supported. For multihop sessions, the peer must be reached only through the VRF associated with the BFD session.
- BFD session creation across multiple VRFs is not supported for overlapping IP addresses.
- If Graceful Restart (GR) is configured, it is recommended to configure the holdover timer for all applications so that a peer does not immediately process a session failure notification, but instead waits for a specified time period in case a restarting router comes up. Refer to Holdover Timer on page 350 for more information.
- If multiple applications initiate the same BFD session with a separate set of configuration parameters, the minimum value of the configuration parameters is considered to program the session by the micro-controller.

- If one application configures a BFD session as active while a different application configures the same BFD session as passive, the BFD session will be programmed as passive by the micro-controller.
- After BFD session information is passed to the micro-controller, the micro-controller can establish the BFD session and maintain the state machines on its own, tracking the session for any state changes and notifying BFD when necessary.
- BFD is only supported for symmetric routing.
- Micro-BFD should always be used for LAG interfaces.
- OSPFv3 BFD sessions across overlapping link local addresses is not supported. For example, if an OSPFv3 BFD session is established on a VE between fe80::1 and fe80::100, where fe80::1 is the local IP and fe80::100 is the IP on the peer. OSPFv3 BFD sessions on another VE will not come up between fe80::1 and fe80::100. The link local address on the remote peer must be changed to a different value for OSPFv3 BFD sessions to come up.
- For multi-unit LAGs, it is recommended to use Micro-BFD. Multi-hop BFD is not supported if micro-BFD is enabled.
- BFD is supported for ICX 7750 platforms.
- In a stack setup, a BFD session is programmed on the unit where the tx-port is present. Incoming BFD packets must be received on the same unit.
- BFD IPv6 sessions are not allowed for the IPv6 address 0xFC00::/7 as this is a reserved address.
- BFD configuration on SPX and MCT setup is not supported but the configuration is not restricted.

Holdover Timer

The holdover time interval for a BFD session is configured at the application or protocol level. An application or protocol pushes the configured holdover time interval to the BFD controller that keeps track of the holdover time per application or protocol per session.

The holdover timer per application or protocol per session is started whenever there is a session failure notification from the micro controller. If the BFD session comes back up before the configured holdover time interval times out, the session state is updated and the timer for the applications or protocols using the BFD session is stopped. The BFD session then resumes. If the BFD session is not back up within the configured time interval, the respective applications or protocols are notified about the session failure. Applications or protocols running at the remote endpoint for the BFD session also require the exact configuration of the holdover timer if the remote device has a similar limitation.

For more information, refer to the following commands in the *Ruckus FastIron Command Reference*:

- **bfd holdover-interval**
- **ip route bfd holdover-interval**
- **neighbor bfd**

BFD Support for OSPF

BFD support for OSPFv2 and OSPFv3 can be configured so that OSPF is a registered protocol with BFD. When BFD support for OSPF is configured, forwarding path detection failure messages are received from BFD. BFD sessions can rapidly detect link faults and notify OSPF so that it quickly responds to network topology changes. BFD support for OSPF is disabled by default.

BFD and the BFD session parameters, as well as active or passive mode, can be configured at a global VRF level and at a neighbor or peer level.

Consider the following when configuring BFD for OSPF:

- OSPF initiates a BFD session when BFD is enabled at the OSPF global or interface level. Sessions are only useful when a neighbor state is 2-Way or more. Therefore, when a neighbor state transitions to 2-Way, OSPF initiates a BFD session. Similarly, when the neighbor state transitions to less than 2-Way, OSPF tears down the BFD session.
- OSPF registration with BFD is global across all VRFs.
- When OSPF receives notification from BFD of a state change to up, OSPF marks the interface level BFD state as up and stops the holdover timer, if it has been configured.
- When OSPF receives notification from BFD of a state change to admin down, OSPF marks the interface level BFD state as down.
- When OSPF receives notification from BFD of a state change to down, the OSPF neighbor transitions to a down state, forcing the necessary route calculation for the routes that are learned from the neighbor. If the holdover timer is configured, OSPF is not immediately notified about the down state and the holdover timer begins. If the configured holdover interval expires, OSPF is notified about the session failure and marks the interface level session as down, forcing the necessary route calculation for the routes that are learned from the neighbor.

Configuring BFD for OSPFv2 Globally

BFD can be enabled globally for all OSPFv2-enabled interfaces for a VRF. The following task enables BFD globally on all OSPFv2 interfaces with default session parameters, and sets the BFD holdover interval.

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 OSPF on the device.

```
device(config)# router ospf
```

3. Enter the **bfd** command to enable BFD globally with default session parameters on OSPFv2 interfaces.

```
device(config-ospf-router)# bfd
```

4. Enter the **bfd holdover-interval** command and specify a value to set the BFD holdover interval globally.

```
device(config-ospf-router)# bfd holdover-interval 10
```

The following example enables BFD globally on all OSPFv2 interfaces with default session parameters, and sets the BFD holdover interval to 10 seconds.

```
device# configure terminal
device(config)# router ospf
device(config-ospf-router)# bfd
device(config-ospf-router)# bfd holdover-interval 10
```

Enabling BFD on an OSPFv2-enabled Interface

BFD sessions can be enabled and BFD session parameters configured for OSPFv2-enabled interfaces. The following task enables BFD for an OSPFv2-enabled Ethernet interface and sets the BFD session timer values.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **interface** command, specifying an interface.

```
device(config)# interface ethernet 1/1/1
```

3. Enter the **ip ospf bfd** command to enable BFD for the OSPFv2 interface.

```
device(config-if-e40000-1/1/1)# ip ospf bfd
```

4. Enter the **ip ospf bfd** command with the **min-tx**, **min-rx** , and **multiplier** keywords, specifying the required intervals, to configure BFD session parameters on the OSPFv2-enabled interface.

```
device(config-if-e40000-1/1/1)# ip ospf bfd min-tx 100 min-rx 300 multiplier 5
```

The following example enables BFD for an OSPFv2-enabled Ethernet interface and sets the BFD session timer values.

```
device# configure terminal
device(config)# interface ethernet 1/1/1
device(config-if-e40000-1/1/1)# ip ospf bfd
device(config-if-e40000-1/1/1)# ip ospf bfd min-tx 100 min-rx 300 multiplier 5
```

The following example enables BFD for an OSPFv2-enabled virtual Ethernet interface and sets the BFD session timer values.

```
device# configure terminal
device(config)# interface ve 25
device(config-vif-25)# ip ospf bfd
device(config-vif-25)# ip ospf bfd min-tx 100 min-rx 300 multiplier 5
```

Configuring BFD for OSPFv3 Globally

BFD can be enabled globally for all OSPFv3-enabled interfaces for a VRF. The following task enables BFD globally on all OSPFv3 interfaces with default session parameters, and sets the BFD holdover interval.

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 on the device.

```
device(config)# ipv6 router ospf
```

3. Enter the **bfd** command to enable BFD globally with default session parameters on OSPFv3 interfaces.

```
device(config-ospf6-router)# bfd
```

4. Enter the **bfd** command with the **holdover-interval** keyword, specifying a value, to set the BFD holdover interval globally.

```
device(config-ospf6-router)# bfd holdover-interval 10
```

The following example enables BFD globally on all OSPFv3 interfaces with default session parameters, and sets the BFD holdover interval to 12 seconds.

```
device# configure terminal
device(config)# ipv6 router ospf
device(config-ospf6-router)# bfd
device(config-ospf6-router)# bfd holdover-interval 12
```

Enabling BFD on an OSPFv3-enabled Interface

BFD sessions can be enabled and BFD session parameters configured for OSPFv3-enabled interfaces. The following task enables BFD for an OSPFv3-enabled Ethernet interface and sets the BFD session timer values.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **interface** command, specifying an interface.

```
device(config)# interface ethernet 1/1/2
```

3. Enter the **ipv6 ospf bfd** command to enable BFD for the OSPFv3 interface.

```
device(config-if-e40000-1/1/2)# ipv6 ospf bfd
```

4. Enter the **ipv6 ospf bfd** command with the **min-tx**, **min-rx** , and **multiplier** keywords, specifying the required intervals, to configure BFD session parameters on the OSPFv3-enabled interface.

```
device(config-if-e40000-1/1/2)# ipv6 ospf bfd min-tx 80 min-rx 85 multiplier 6
```

The following example enables BFD for an OSPFv3-enabled Ethernet interface and sets the BFD session timer values.

```
device# configure terminal
device(config)# interface ethernet 1/1/2
device(config-if-e40000-1/1/2)# ipv6 ospf bfd
device(config-if-e40000-1/1/2)# ipv6 ospf bfd min-tx 80 min-rx 85 multiplier 6
```

The following example enables BFD for an OSPFv3-enabled virtual Ethernet interface and sets the BFD session timer values.

```
device# configure terminal
device(config)# interface ve 26
device(config-vif-26)# ipv6 ospf bfd
device(config-vif-26)# ipv6 ospf bfd min-tx 120 min-rx 130 multiplier 9
```

Setting a BFD Session to Passive

BFD sessions can be set to passive for OSPFv2-enabled and OSPFv3-enabled interfaces. The following task enables BFD for an OSPFv2-enabled Ethernet interface and sets the BFD session to passive.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **interface** command, specifying an interface.

```
device(config)# interface ethernet 1/1/1
```

3. Enter the **ip ospf bfd** command to enable BFD for the OSPFv2 interface.

```
device(config-if-e40000-1/1/1)# ip ospf bfd
```

4. Enter the **ip ospf bfd** command with the **passive** keyword to set the BFD session as passive.

```
device(config-if-e40000-1/1/1)# ip ospf bfd passive
```

The following example enables BFD for an OSPFv2-enabled Ethernet interface and sets the BFD session to passive.

```
device# configure terminal
device(config)# interface ethernet 1/1/1
device(config-if-e40000-1/1/1)# ip ospf bfd
device(config-if-e40000-1/1/1)# ip ospf bfd passive
```

The following example enables BFD for an OSPFv3-enabled virtual Ethernet interface and sets the BFD session to passive.

```
device# configure terminal
device(config)# interface ve 26
device(config-vif-26)# ipv6 ospf bfd
device(config-vif-26)# ipv6 ospf bfd passive
```

Disabling BFD for OSPF-enabled Interfaces

BFD sessions can be disabled on OSPF-enabled interfaces if it has been enabled. The following task disables BFD for an OSPFv2-enabled Ethernet interface, overriding the global setting.

1. Enter the **configure terminal** command to access global configuration mode.

   ```
   device# configure terminal
   ```

2. Enter the **interface** command, specifying an interface.

   ```
   device(config)# interface ethernet 1/1/1
   ```

3. Enter the **ip ospf bfd** command with the **disable** keyword to disable BFD on the OSPF-enabled interface.

   ```
   device(config-if-e40000-1/1/1)# ip ospf bfd disable
   ```

The following example disables BFD on an OSPFv2-enabled Ethernet interface.

```
device# configure terminal
device(config)# interface ethernet 1/1/1
device(config-if-e40000-1/1/1)# ip ospf bfd disable
```

The following example disables BFD on an OSPFv3-enabled virtual Ethernet interface.

```
device# configure terminal
device(config)# interface ve 26
device(config-vif-26)# ipv6 ospf bfd disable
```

Configuring BFD for OSPFv2 Globally in a Nondefault VRF Instance

BFD can be enabled globally for all OSPFv2-enabled interfaces for a nondefault VRF instance. The following task enables BFD globally on all OSPFv2 interfaces with default session parameters, and sets the BFD holdover interval, for a nondefault VRF instance.

1. Enter the **configure terminal** command to access global configuration mode.

   ```
   device# configure terminal
   ```

2. Enter the **router ospf** command with the **vrf** keyword, specifying a VRF, to enable OSPFv2 in a nondefault VRF instance.

   ```
   device(config)# router ospf vrf red
   ```

3. Enter the **bfd** command to enable BFD globally with default session parameters on OSPFv2 interfaces for the VRF instance.

```
device(config-ospf-router-vrf-red)# bfd
```

4. Enter the **bfd** command with the **holdover-interval** keyword, specifying a value, to set the BFD holdover interval globally.

```
device(config-ospf-router-vrf-red)# bfd holdover-interval 10
```

The following example enables BFD globally on all OSPFv2 interfaces with default session parameters, and sets the BFD holdover interval to 10 seconds for VRF red.

```
device# configure terminal
device(config)# router ospf vrf red
device(config-ospf-router-vrf-red)# bfd
device(config-ospf-router-vrf-red)# bfd holdover-interval 10
```

Configuring BFD for OSPFv3 Globally in a Nondefault VRF Instance

BFD can be enabled globally for all OSPFv3-enabled interfaces for a nondefault VRF instance. The following task enables BFD globally on all OSPFv3 interfaces with default session parameters, and sets the BFD holdover interval, for a nondefault VRF instance.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **ipv6 router ospf** command with the **vrf** keyword, specifying a VRF, to enable OSPFv3 in a nondefault VRF instance.

```
device(config)# ipv6 router ospf vrf red
```

3. Enter the **bfd** command to enable BFD globally with default session parameters on OSPFv3 interfaces for the VRF instance.

```
device(config-ospf6-router-vrf-red)# bfd
```

4. Enter the **bfd** command with the **holdover-interval** keyword, specifying a value, to set the BFD holdover interval globally.

```
device(config-ospf6-router-vrf-red)# bfd holdover-interval 14
```

The following example enables BFD globally on all OSPFv3 interfaces with default session parameters, and sets the BFD holdover interval to 14 seconds for VRF red.

```
device# configure terminal
device(config)# ipv6 router ospf vrf red
device(config-ospf6-router-vrf-red)# bfd
device(config-ospf6-router-vrf-red)# bfd holdover-interval 14
```

BFD Support for BGP

BFD support for BGP4 and BGP4+ can be configured so that BGP is a registered protocol with BFD, and forwarding path detection failure messages are received from BFD. BFD for BGP is disabled by default. When BFD support 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 single-hop and multihop

iBGP and eBGP sessions with IPv4 or IPv6 neighbors. BFD for BGP is supported for both the default VRF and nondefault VRF instances.

BFD and the BFD session parameters, as well as active or passive mode, can be configured at a global VRF level and at a neighbor or peer level.

Consider the following when configuring BFD for BGP:

- BGP initiates a BFD session when BFD is enabled at the BGP global or peer level and the BGP neighbor state transitions to the Established state. BGP tears down the session when the BGP neighbor state moves out of the Established state.
- When BGP receives notification from BFD of a state change to up, BGP marks the interface level BFD state as up and stops the holdover timer if it has been configured.
- When BGP receives notification from BFD of a state change to admin down, BGP marks the interface level BFD state as down.
- When BGP receives notification from BFD of a state change to down, the BGP neighbor transitions to idle, forcing the necessary route calculation for the NLRIs that are learnt from the peer. If the holdover timer is configured, BGP is not immediately notified about the down state and the holdover timer begins. If the configured holdover time interval expires, BGP is notified about the session failure and marks the interface level session as down. The BGP neighbor transitions to idle.

For detailed configuration examples that include iBGP and eBGP configurations, refer to BFD Configuration Examples on page 363.

Configuring BFD for BGP

BFD can be enabled and BFD session parameters can be set globally for BGP-enabled interfaces. The following task enables BFD globally for BGP, configures the BFD session parameters, and sets the BFD holdover interval.

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** command to enable BFD globally.

```
device(config-bgp-router)# bfd
```

4. Enter the **bfd min-tx** command with the **min-rx** and **multiplier** keywords, specifying the required intervals, to configure BFD session parameters for BGP.

```
device(config-bgp-router)# bfd min-tx 220 min-rx 330 multiplier 8
```

5. Enter the **bfd holdover-interval** command and specify a value to set the BFD holdover interval globally.

```
device(config-bgp-router)# bfd holdover-interval 10
```

The following example enables BFD globally for BGP, configures the BFD session parameters, and sets the BFD holdover interval to 10.

```
device# configure terminal
device(config)# router bgp
device(config-bgp-router)# bfd
device(config-bgp-router)# bfd min-tx 220 min-rx 330 multiplier 8
device(config-bgp-router)# bfd holdover-interval 10
```

Configuring BFD for BGP for a Nondefault VRF Instance

BFD can be enabled and BFD session parameters can be set globally for BGP-enabled interfaces for a nondefault VRF instance. The following task enables BFD globally for BGP, configures the BFD session parameters, and sets the BFD holdover interval for 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)# address-family ipv4 unicast vrf red
```

4. Enter the **bfd** command to enable BFD globally for the VRF instance.

```
device(config-bgp-ipv4u-vrf)# bfd
```

5. Enter the **bfd min-tx** command with the **min-rx** and **multiplier** keywords, specifying the required intervals, to configure BFD session parameters for BGP for the VRF instance.

```
device(config-bgp-ipv4u-vrf)# bfd min-tx 260 min-rx 265 multiplier 8
```

6. Enter the **bfd holdover-interval** command and specify a value to set the BFD holdover interval globally for the VRF instance.

```
device(config-bgp-ipv4u-vrf)# bfd holdover-interval 16
```

The following example enables BFD globally for BGP, configures the BFD session parameters, and sets the BFD holdover interval to 16 for VRF red for the IPv4 address family.

```
device# configure terminal
device(config)# router bgp
device(config)# address-family ipv4 unicast vrf red
device(config-bgp-ipv4u-vrf)# bfd
device(config-bgp-ipv4u-vrf)# bfd min-tx 260 min-rx 265 multiplier 8
device(config-bgp-ipv4u-vrf)# bfd holdover-interval 16
```

The following example enables BFD globally for BGP, configures the BFD session parameters, and sets the BFD holdover interval to 17 for VRF green for the IPv6 address family.

```
device# configure terminal
device(config)# router bgp
device(config)# address-family ipv6 unicast vrf green
device(config-bgp-ipv6u-vrf)# bfd
device(config-bgp-ipv6u-vrf)# bfd min-tx 290 min-rx 280 multiplier 6
device(config-bgp-ipv6u-vrf)# bfd holdover-interval 17
```

Enabling BFD Sessions for a BGP Neighbor

BFD sessions can be configured for BGP neighbors. The following task enables a BFD session for a BGP neighbor with the IP address 10.2.1.1 and configures the BFD session parameters.

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 bfd** command, specifying an IP address, to enable BFD sessions for the neighbor.

```
device(config-bgp-router)# neighbor 10.2.1.1 bfd
```

4. Enter the **neighbor bfd** command, specifying an IP address, with the **min-tx** , **min-rx** , and **multiplier** keywords, specifying the required intervals, to set the BFD session parameters for the BGP neighbor.

```
device(config-bgp-router)# neighbor 10.2.1.1 bfd min-tx 220 min-rx 225 multiplier 9
```

5. Enter the **neighbor bfd** command, specifying an IP address, with the **holdover-interval** keyword and specify a value to set the BFD holdover interval for the BGP neighbor.

```
device(config-bgp-router)# neighbor 10.2.1.1 bfd holdover-interval 12
```

The following example enables a BFD session for a BGP neighbor with the IP address 10.2.1.1, configures the BFD session parameters, and sets the holdover interval to 12.

```
device# configure terminal
device(config)# router bgp
device(config-bgp-router)# neighbor 10.2.1.1 bfd
device(config-bgp-router)# neighbor 10.2.1.1 bfd min-tx 220 min-rx 225 multiplier 9
device(config-bgp-router)# neighbor 10.2.1.1 bfd holdover-interval 12
```

The following example configures BFD for a BGP neighbor with the IPv6 address 2001:2018:8192::125 and sets the BFD session timer values. No holdover interval is configured, so the default of 0 will be applied.

```
device# configure terminal
device(config)# router bgp
device(config-bgp-router)# neighbor 2001:2018:8192::125 bfd
device(config-bgp-router)# neighbor 2001:2018:8192::125 bfd min-tx 201 min-rx 301 multiplier 8
```

Enabling BFD Sessions for a BGP Peer Group

BFD sessions can be configured for BGP peer groups. The following task enables a BFD session for a BGP peer group called “mypeergroup” and configures the BFD session parameters and holdover interval.

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** *peer-group name* **peer-group** command to create a BGP peer group.

```
device(config-bgp-router)# neighbor mypeergroup peer-group
```

4. Enter the **neighbor bfd** command, specifying a BGP peer group, to enable BFD sessions for the peer group.

```
device(config-bgp-router)# neighbor mypeergroup bfd
```

5. Enter the **neighbor bfd** command, specifying a BGP peer group, with the **min-tx** , **min-rx** , and **multiplier** keywords, specifying the required intervals, to set the BFD session parameters for the BGP peer group.

```
device(config-bgp-router)# neighbor mypeergroup bfd min-tx 240 min-rx 230 multiplier 9
```

6. Enter the **neighbor bfd** command, specifying a BGP peer group, with the **holdover-interval** keyword and specify a value to set the BFD holdover interval for the BGP peer group.

```
device(config-bgp-router)# neighbor mypeergroup bfd holdover-interval 7
```

The following example enables a BFD session for a BGP peer group, configures the BFD session parameters, and sets the holdover interval to 7.

```
device# configure terminal
device(config)# router bgp
device(config-bgp-router)# neighbor mypeergroup peer-group
device(config-bgp-router)# neighbor mypeergroup bfd
device(config-bgp-router)# neighbor mypeergroup bfd min-tx 240 min-rx 230 multiplier 9
device(config-bgp-router)# neighbor mypeergroup bfd holdover-interval 7
```

Enabling BFD Sessions for a BGP Neighbor for a Nondefault VRF Instance

BFD sessions can be configured for specified BGP neighbors in a nondefault VRF instance for both the IPv4 and IPv6 unicast address families The following task enables a BFD session for a BGP neighbor with the IP address 10.5.1.1 and configures the BFD session parameters for VRF instance “red”.

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-bgp-router)# 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)# address-family ipv4 unicast vrf red
```

4. Enter the **neighbor bfd** command, specifying an IP address, to enable BFD sessions for the for the neighbor in the nondefault VRF instance.

```
device(config-bgp-ipv4u-vrf)# neighbor 10.5.1.1 bfd
```

5. Enter the **neighbor bfd** command, specifying an IP address, with the **min-tx** , **min-rx** , and **multiplier** keywords, specifying the required intervals, to set the BFD session parameters for the BGP neighbor for the nondefault VRF instance.

```
device(config-bgp-ipv4u-vrf)# neighbor 10.5.1.1 bfd min-tx 325 min-rx 330 multiplier 14
```

6. Enter the **neighbor bfd** command, specifying an IP address, with the **holdover-interval** keyword and specify a value to set the BFD holdover interval for the BGP neighbor for the nondefault VRF instance.

```
device(config-bgp-ipv4u-vrf)# neighbor 10.5.1.1 bfd holdover-interval 6
```

The following example enables a BFD session for a BGP neighbor with the IP address 10.5.1.1 and configures the BFD session parameters and holdover interval for VRF instance “red”.

```
device# configure terminal
device(config)# router bgp
device(config-bgp-router)# address-family ipv4 unicast vrf red
device(config-bgp-ipv4u-vrf)# neighbor 10.5.1.1 bfd
device(config-bgp-ipv4u-vrf)# neighbor 10.5.1.1 bfd min-tx 325 min-rx 330 multiplier 14
device(config-bgp-ipv4u-vrf)# neighbor 10.5.1.1 bfd holdover-interval 6
```

The following example enables a BFD session for a BGP neighbor with the IPv6 address 2001:2018:8192::125 and sets the BFD session timer values and holdover interval for VRF instance “green”.

```
device# configure terminal
device(config)# router bgp
device(config-bgp-router)# address-family ipv6 unicast vrf green
device(config-bgp-ipv6u)# neighbor 2001:2018:8192::125 bfd
device(config-bgp-ipv6u)# neighbor 2001:2018:8192::125 bfd min-tx 310 min-rx 210 multiplier 12
device(config-bgp-ipv6u)# neighbor 2001:2018:8192::125 bfd holdover-interval 7
```

Setting a BFD Session as Passive

BFD sessions with BGP neighbors can be set to passive. The following example configures BFD for a BGP neighbor with the IP address 10.6.1.1 and sets the BFD session as passive.

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 bfd** command, specifying an IP address, to enable BFD sessions for the neighbor.

```
device(config-bgp-router)# neighbor 10.6.1.1 bfd
```

4. Enter the **neighbor bfd** command, specifying an IP address, with the **passive** keyword to set the BFD session as passive.

```
device(config-bgp-router)# neighbor 10.6.1.1 bfd passive
```

The following example configures BFD for a BGP neighbor with the IP address 10.6.1.1 and sets the BFD session as passive.

```
device# configure terminal
device(config)# router bgp
device(config-bgp-router)# neighbor 10.6.1.1 bfd
device(config-bgp-router)# neighbor 10.6.1.1 bfd passive
```

Disabling a BFD Session for a BGP Neighbor

BFD sessions can be disabled for BGP neighbors or peer groups. The following task disables BFD for a BGP neighbor with the IPv6 address 2001:2018:8192::124.

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 BGP address-family IPv6 unicast configuration mode.

```
device(config-bgp-router)# address-family ipv6 unicast
```

4. Enter the **neighbor bfd** command, specifying an IPv6 address, with the **disable** keyword to disable BFD for the BGP neighbor.

```
device(config-bgp-ipv6u)# neighbor 2001:2018:8192::124 bfd disable
```

The following example disables BFD for a BGP neighbor with the IPv6 address 2001:2018:8192::124.

```
device# configure terminal
device(config)# router bgp
device(config-bgp-router)# address-family ipv6 unicast
device(config-bgp-ipv6u)# neighbor 2001:2018:8192::124 bfd disable
```

BFD for Static Routes

Unlike routing protocols such as OSPF and BGP, static routing has no method of fault detection or peer discovery. BFD for IPv4 static routes provides rapid detection of failure in the bidirectional forwarding path between BFD peers.

When a static route is configured with the BFD option, the BFD session state for the configured next-hop is checked. If the BFD session is up, the static route is installed in the routing table manager (RTM). If the BFD session is not up, the route is not installed in the RTM. BFD can remove the associated static route from the routing table if the next-hop becomes unreachable.

When BFD is configured for a static route next-hop, the creation of a BFD session is initiated. Similarly, when BFD is unconfigured from the static route next-hop, the BFD session is torn down.

Consider the following when configuring BFD for static routes:

- When a session state changes to up, the route is added to the RTM and is used for routing traffic.
- When a session state changes to admin down, no action is taken.
- When a session state changes to down, the static route is removed from the RTM and is not used for routing traffic.
- If the holdover timer is configured, the static route is not immediately notified about the down state and the holdover timer begins. If the configured holdover timer expires and the BFD session is not up, static routes are removed from the RTM.
- A BFD enabled static route is installed in the RTM only when the BFD session is up.

For detailed configuration examples for both single-hop and multihop sessions for an IP static route, refer to BFD Configuration Examples on page 363.

Configuring BFD for an IP Static Route

BFD can be configured globally on IP static routes. The following task enables BFD and configures BFD session parameters on an IP static route where the destination IP address is 10.2.2.0/24 and the nexthop IP address is 10.0.0.2. BFD session timer values are also configured.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **ip route** command with the **bfd** keyword, specifying a destination IP address and a next-hop IP address, to configure BFD for the IP static route.

```
device(config)# ip route 10.2.2.0/24 10.0.0.2 bfd
```

3. Enter the **ip route bfd** command and specify an IP address to configure a BFD session on the IP static route.

```
device(config)# ip route bfd 10.0.0.2
```

4. Enter the **ip route bfd** command with the **min-tx**, **min-rx**, and **multiplier** keywords, specifying the required intervals, to configure BFD session parameters for the IP static route.

```
device(config)# ip route bfd 10.0.0.2 min-tx 50 min-rx 50 multiplier 3
```

5. Enter the **ip route bfd holdover-interval** command, and enter a value, to configure the BFD holdover interval for the IP static route.

```
device(config)# ip route bfd holdover-interval 10
```

The following task enables BFD and configures BFD session parameters on an IP static route where the destination IP address is 10.2.2.0/24 and the next-hop IP address is 10.0.0.2. BFD session timer values are also configured. The holdover-timer is also set.

```
device# configure terminal
device(config)# ip route 10.2.2.0/24 10.0.0.2 bfd
device(config)# ip route bfd 10.0.0.2
device(config)# ip route bfd 10.0.0.2 min-tx 50 min-rx 50 multiplier 3
device(config)# ip route bfd holdover-interval 10
```

The following example configures a BFD session for an IP static route in a nondefault VRF instance.

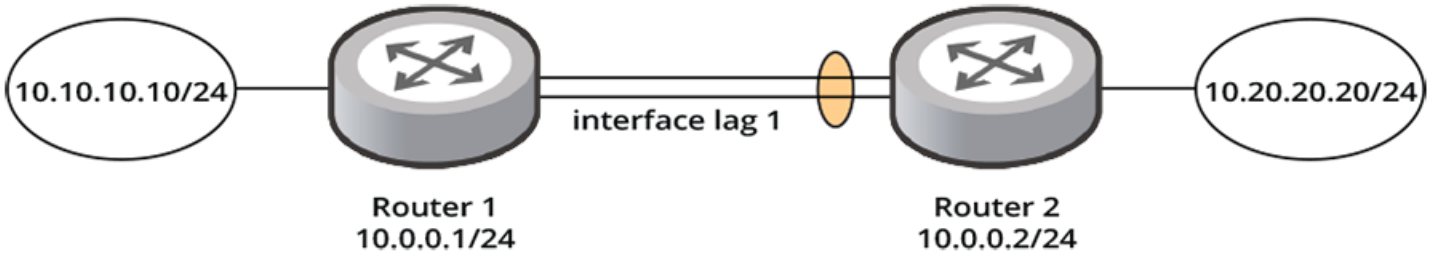
```
device# configure terminal
device(config)# ip route vrf red bfd 10.2.2.2
```

BFD Configuration Examples

Configuration Example: Single-Hop Static BFD Session for a LAG

The following figure shows a single-hop static BFD session for a LAG interface. BFD can monitor the next-hop. Micro-BFD is also configured. All LAG member links are directly connected between two session endpoints.

FIGURE 36 Single-hop Static BFD Session for a LAG



The following configuration example configures a single-hop static BFD session for a LAG interface, as illustrated in the preceding figure. Router 1 has a static route to 10.20.20.20/24 with the next-hop as 10.0.0.2/24. Router 2 has a static route to 10.10.10.10/24 with the next-hop as 10.0.0.1/24. BFD session parameters are also configured. Micro-BFD is enabled.

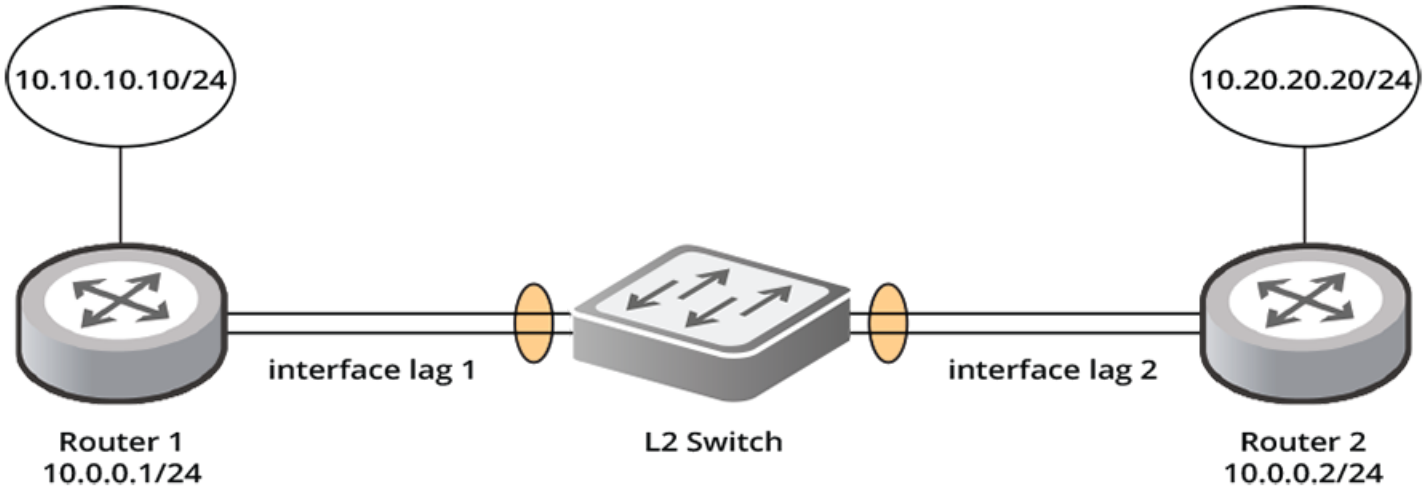
```
Router1# configure terminal
Router1(config)# interface lag 1
Router1(config-lag-if-lg1)# ip address 10.0.0.1/24
Router1(config-lag-if-lg1)# bfd per-link
Router1(config-lag-if-lg1)# exit
Router1(config)# ip route 10.20.20.20/24 10.0.0.2 bfd
Router1(config)# ip route bfd 10.0.0.2
Router1(config)# ip route bfd 10.0.0.2 min-tx 50 min-rx 50 multiplier 3

Router2# configure terminal
Router2(config)# interface lag 1
Router2(config-lag-if-lg1)# ip address 10.0.0.2/24
Router2(config-lag-if-lg1)# bfd per-link
Router2(config-lag-if-lg1)# exit
Router2(config)# ip route 10.10.10.10/24 10.0.0.1 bfd
Router2(config)# ip route bfd 10.0.0.1
Router2(config)# ip route bfd 10.0.0.1 min-tx 50 min-rx 50 multiplier 3
```

For more information and configuration examples for BFD for IP static routes, refer to BFD for Static Routes on page 361.

Configuration Example: Single-Hop Session for One Member Link for a LAG

The following figure shows a single-hop static BFD session for one member link for a LAG interface. Micro-BFD is not configured because the LAG member links are not directly connected between two session endpoints. A single BFD session is configured for the LAG interface.

FIGURE 37 Single-Hop BFD Session for One Member Link



The following configuration example configures a single-hop static BFD session for one member link for a LAG interface, as illustrated in the preceding figure. Only one BFD session is configured for the LAG on one of its member links. Router 2 is configured as active and will, therefore, self-initiate the sending of BFD packets on one of it's member links. BFD passive mode is configured for Router 1, which means that Router 1 will bootstrap the BFD session only over the member link on which it receives BFD packets from Router 2.

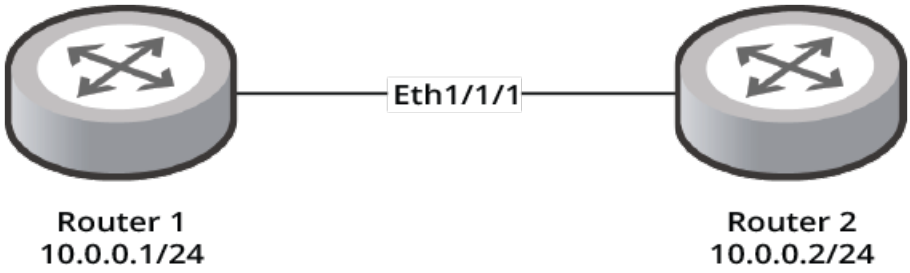
```
Router1# configure terminal
Router1(config)# interface lag 1
Router1(config-lag-if-lg1)# ip address 10.0.0.1/24
Router1(config-lag-if-lg1)# exit
Router1(config)# ip route 10.20.20.20/24 10.0.0.2 bfd
Router1(config)# ip route bfd 10.0.0.2 passive
Router1(config)# ip route bfd 10.0.0.2 min-tx 50 min-rx 50 multiplier

Router2# configure terminal
Router2(config)# interface lag 2
Router2(config-lag-if-lg2)# ip address 10.0.0.2/24
Router2(config-lag-if-lg2)# exit
Router2(config)# ip route 10.10.10.10/24 10.0.0.1 bfd
Router2(config)# ip route bfd 10.0.0.1
Router2(config)# ip route bfd 10.0.0.1 min-tx 50 min-rx 50 multiplier 3
```

For more information and configuration examples for BFD for IP static routes, refer to BFD for Static Routes on page 361.

Configuration Example: BFD for OSPF Single-Hop Session for an Ethernet Interface

The following figure shows a BFD single-hop session that is configured for OSPFv2 for Ethernet interface 1/1/1 on two devices, Router 1 and Router2.

FIGURE 38 BFD for OSPF Single-Hop Session on an Ethernet Interface



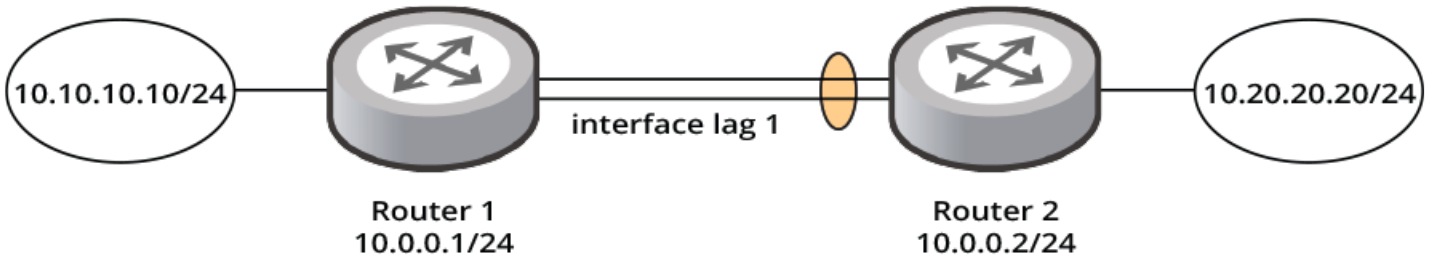
The following configuration configures a BFD single-hop session for OSPFv2, as illustrated in the preceding figure. BFD for OSPFv2 is configured for Ethernet interface 1/1/1 on both Router 1 and Router 2. The BFD session parameters are also configured.

```
Router1# configure terminal
Router1(config)# router ospf
Router1(config-ospf-router)# bfd
Router1(config-ospf-router)# bfd min-tx 50 min-rx 50 multiplier 3

Router1(config)# interface ethernet 1/1/1
Router1(config-if-e40000-1/1/1)# ip address 10.0.0.1/24
Router1(config-if-e40000-1/1/1)# ip ospf bfd
Router1(config-if-e40000-1/1/1)# ip ospf bfd passive
Router1(config-if-e40000-1/1/1)# ip ospf bfd min-tx 50 min-rx 50 multiplier 3

Router2# configure terminal
Router2(config)# router ospf
Router2(config-ospf-router)# bfd
Router2(config-ospf-router)# bfd min-tx 50 min-rx 50 multiplier 3

Router2(config)# interface ethernet 1/1/1
Router2(config-if-e40000-1/1/1)# ip address 10.0.0.2/24
Router2(config-if-e40000-1/1/1)# ip ospf bfd
Router2(config-if-e40000-1/1/1)# ip ospf bfd min-tx 50 min-rx 50 multiplier 3
```

Configuration Example: BFD for OSPF Single-Hop Session for a LAG Interface

The following figure shows a BFD single-hop session that is configured for OSPFv2 for a LAG interface on two devices, Router 1 and Router 2. Micro-BFD is also configured for the LAG interface. All LAG member links are directly connected between two session endpoints.

FIGURE 39 LAG Member Links Directly Connected



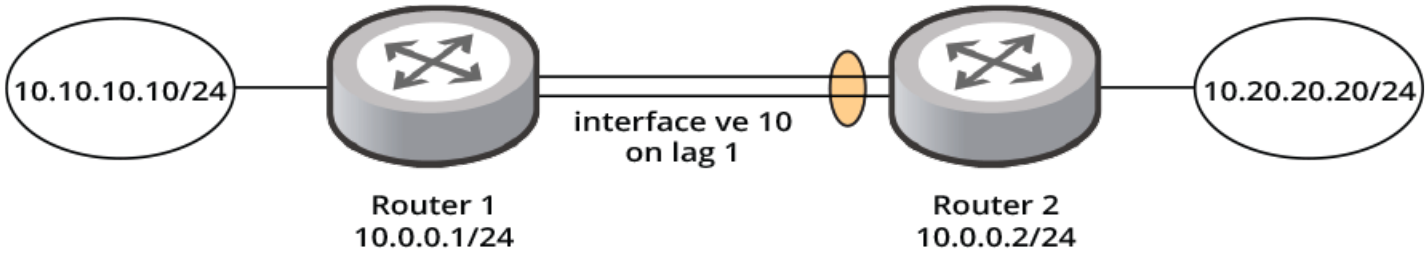
The following configuration example enables BFD for OSPFv2 for LAG interface 1 on both Router 1 and Router 2, as illustrated in the preceding figure. The BFD session parameters are configured, and micro-BFD is enabled.

```
Router1# configure terminal
Router1(config)# interface lag 1
Router1(config-lag-if-lg1)# ip address 10.0.0.1/24
Router1(config-lag-if-lg1)# bfd per-link
Router1(config-lag-if-lg1)# exit

Router1(config)# router ospf
Router1(config-ospf-router)# bfd
Router1(config-ospf-router)# bfd min-tx 50 min-rx 50 multiplier 3
Router1(config-ospf-router)# exit

Router1(config)# interface lag 1
Router1(config-lag-if-lg1)# ip address 10.0.0.1/24
Router1(config-lag-if-lg1)# bfd per-link
Router1(config-lag-if-lg1)# ip ospf bfd
Router1(config-lag-if-lg1) ip ospf bfd min-tx 50 min-rx 50 multiplier 3

Router2# configure terminal
Router2(config)# interface lag 1
Router2(config-lag-if-lg1)# ip address 10.0.0.2/24
Router2(config-lag-if-lg1)# bfd per-link
Router2(config-lag-if-lg1)# exit

Router2(config)# router ospf
Router2(config-ospf-router)# bfd
Router2(config-ospf-router)# bfd min-tx 50 min-rx 50 multiplier 3
Router2(config-ospf-router)# exit

Router2(config)# interface lag 1
Router2(config-lag-if-lg1)# ip address 10.0.0.2/24
Router2(config-lag-if-lg1)# bfd per-link
Router2(config-lag)# ip ospf bfd
Router2(config-lag)# ip ospf bfd min-tx 50 min-rx 50 multiplier 3
```

Configuration Example: BFD for OSPF Single-Hop Session for a VE Interface

The following figure shows a BFD single-hop session that is configured for OSPFv2 for a Virtual Ethernet (VE) interface on a LAG on two devices, Router 1 and Router 2. A single-hop BFD session is configured and micro-BFD is enabled.

FIGURE 40 BFD Session for a VE Interface on a LAG



The following configuration example enables BFD for OSPFv2 for a VE interface on a LAG on both Router 1 and Router 2, as illustrated in the preceding figure. The BFD session parameters are configured, and micro-BFD is enabled for the LAG interface.

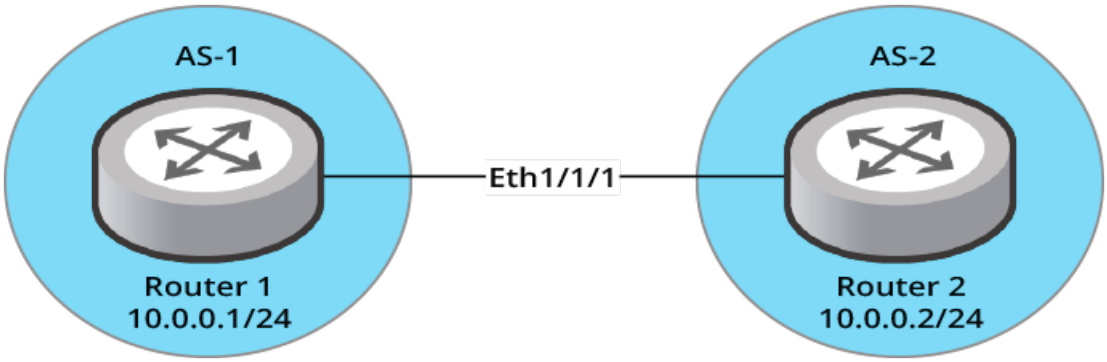
```
Router1# configure terminal
Router1(config)# vlan 10
Router1(config-vlan-10)# tag lag 1
Router1(config-vlan-10)# router-interface ve 10
Router1(config-vlan-10)# exit

Router1(config)# interface lag 1
Router1(config-lag-if-lg1)# bfd per-link
Router1(config-lag-if-lg1)# exit

Router1(config)# interface ve 10
Router1(config-vif-10)# ip address 10.0.0.1/24
Router1(config-vif-10)# exit

Router1(config)# router ospf
Router1(config-ospf-router)# bfd
Router1(config-ospf-router)# bfd min-tx 50 min-rx 50 multiplier 3
Router1(config-ospf-router)# exit

Router1(config)# interface lag 1
Router1(config-lag-if-lg1)# bfd per-link
Router1(config-lag-if-lg1)exit

Router1(config)# interface ve 10
Router1(config-vif-10)# ip address 10.0.0.1/24
Router1(config-vif-10)# ip ospf bfd
Router1(config-vif-10) ip ospf bfd min-tx 50 min-rx 50 multiplier 3



Router2# configure terminal
Router2(config)# vlan 10
Router2(config-vlan-10)# tag lag 1
Router2(config-vlan-10)# router-interface ve 10
Router2(config-vlan-10)# exit

Router2(config)# interface lag 1
Router2(config-lag-if-lg1)# bfd per-link
Router2(config-lag-if-lg1)# exit

Router2(config)# interface ve 10
Router2(config-vif-10)# ip address 10.0.0.2/24
```

```
Router2(config-vif-10)# exit

Router2(config)# router ospf
Router2(config-ospf-router)# bfd
Router2(config-ospf-router)# bfd min-tx 50 min-rx 50 multiplier 3
Router2(config-ospf-router)# exit

Router2(config)# interface lag 1
Router2(config-lag-if-lg1)# bfd per-link
Router2(config-lag-if-lg1)exit

Router2(config)# interface ve 10
Router2(config-vif-10)# ip address 10.0.0.2/24
Router2(config-vif-10)# ip ospf bfd
Router2(config-vif-10) ip ospf bfd min-tx 50 min-rx 50 multiplier 3
```

Configuration Example: BFD for eBGP Single-Hop Session

The following figure shows a BFD single-hop session that is configured for eBGP for Ethernet interface 1/1/1 on two devices, Router 1 and Router 2.

FIGURE 41 BFD for eBGP Single-Hop Session



The following configuration example enables BFD for eBGP for Ethernet interface 1/1/1 on Router 1 globally. BFD session parameters are also configured.

```
Router1# configure terminal
Router1(config)# interface ethernet 1/1/1
Router1(config-if-e40000-1/1/1)# ip address 10.0.0.1/24
Router1(config-if-e40000-1/1/1)# exit
Router1(config)# router bgp
Router1(config-bgp-router)# local-as 1
Router1(config-bgp-router)# bfd
Router1(config-bgp-router)# bfd min-tx 50 min-rx 50 multiplier 3
Router1(config-bgp-router)# neighbor 10.0.0.2 remote-as 1
```

The following configuration example enables BFD for eBGP for a BGP peer on Router 1. BFD session parameters are also configured.

```
Router1# configure terminal
Router1(config)# router bgp
Router1(config-bgp-router)# local-as 1
Router1(config-bgp-router)# neighbor 10.0.0.2 remote-as 2
Router1(config-bgp-router)# neighbor 10.0.0.2 bfd
Router1(config-bgp-router)# neighbor 10.0.0.2 bfd min-tx 50 min-rx 50 multiplier 3
```

The following configuration example enables BFD for a BGP peer group on Router 1. BFD session parameters are also configured.

```
Router1# configure terminal
Router1(config)# router bgp
Router1(config-bgp-router)# local-as 1
Router1(config-bgp-router)# neighbor pg peer-group
Router1(config-bgp-router)# neighbor pg remote-as 2
Router1(config-bgp-router)# neighbor pg bfd
Router1(config-bgp-router)# neighbor pg bfd min-tx 50 min-rx 50 multiplier 3
Router1(config-bgp-router)# neighbor 10.0.0.2 pg
```

The following configuration example enables BFD for eBGP for Ethernet interface 1/1/1 on Router 2 globally. BFD session parameters are also configured.

```
Router2# configure terminal
Router2(config)# interface ethernet 1/1/1
Router2(config-if-e40000-1/1/1)# ip address 10.0.0.2/24
Router2(config-if-e40000-1/1/1)# exit
Router2(config)# router bgp
Router2(config-bgp-router)# local-as 2
Router2(config-bgp-router)# bfd
Router2(config-bgp-router)# bfd min-tx 50 min-rx 50 multiplier 3
Router2(config-bgp-router)# neighbor 10.0.0.1 remote-as 1
```

The following configuration example enables BFD for eBGP for a BGP peer on Router 2. BFD session parameters are also configured.

```
Router2# configure terminal
Router2(config)# router bgp
Router2(config-bgp-router)# local-as 2
Router2(config-bgp-router)# neighbor 10.0.0.1 remote-as 1
Router2(config-bgp-router)# neighbor 10.0.0.1 bfd
Router2(config-bgp-router)# neighbor 10.0.0.1 bfd min-tx 50 min-rx 50 multiplier 3
```

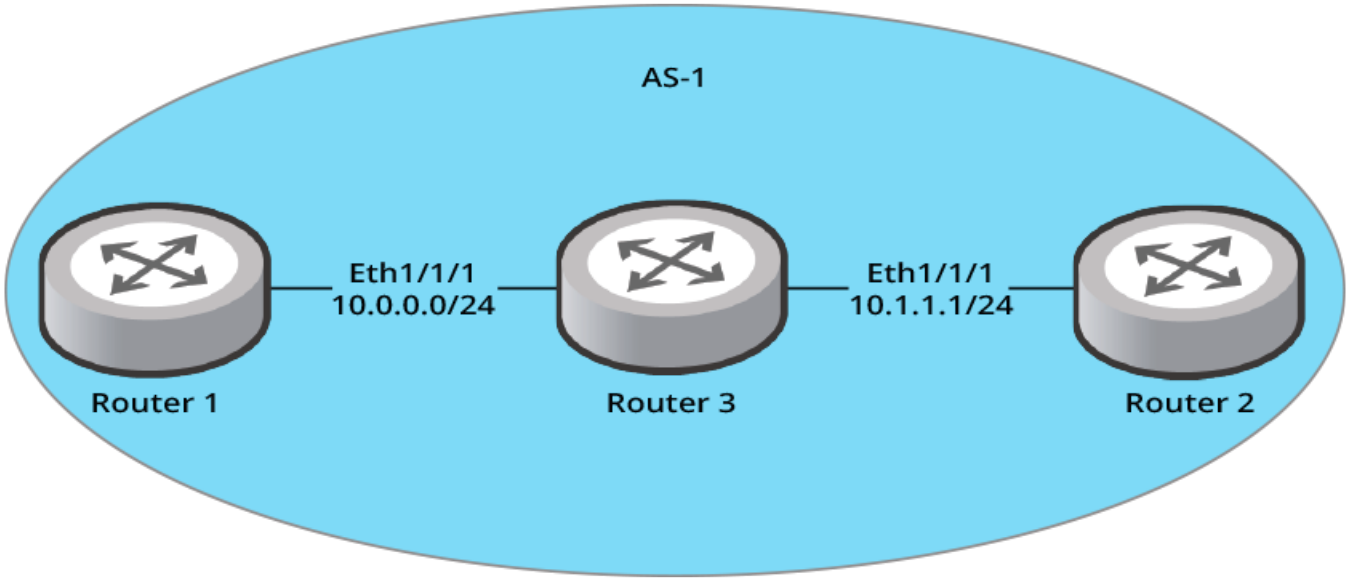
The following configuration example enables BFD for a BGP peer group on Router 2. BFD session parameters are also configured.

```
Router2# configure terminal
Router2(config)# router bgp
Router2(config-bgp-router)# local-as 2
Router2(config-bgp-router)# neighbor pg peer-group
Router2(config-bgp-router)# neighbor pg remote-as 1
Router2(config-bgp-router)# neighbor pg bfd
Router2(config-bgp-router)# neighbor pg bfd min-tx 50 min-rx 50 multiplier 3
Router2(config-bgp-router)# neighbor 10.0.0.1 pg
```

Configuration Example: BFD for iBGP Single-Hop Session

The following figure shows a BFD session configured for iBGP for Ethernet interface 1/1/1 on two devices, Router 1 and Router 2.

FIGURE 42 BFD for iBGP Single-Hop Session



The following configuration example enables BFD for iBGP for Ethernet interface 1/1/1 on Router 1 globally. BFD session parameters are also configured.

```
Router1# configure terminal
Router1(config)# interface ethernet 1/1/1
Router1(config-if-e40000-1/1/1)# ip address 10.0.0.0/24
Router1(config-if-e40000-1/1/1)# exit
Router1(config)# router bgp
Router1(config-bgp-router)# local-as 1
Router1(config-bgp-router)# bfd
Router1(config-bgp-router)# bfd min-tx 50 min-rx 50 multiplier 3
Router1(config-bgp-router)# neighbor 10.1.1.1 remote-as 1
```

The following configuration example enables BFD for iBGP for a BGP peer on Router 1. BFD session parameters are also configured.

```
Router1# configure terminal
Router1(config)# router bgp
Router1(config-bgp-router)# local-as 1
Router1(config-bgp-router)# neighbor 10.1.1.1 remote-as 1
Router1(config-bgp-router)# neighbor 10.1.1.1 bfd
Router1(config-bgp-router)# neighbor 10.1.1.1 bfd min-tx 50 min-rx 50 multiplier 3
```

The following configuration example enables BFD for a BGP peer group on Router 1. BFD session parameters are also configured.

```
Router1# configure terminal
Router1(config)# router bgp
Router1(config-bgp-router)# local-as 1
Router1(config-bgp-router)# neighbor pg1 peer-group
Router1(config-bgp-router)# neighbor pg1 remote-as 1
Router1(config-bgp-router)# neighbor pg bfd
Router1(config-bgp-router)# neighbor pg1 bfd min-tx 50 min-rx 50 multiplier 3
Router1(config-bgp-router)# neighbor 10.1.1.1 pg
```

The following configuration example enables BFD for iBGP for Ethernet interface 1/1/1 on Router 2 globally. BFD session parameters are also configured.

```
Router2# configure terminal
Router2(config)# interface ethernet 1/1/1
Router2(config-if-e40000-1/1/1)# ip address 10.1.1.1/24
Router2(config-if-e40000-1/1/1)# exit
Router2(config)# router bgp
Router2(config-bgp-router)# local-as 1
Router2(config-bgp-router)# bfd
Router2(config-bgp-router)# bfd min-tx 50 min-rx 50 multiplier 3
Router2(config-bgp-router)# neighbor 10.0.0.0 remote-as 1
```

The following configuration example enables BFD for iBGP for a BGP peer on Router 2. BFD session parameters are also configured.

```
Router2# configure terminal
Router2(config)# router bgp
Router2(config-bgp-router)# local-as 1
Router2(config-bgp-router)# neighbor 10.0.0.0 remote-as 1
Router2(config-bgp-router)# neighbor 10.0.0.0 bfd
Router2(config-bgp-router)# neighbor 10.0.0.0 bfd min-tx 50 min-rx 50 multiplier 3
```

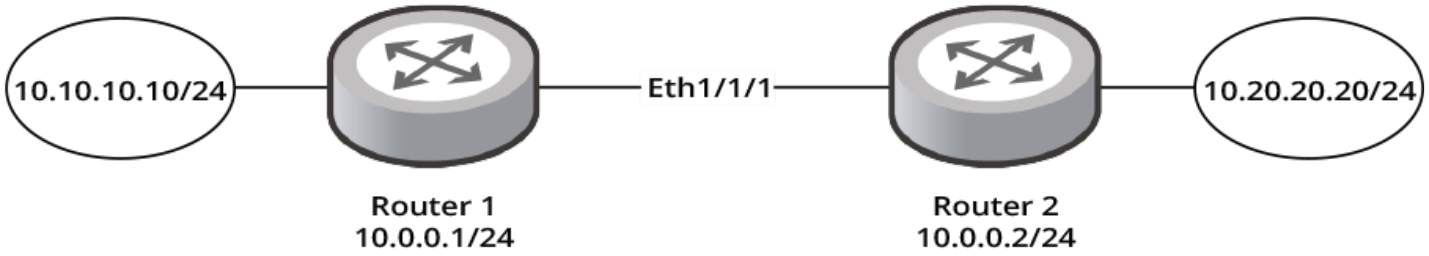
The following configuration example enables BFD for a BGP peer group on Router 2. BFD session parameters are also configured.

```
Router2# configure terminal
Router2(config)# router bgp
Router2(config-bgp-router)# local-as 1
Router2(config-bgp-router)# neighbor pg1 peer-group
Router2(config-bgp-router)# neighbor pg1 remote-as 1
Router2(config-bgp-router)# neighbor pg1 bfd
Router2(config-bgp-router)# neighbor pg1 bfd min-tx 50 min-rx 50 multiplier 3
Router2(config-bgp-router)# neighbor 10.0.0.0 pg1
```

Configuration Example: BFD for IP Static Routes Single-Hop Session

The following figure shows a single-hop static BFD session. BFD can monitor the next-hop.

FIGURE 43 Single-Hop Static BFD Session



The following configuration example configures a single-hop static BFD session as illustrated in the preceding figure. Router 1 has a static route to 10.20.20.20/24 with the next-hop as 10.0.0.2/24. Router 2 has a static route to 10.10.10.10/24 with the next-hop as 10.0.0.1/24. BFD session parameters are also configured, and the holdover timer is set.

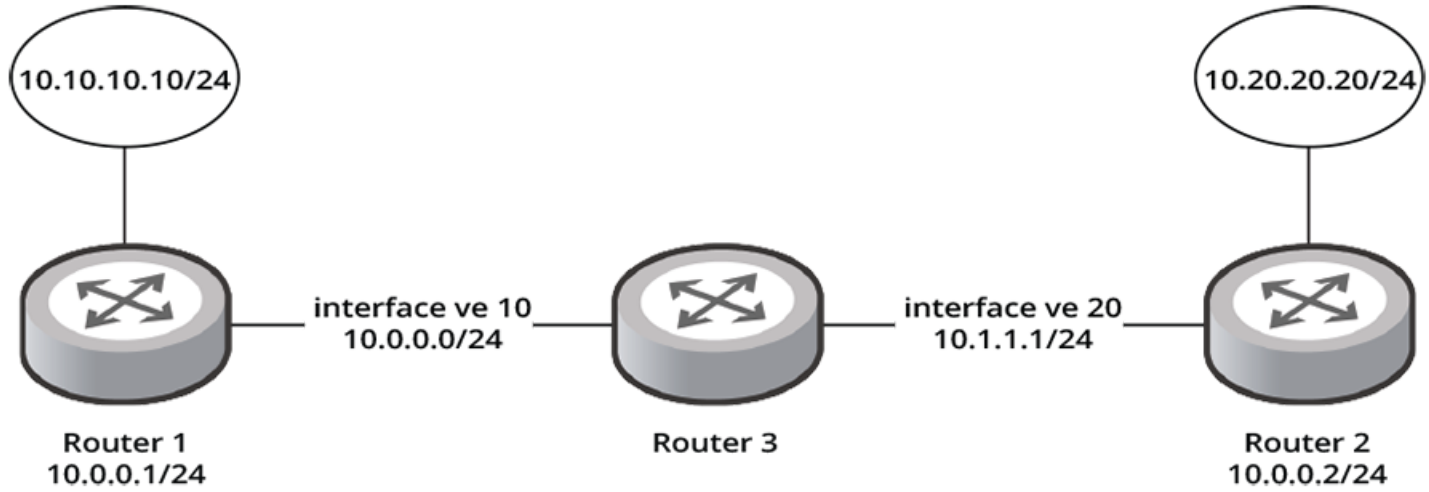
```
Router1# configure terminal
Router1(config)# interface ethernet 1/1/1
Router1(config-if-e40000-1/1/1)# ip address 10.0.0.1/24
Router1(config-if-e40000-1/1/1)# exit
Router1(config)# ip route 10.20.20.20/24 10.0.0.2 bfd
Router1(config)# ip route bfd 10.0.0.2
```

```
Router1(config)# ip route bfd 10.0.0.2 min-tx 50 min-rx 50 multiplier 3
Router1(config)# ip route bfd holdover-interval 40

Router2# configure terminal
Router2(config)# interface ethernet 1/1/1
Router2(config-if-e40000-1/1/1)# ip address 10.0.0.2/24
Router2(config-if-e40000-1/1/1)# exit
Router2(config)# ip route 10.10.10.10/24 10.0.0.1 bfd
Router2(config)# ip route bfd 10.0.0.1
Router2(config)# ip route bfd 10.0.0.1 min-tx 50 min-rx 50 multiplier 3
Router2(config)# ip route bfd holdover-interval 40
```

Configuration Example: BFD for IP Static Routes Multihop Session

The following figure shows a multihop static BFD session.

FIGURE 44 Multihop Static BFD Session



The following configuration example configures a multihop static BFD session as illustrated in the preceding figure. Router 1 has a static route to 10.20.20.20/24 with the next-hop as 10.0.0.2/24. A multihop BFD session is configured with 10.0.0.1 specified as the local source IP address to use for the multihop BFD session. BFD session parameters are also configured. Router 2 has a static route to 10.10.10.10/24 with the next-hop as 10.0.0.1. A multihop BFD session is configured with 10.0.0.2 specified as the local source IP address to use for the multihop BFD session. BFD session parameters are also configured, and the BFD holdover timer is set.

```
Router1# configure terminal
Router1(config)# interface ve 10
Router1(config-vif-10)# ip address 10.0.0.1/24
Router1(config-vif-10)# exit
Router1(config)# ip route 10.20.20.20/24 10.0.0.2 bfd
Router1(config)# ip route bfd 10.0.0.2 multi-hop 10.0.0.1
Router1(config)# ip route bfd 10.0.0.2 multi-hop 10.0.0.1 min-tx 50 min-rx 50 multiplier 3
Router1(config)# ip route bfd holdover-interval 35


Router2# configure terminal
Router2(config)# interface ve 20
Router2(config-vif-20)# ip address 10.0.0.2/24
Router2(config-vif-20)# exit
Router2(config)# ip route 10.10.10.10/24 10.0.0.1 bfd
Router2(config)# ip route bfd 10.0.0.1 multi-hop 10.0.0.2
```

```
Router2(config)# ip route bfd 10.0.0.1 multi-hop 10.0.0.2 min-tx 50 min-rx 50 multiplier 3
Router2(config)# ip route bfd holdover-interval 35
```

Displaying BFD Information

Various show commands can display statistical information about BFD configurations.

Use one or more of the following commands to verify BFD information. Using these commands is optional, and the variations of the commands can be entered in any order. For full documentation of all the show commands for BFD, refer to the *Ruckus FastIron Command Reference*.

1. Enter the **show bfd** command to display general BFD information.

```
device> show bfd

 BFD State: ENABLE    Version: 1
 Sessions      : Total count: 2    Max Allowed: 256    Max Reach Count: 0
 Micro Sessions: Total count: 0    Max Allowed: 0    Max Reach Count: 0
 Micro-bfd: oper-state:  disabled,  config-state:  disabled
 Protocols Registration Count: 5
     bgp/default-vrf  ospf/white  static/default-vrf  static/white  ospf/default-vrf

 Number of Interfaces with controller configuration: 2
    Nhop-Intf   Sessions-count  M-BFD[Lag per-link]
    ---------   --------------  --------------------
    1/1/13      1               no
    1/1/14      1               no
```

2. Enter the **show bfd applications** command to display information about the applications that are registered to BFD.

```
device> show bfd applications

  Registered Protocols Count: 2
     Protocol  VRFID     Parameter HoldoverInterval(ms)
     bgp       0         0            0
     ospf      0         0            0
```

3. Enter the **show bfd neighbors** command and specify an IP address to display BFD information about a specific IPv4 neighbor.

```
device> show bfd neighbors 20.20.20.1

Flags: H:Hop [S:Single/M:Multi] P:Passive-Mode M:Micro-BFD R:RxRemote [Y:Yes/N:No]
MinTx/MinRx/NegTx/NegRx/DTime [milliseconds]  HoldTime [seconds]

Id  Neighbor          Local             State  NHopIntf   NegRx|NegTx|DTime    P/H/M/R
--  --------          -----             -----  --------   -----------------    -------
1   20.20.20.1        10.10.10.1        UP     1/1/13     300   300   900      N/M/N/Y
```

4. Enter the **show bfd v6-neighbors** command to display BFD information about IPv6 neighbors.

```
device> show bfd v6-neighbors

Id  Neighbor                                Local                                   State  NHopIntf
NegRx|NegTx|DTime   P/H/C/R
156 fe80::bfd:80:1:100:2                    fe80::bfd:80:1:100:1                    DOWN   lg100
30000 350   900000  N/S/Y/Y
```

5. Enter the **show bfd neighbors** command with the **details** keyword and specify an IPv6 address to display detailed BFD information about a specific IPv6 neighbor.

```
device> show bfd neighbors details 12::6

NeighborAddress  State  Interface    Holddown(ms)   Interval(ms)  R/H
12::6            UP     ve 10       500000         100000        Y/S
Registered Protocols(Protocol/VRFID): bgp/4
Local: Disc: 135, Diag: 3, Demand: 0 Poll: 0, Port: 49153
       MinTxInterval(ms): 100000, MinRxInterval(ms): 100000, Multiplier: 5
Remote: Disc: 215, Diag: None, Demand: 0 Poll: 0, Port: 3784
       MinTxInterval(ms): 100000, MinRxInterval(ms): 100000, Multiplier: 5
Negotiated: Neg Tx: 100000, Neg Rx: 100000 Multiplier: 5
Stats: RX: 103592 TX: 85739 SessionUpCount: 3 at SysUpTime: 215:8:17:4.674
Session Uptime:0:0:56:39.640, LastSessionDownTimestamp: 215:7:18:51.835
```

6. Enter the **show bfd neighbors bgp** command to display BFD neighbor session information for BGP.

```
device> show bfd neighbors bgp

Total Entries:2 R:RxRemote(Y:Yes/N:No)H:Hop(S:Single/M:Multi)
NeighborAddress   State  Interface    Holddown(ms)   Interval(ms)  R/H
10.12.12.6        UP     ve 10       500000         100000        Y/S
12::6             UP     ve 10       500000         100000        Y/S
```

7. Enter the **show bfd neighbors bgp** command with the **details** keyword to display detailed BFD neighbor session information for BGP.

```
device> show bfd neighbors bgp details

Total Entries:2 R:RxRemote(Y:Yes/N:No)H:Hop(S:Single/M:Multi)
NeighborAddress  State  Interface       Holddown   Interval  R/H
10.12.12.6       UP     ve 10          500000     100000    Y/S
  Registered Protocols(Protocol/VRFID): bgp/4 ospf/4 static/4
  Local: Disc: 133, Diag: None, Demand: 0 Poll: 0, Port: 49153
         MinTxInterval(ms): 100000, MinRxInterval(ms): 100000, Multiplier: 5
  Remote: Disc: 210, Diag: None, Demand: 0 Poll: 0, Port: 3784, State: UP,
         MinTxInterval(ms): 100000, MinRxInterval(ms): 100000, Multiplier: 5
  Negotiated: Neg Tx: 100000, Neg Rx: 100000, Multiplier: 5
  Stats: RX: 1242514 TX: 1082496 SessionUpCount: 1 at SysUpTime:    215:8:8:54.749
  Session Uptime: 1:3:37:16.897, LastSessionDownTimestamp: 0:0:0:0.0

NeighborAddress  State  Interface       Holddown   Interval  R/H
12::6            UP     ve 10          500000     100000    Y/S
  Registered Protocols(Protocol/VRFID): bgp/4
  Local: Disc: 135, Diag: None, Demand: 0 Poll: 0, Port: 49153
         MinTxInterval: 100000, MinRxInterval: 100000, Multiplier: 5
  Remote: Disc: 215, Diag: None, Demand: 0 Poll: 0, Port: 3784, State: UP,
          MinTxInterval: 100000, MinRxInterval: 100000, Multiplier: 5
  Negotiated: Neg Tx: 100000, Neg Rx: 100000, Multiplier: 5
  Stats: RX: 96595 TX: 80404 SessionUpCount: 3 at SysUpTime: 215:8:8:54.749
  Session Uptime: 0:0:48:29.723, LastSessionDownTimestamp: 215:7:18:51.827
```

8. Enter the **show bfd agent** command with the **ipv4** and **all** keywords to display IPv4 BFD agent-related information for all sessions.

```
device> show bfd agent sessions ipv4 all

Flags: H:Hop [S:Single/M:Multi] P:Passive-Mode M:Micro-BFD
MinTx/MinRx/Mult [milliseconds]  HoldTime [seconds]

Id  Neighbor          Local             NHopIntf   Phy-Port Lag-Port MinRx|MinTx|Mult  P/H/M
vrf
--  --------          -----             --------   -------- -------- ----------------- -----
---------
161 2.2.2.2           2.2.2.1           v2         lg9      1/1/7    300   300   3     N/S/Y default-
vrf
162 2.2.2.2           2.2.2.1           v2         lg9      1/1/10   300   300   3     N/S/Y default-
vrf

#show bfd agent info
Agent id - 1
  HW Assist      : 1       MicroBFD       : 1           Max Session    : 256
  HA Epoch       : 0
  HA in Prog     : No      HA Start time  : 0:0:0       HA End time    : in future
  UC Session     : 2       AG Session     : 2
  Tx-id          : 37      Rx-Id          : 22
```

9. Enter the **show bfd neighbors ospf** command to display neighbor session information for OSPF.

```
device> show bfd neighbors ospf

Total Entries:2 R:RxRemote(Y:Yes/N:No)H:Hop(S:Single/M:Multi)
NeighborAddress    State  Interface    Holddown(ms)   Interval(ms)  R/H
10.12.12.6         UP     ve 10        500000         100000        Y/S
10.16.16.6         UP     eth 1/1/1    250000         50000         Y/S
```

10. Enter the **show bfd vrf** command with the **applications** keyword, and specify a VRF to display information about the applications that are registered to BFD for a non-default VRF instance.

```
device> show bfd vrf red applications

  Protocols Registration Count for VRF [red]: 3
     Protocol  Parameter  Global-Holdtime
     ospf6     0          99
     bgp       1          88
     ospf      1          88
```

11. Enter the **show bfd ha info** command to display high availability (HA) information from active agents for BFD.

```
device> show bfd ha info
SO in Prog: No    SO-Completed from 0 agents
SO Epoch: 0
Agent Id - 1 (STANDALONE)
  Agent State         : so not prog       HA Recon state      : NONE
  tx id               : 52                Agent IPC State     : Lost
  rx id               : 202
```

12. Enter the **show bfd micro-session** command, specifying a session ID, to display information about a specific micro-BFD session.

```
device> show bfd micro-session 3

Neighbor: 2.2.2.2
Local   : 2.2.2.1
Outgoing Lag: lg9 MBfd Sessions: 2
     M-Id State  R-State  LagPort  NegRx|NegTx|DTime   R
     ---- -----  -------  -------  ------------------ -
     162  UP     UP       1/1/10   300   300   900     Y
     161  UP     UP       1/1/7    300   300   900     Y
```

VRRPv2

VRRPv2 overview

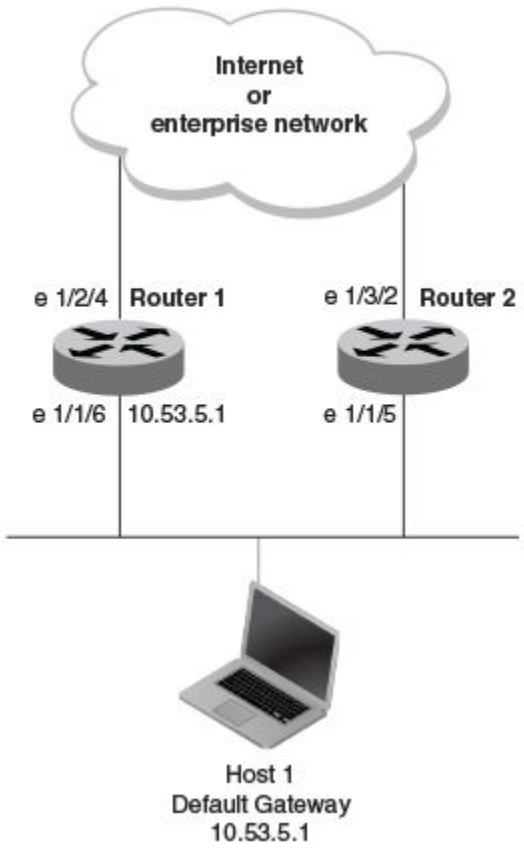
Virtual Router Redundancy Protocol (VRRP) is an election protocol that provides redundancy to routers within a Local Area Network (LAN).

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. Ruckus 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.

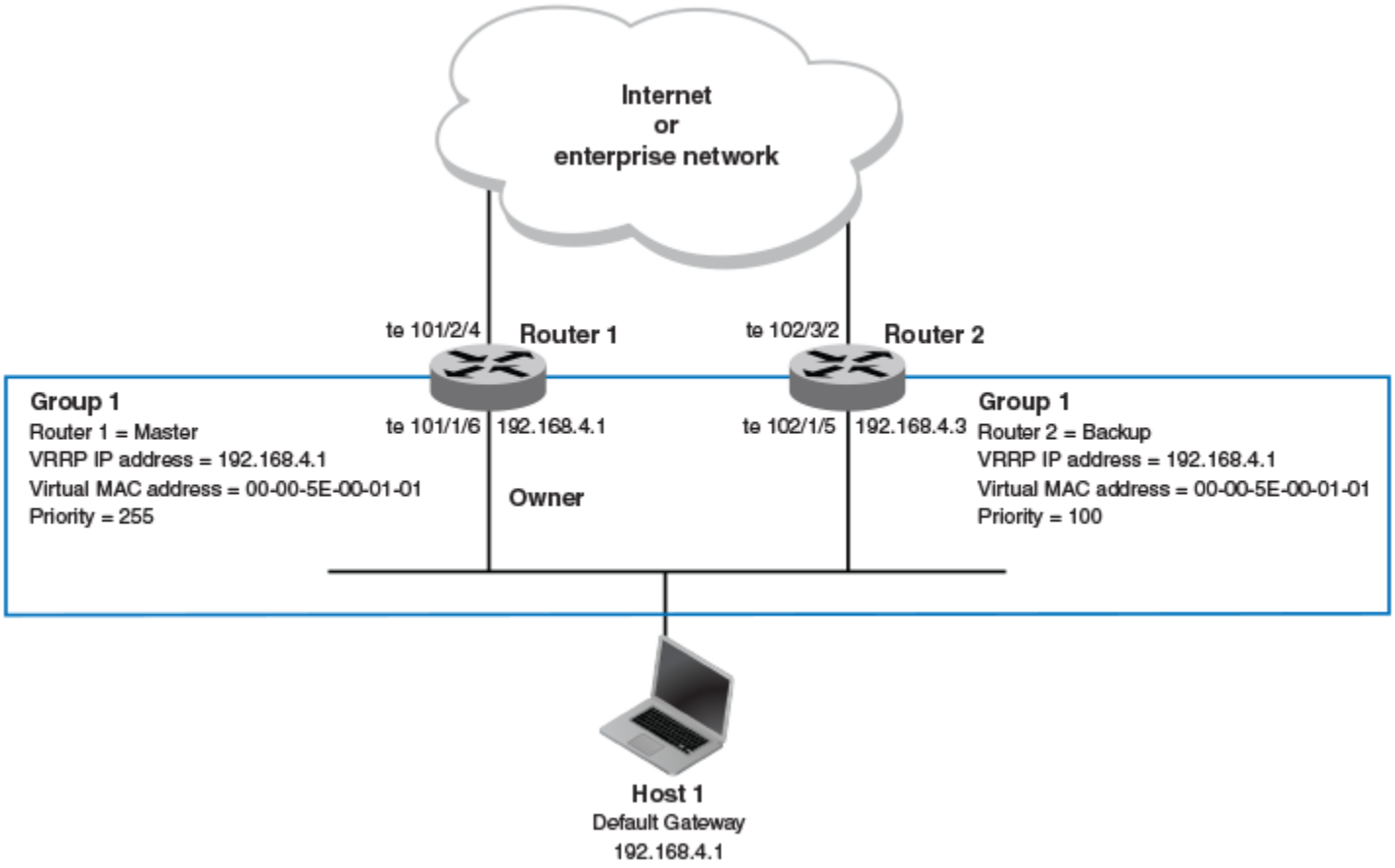
FIGURE 45 Single point of failure with Device 1 being the Host1 default gateway



To connect to the Internet or an internal intranet Host 1, in the figure, uses the IP address of 10.53.5.1 on Router 1 as its default gateway. If this interface goes down, Host1 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 46 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.

NOTE
Only 16 VRRP instances are configurable on the ICX 7150 device.

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 limitations on ICX devices

The implementation of VRRP varies across the ICX products.

Virtual router IDs can range from 1-255, but some ICX devices only support up to 16 VRRP instances.

Only IPv4 support is provided in VRRPv2. VRRPv3 supports both IPv4 and IPv6.

System Resource	ICX 7150
Max # of VRRP and VRRP-E sessions	16

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.

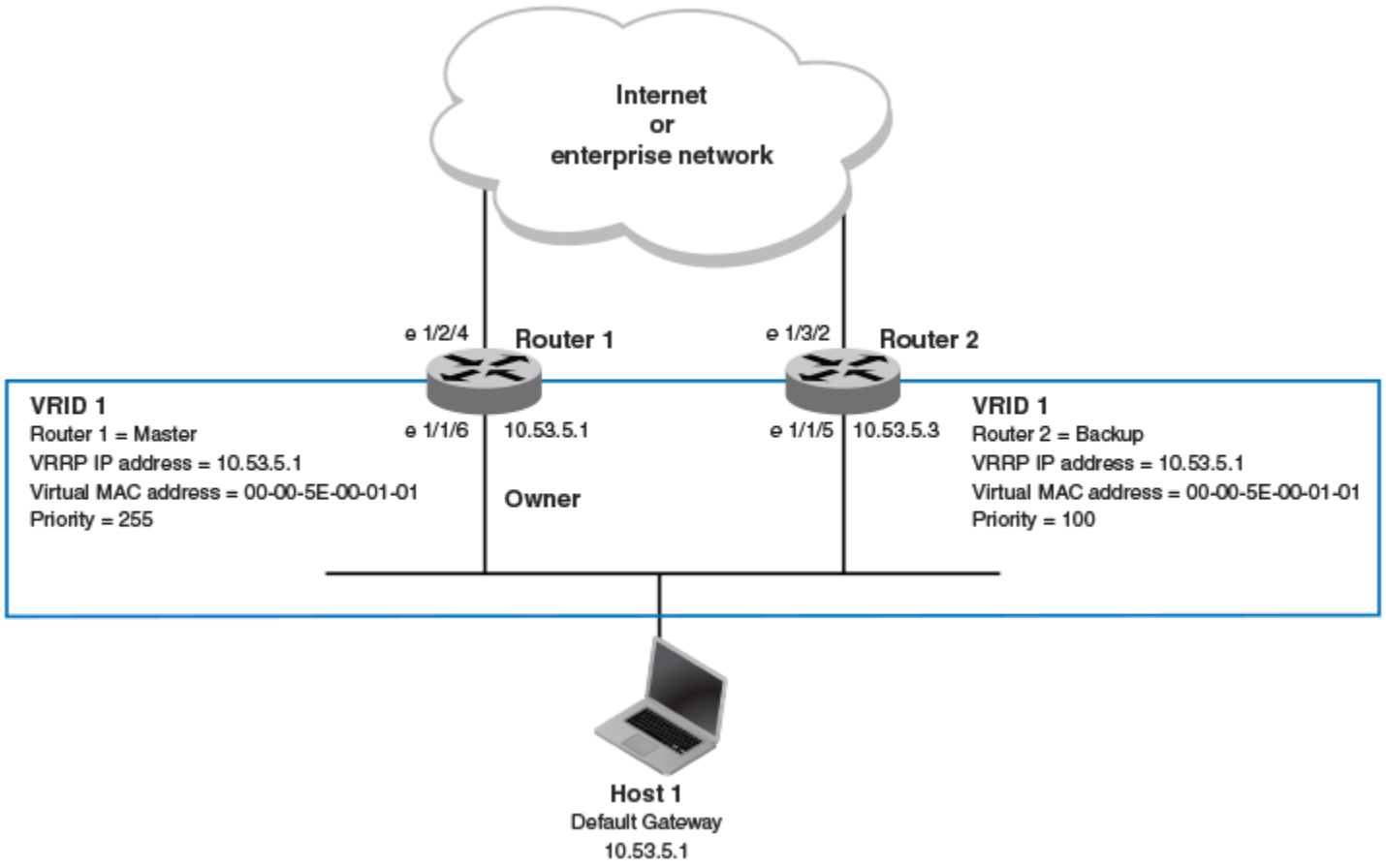
- VRRP—A control message is sent only once when the VRRP device assumes the role of the master.
- VRRP-E—A control message is sent every 2 seconds by the VRRP-E master device because VRRP-E control packets do not use the virtual MAC address.

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.

NOTE
Only 16 VRRP instances are configurable on the ICX 7150 device.

FIGURE 47 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 1/1/6
```

4. Configure the IP address of the interface.

```
device(config-if-e1000-1/1/6)# ip address 10.53.5.1/24
```

5. Assign Router 1 to the virtual router ID (VRID) 1.

```
device(config-if-e1000-1/1/6)# 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-e1000-1/1/6-vrid-1)# owner
```

7. Configure the VRRP version.

```
device(config-if-e1000-1/1/6-vrid-1)# version 2
```

8. Configure the IP address of the VRID.

```
device(config-if-e1000-1/1/6-vrid-1)# ip-address 10.53.5.1
```

9. Enable the VRRP session.

```
device(config-if-e1000-1/1/6-vrid-1)# activate
```

The following example configures a VRRP owner device.

```
device# configure terminal
device(config)# router vrrp
device(config)# interface ethernet 1/1/6
device(config-if-e1000-1/1/6)# ip address 10.53.5.1/24
device(config-if-e1000-1/1/6)# ip vrrp vrid 1
device(config-if-e1000-1/1/6-vrid-1)# owner
device(config-if-e1000-1/1/6-vrid-1)# version 2
device(config-if-e1000-1/1/6-vrid-1)# ip-address 10.53.5.1
device(config-if-e1000-1/1/6-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 47 on page 383 is assigned as a backup device. Repeat this task for all devices that are to be designated as backup devices.

NOTE
Only 16 VRRP instances are configurable on the ICX 7150 device.

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 for Router 2. All devices configured for the same virtual router ID (VRID) must be on the same subnet.

```
device(config-if-e1000-1/1/5)# ip address 10.53.5.3/24
```

5. Assign Router 2 to VRID 1, the same VRID as Router 1.

```
device(config-if-e1000-1/1/5)# 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-e1000-1/1/5-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 VRRP version.

```
device(config-if-e1000-1/1/5-vrid-1)# version 2
```

8. Configure the number of seconds between hello messages.

```
device(config-if-e1000-1/1/5-vrid-1)# hello-interval 10
```

9. 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-e1000-1/1/5-vrid-1)# advertise backup
```

10. Configure the IP address of the VRID.

```
device(config-if-e1000-1/1/5-vrid-1)# ip-address 10.53.5.1
```

The VRID IP address is the same virtual IP address you used for Router 1.

11. Enable the VRRP session.

```
device(config-if-e1000-1/1/5-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 1/1/5
device(config-if-e1000-1/1/5)# ip address 10.53.5.3/24
device(config-if-e1000-1/1/5)# ip vrrp vrid 1
device(config-if-e1000-1/1/5-vrid-1)# backup priority 110
device(config-if-e1000-1/1/5-vrid-1)# version 2
device(config-if-e1000-1/1/5-vrid-1)# hello-interval 10
device(config-if-e1000-1/1/5-vrid-1)# advertise backup
device(config-if-e1000-1/1/5-vrid-1)# ip-address 10.53.5.1
device(config-if-e1000-1/1/5-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 1/1/6
```

4. Enter the simple text password configuration using the **ip vrrp auth-type** command with a text password.

```
device(config-if-e1000-1/1/6)# 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-e1000-1/1/6-vrid-1)# show ip vrrp

Total number of VRRP routers defined: 1
Interface ethernet 1/1/6
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 1/1/6
   ```

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-e1000-1/1/6)# ip vrrp auth-type md5-auth gy42mb
   ```

5. Verify the password on the interface using the **show ip vrrp** command.

   ```
   device(config-if-e1000-1/1/6-vrid-1)# show ip vrrp

   Total number of VRRP routers defined: 1
   Interface ethernet 1/1/6
   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 1/1/6 and verifies the authentication type.

```
device# configure terminal
device(config)# router vrrp
device(config)# interface ethernet 1/1/6
device(config-if-e1000-1/1/6)# ip vrrp auth-type MD5 yourpwd
device(config-if-e1000-1/1/6-vrid-1)# show ip vrrp

Total number of VRRP routers defined: 1
Interface ethernet 1/1/6
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 1/1/6
   ```

4. Enter the virtual router ID (VRID) for which the device is the VRRP owner.

   ```
   device(config-if-e1000-1/1/6)# 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-e1000-1/1/6-vrid-1)# owner priority 99
   ```

6. Verify the abdication of the master device using the **show ip vrrp** command.

```
device(config-if-e1000-1/1/6-vrid-1)# show ip vrrp

Total number of VRRP routers defined: 1
Interface ethernet 1/1/6
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.

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 1/1/6
```

4. Enter the IP address for the interface to be used for the virtual router ID (VRID).

```
device(config-if-e1000-1/1/6)# 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-e1000-1/1/6)# ip vrrp vrid 1
```

6. Enter the **track-port** command to set the track port and priority:

```
device(config-if-e1000-1/1/6-vrid-1)# track-port ethernet 1/2/4 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 Ethernet interface 1/2/4 on virtual router 1 to be tracked; if the interface 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 1/1/6
device(config-if-e1000-1/1/6)# ip address 10.53.5.1/24
device(config-if-e1000-1/1/6)# ip vrrp vrid 1
device(config-if-e1000-1/1/6-vrid-1)# track-port ethernet 1/2/4 priority 20
```

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 1/1/5
```

2. Enter the IP address for the interface to be used for the virtual router ID (VRID).

```
device(config-if-e1000-1/1/5)# ip address 10.53.5.3/24
```

3. Enter the following command to enter the appropriate VRRP VRID mode.

```
device(config-if-e1000-1/1/5)# ip vrrp vrid 1
```

4. Enter the **non-preempt-mode** command to disable backup preemption.

```
device(config-if-e1000-1/1/5-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 1/1/5
device(config-if-e1000-1/1/5)# ip address 10.53.5.3/24
device(config-if-e1000-1/1/5)# ip vrrp vrid 1
device(config-if-e1000-1/1/5-vrid-1)# non-preempt-mode
```

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. When the device acting as the master device is not the IP address owner (the device with the interface whose actual IP address is used as the virtual device's IP address), the master device accepts only the ARP packets sent to the virtual IP address. When accept mode is configured, the master device responds to ping, TELNET, and traceroute packets sent to the virtual IP address even when the master device is not the IP address owner.

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.

```
device# configure terminal
device(config)# router vrrp
device(config)# interface ethernet 1/1/5
device(conf-if-e1000-1/1/5)# ip address 10.53.5.3/24
device(conf-if-e1000-1/1/5)# ip vrrp vrid 1
device(conf-if-e1000-1/1/5-vrid-1)# backup priority 110
device(conf-if-e1000-1/1/5-vrid-1)# accept-mode
```

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.

To create VRRP-E, Ruckus has 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.

VRRP-E is not supported on non-Ruckus devices and does not interoperate with VRRP sessions on Ruckus devices.

NOTE
Only 16 VRRP instances are configurable on the ICX 7150 device.

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.

NOTE
Only 16 VRRP instances are configurable on the ICX 7150 device.

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 1/1/5
```

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-e1000-1/1/5)# ip address 10.53.5.3/24
```

5. Assign the device to VRID 1.

```
device(config-if-e1000-1/1/5)# 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-e1000-1/1/5-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 VRRP version.

```
device(config-if-e1000-1/1/5-vrid-1)# version 2
```

8. Configure the IP address of the VRID.

```
device(config-if-e1000-1/1/5-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.

9. Enable the VRRP-E session.

```
device(config-if-e1000-1/1/5-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 1/1/5
device(config-if-e1000-1/1/5)# ip address 10.53.5.3/24
device(config-if-e1000-1/1/5)# ip vrrp-extended vrid 1
device(config-if-e1000-1/1/5-vrid-1)# backup priority 110
device(config-if-e1000-1/1/5-vrid-1)# version 2
device(config-if-e1000-1/1/5-vrid-1)# ip-address 10.53.5.254
device(config-if-e1000-1/1/5-vrid-1)# activate
VRRP-E router 1 for this interface is activating
```

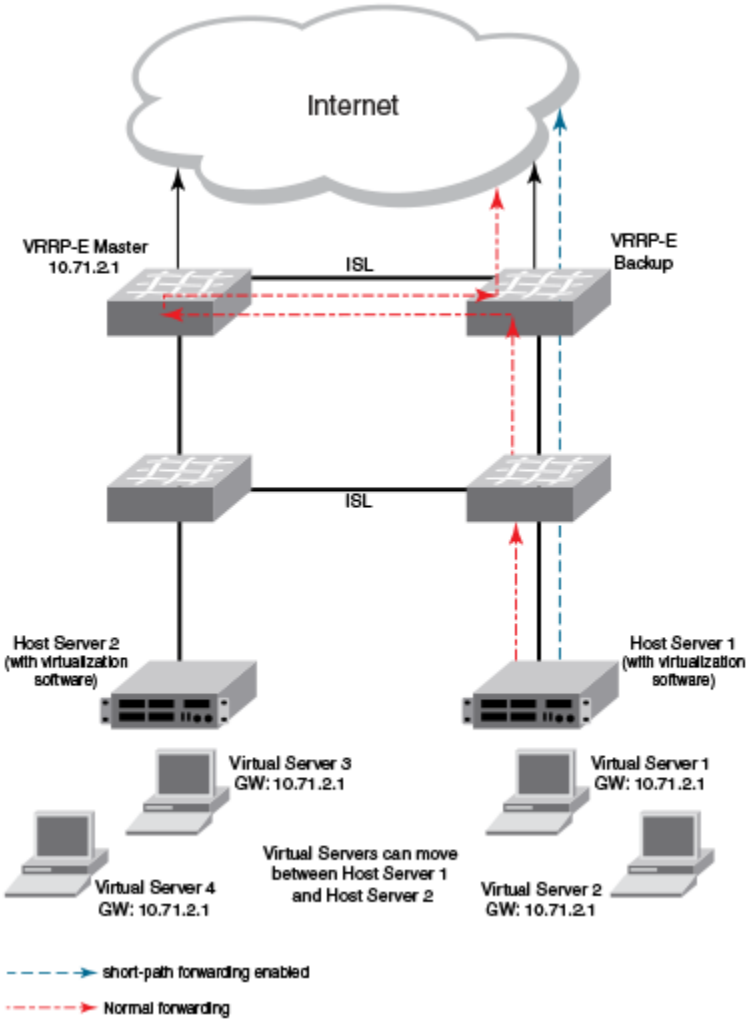
VRRP-E load-balancing using short-path forwarding

The VRRP-E Extension for Server Virtualization feature allows Ruckus 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 48 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.

Short-path forwarding with revert priority

Revert priority is used to dynamically enable or disable VRRP-E short-path forwarding.

If short-path forwarding is configured with revert priority on a backup router, the revert priority represents a threshold for the current priority of the VRRP-E session. When the backup device priority is higher than the configured revert priority, 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.

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 hitless upgrade

When the Virtual Routing Redundancy Protocol Extended (VRRP-E) hitless upgrade capability is configured, traffic loss can be avoiding during the failover process when an upgrade is installed or when troubleshooting is being performed on a VRRP-E master device.

Virtual Router Redundancy Protocol Extended (VRRP-E) backup devices elect a master VRRP-E device based on the device with the highest virtual router ID (VRID) priority value. When scheduling an upgrade, the standard failover behavior involves increasing the priority value of a backup VRRP-E device to enable it to assume the role of VRRP-E master. VRRP-E hitless upgrade provides functionality to decrease the priority of the virtual router identifier (VRID) to a value of 1 for a master VRRP-E device. When the short-path forwarding functionality is also configured, the potential for traffic loss is avoided because backup devices can forward traffic. A new command-line interface (CLI) command is introduced to decrease the priority of VRIDs on the master device before an upgrade. A second use case is when troubleshooting must be performed on the master VRRP-E device.

To implement the hitless upgrade, configure short-path forwarding on all the devices assigned to the same virtual router ID (VRID) as the master device. Load the upgrade image and, before the device reloads, enable the CLI that decreases the VRID priority of the master device. If you save the configuration to the startup configuration file before the reload, the device will remain a backup device after rebooting to allow time to check that the new image is working correctly. After the hitless upgrade functionality is configured, VRRP-E activates the automatic backup by setting the master device VRID priority to one. All VRRP-E devices become backup devices and the software chooses the device with the highest VRID priority to become the master device. Using short-path forwarding where a backup device can forward traffic, any data traffic loss is avoided.

After the upgrade has been performed, the previous master device boots up as a VRRP-E backup device. To return the device to its previous role, remove the hitless upgrade configuration. Remember to save the startup configuration after the hitless upgrade command removal because the system is in maintenance mode after the VRRP-E hitless upgrade is enabled and no other device configuration changes are recommended.

VRRP-E hitless upgrade functionality is only supported in VRRP-E IPv4. This feature enables short-path forwarding support on all FastIron devices.

Configuring VRRP-E hitless upgrade

Configure the VRRP-E hitless upgrade capability to avoid traffic loss while upgrading or troubleshooting a VRRP-E master device.

Before configuring VRRP-E hitless upgrade, VRRP-E must be configured on all devices used in the VRRP-E session. To avoid any traffic loss during the failover process, enable short-path forwarding on all VRRP-E devices.

Perform this task on the master VRRP-E Layer 3 devices to configure the VRRP-E hitless upgrade capability.

1. 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. Configure the Virtual Ethernet (VE) interface for the VRRP-E device.

```
device(config-vrrpe-router)# interface ve 10
```

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-vif-10)# ip address 192.168.4.1/24
```

5. Assign this device to VRID 5.

```
device(config-vif-10)# ip vrrp-extended vrid 5
```

6. Designate this router as a backup VRRP device. The backup device with the highest priority assumes the role of master VRRP-E device.

```
device(config-vif-10-vrid-5)# backup priority 110
```

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.

```
device(config-vif-10-vrid-5)# short-path-forwarding
```

9. Activate the VRRP-E instance.

```
device(config-vif-10-vrid-5# activate
```

10. Return to virtual interface configuration mode.

```
device(config-vif-10-vrid-5# exit
```

11. Return to global configuration mode.

```
device(config-vif-10)# exit
```

12. Enter the **activate backup** command to work in conjunction with the short-path forwarding configuration to enable VRRP-E hitless upgrade.

```
device(config-vrrpe-router)# activate backup
```

    In this example, VRRP-E hitless upgrade is enabled on the master VRRP-E device. The priority of the master VRRP-E device is set to 1 and the backup device with the highest priority assumes the role of the master VRRP-E device.

13. Return to global configuration mode.

```
device(config-vrrpe-router)# exit
```

14. (Optional) You can write the running configuration file to the startup configuration file to ensure that the device remains as a backup device until after the reload.

```
device(config)# write memory
```

Be aware that while the hitless upgrade is enabled, no other device configuration is recommended; the system is in maintenance mode. After your upgrade or troubleshooting is complete, remove the **activate backup** command.

In the following example, VRRP-E hitless upgrade is enabled and the running-config file is saved to the startup-config file.

```
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 110
device(config-vif-10-vrid-5)# ip-address 192.168.4.254
device(config-vif-10-vrid-5)# short-path-forwarding
device(config-vif-10-vrid-5)# activate
device(config-vif-10-vrid-5)# exit
device(config-vif-10)# exit
device(config-vrrpe-router)# activate backup
device(config-vrrpe-router)# exit
device(config)# write memory
```

While the VRRP-E hitless upgrade is enabled, load your upgrade image and reboot the device or perform any troubleshooting.

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
```

Configuration example: ISSU upgrade using VRRP-E

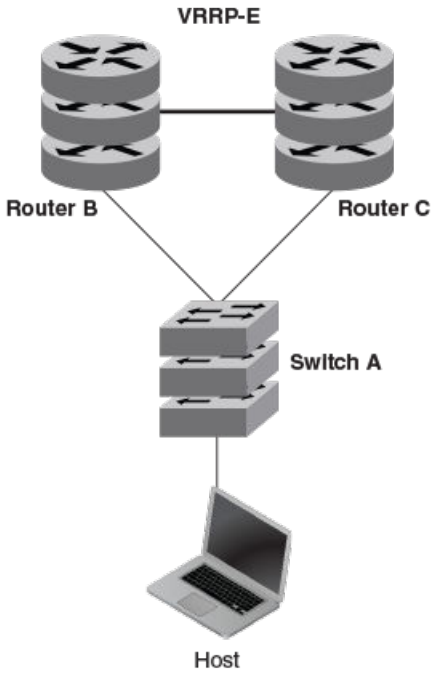
Using VRRP-E, an In Service Software Upgrade (ISSU) can be performed with minimal downtime.

VRRP-E supports ISSU and combined with the short-path forwarding feature, high availability can be achieved. When a software upgrade has to be performed, the backup router can be upgraded first and after it comes back online, the VRRP-E priority can be set to be higher than the current master. A transition is initiated by the software, and with minimal packet loss, the backup router becomes the master router running the upgraded software version. Perform the following steps that utilize the configurations and network diagram.

NOTE
Before configuring VRRP-E, configure your network with Layer 3 protocols using OSPF and RIP.

1. On Router B and Router C in the diagram apply the example configurations.
2. The software selects Router C as the master VRRP-E device because the priority and IP address are higher than Router B.
3. Upgrade the software version on Router B, the backup router, and reload.
4. Router B comes online and joins the network.
5. Increase the priority on Router B using the **backup priority 254** command and options in VRID interface configuration mode.
6. The software transitions the role of VRRP-E master to Router B with only 30 milliseconds of packet loss.
7. Upgrade the software version on Router C, which has become the backup router, and reload.
8. Router C comes online and joins the network.

FIGURE 49 VRRP-E network setup for ISSU



Router B configuration

The following example configuration configures VRRP-E using the short-path forwarding feature. On this device, the priority value for VRID 23 is set to 50.

```
configure terminal
 router vrrp-extended
  interface ve 123
  ip address 192.168.4.11 255.255.255.0
  ip vrrp-extended vrid 23
   backup priority 50
   advertise backup
   ip-address 192.168.4.254
   short-path-forwarding
   activate
```

Router C configuration

The following example configuration configures VRRP-E using the short-path forwarding feature. On this device, the priority value for VRID 23 is set to 250.

```
configure terminal
 router vrrp-extended
  interface ve 123
  ip address 192.168.4.12 255.255.255.0
  ip vrrp-extended vrid 23
   backup priority 250
   advertise backup
   ip-address 192.168.4.254
   short-path-forwarding
   activate
```

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 1/1/1
---------------
auth-type no authentication
VRID 1 (index 1)
interface 1/1/1
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
-------------------------------------------------------------------
1/1/1 10    255      P2-      Master  Local     Unknown    10.30.30.2
1/1/3 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 1/1/5.

```
device# show ip vrrp-extended statistics ethernet 1/1/5

Interface 1/1/5
----------------
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 1/1/5.

```
device# show ip vrrp statistics ethernet 1/1/5

Interface 1/1/5
----------------
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 1/1/5.

```
device# show ip vrrp statistics ethernet 1/1/5

Interface 1/1/5
----------------
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 = 8
  . 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. Ruckus has implemented IPv6 VRRPv3 functionality to comply with RFC 5798 and will interoperate comfortably with other vendors that support RFC 5798.

VRRP limitations on ICX devices

The implementation of VRRP varies across the ICX products.

Virtual router IDs can range from 1-255, but some ICX devices only support up to 16 VRRP instances.

Only IPv4 support is provided in VRRPv2. VRRPv3 supports both IPv4 and IPv6.

System Resource	ICX 7150
Max # of VRRP and VRRP-E sessions	16

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. Ruckus has implemented IPv6 VRRPv3 functionality to comply with RFC 5798 and will interoperate well with other vendors that support RFC 5798.

NOTE
Only 16 VRRP instances are configurable on the ICX 7150 device.

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-ipv6-vrrp-router)# interface ethernet 1/1/5
```

5. Configure the IPv6 address of the interface.

```
device(config-if-e1000-1/1/5)# ipv6 address fd2b::2/64
```

6. Assign the owner device to the virtual router ID (VRID) 2.

```
device(config-if-e1000-1/1/5)# 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-e1000-1/1/5-vrid-2)# owner
```

8. Configure the VRRP version.

```
device(config-if-e1000-1/1/5-vrid-2)# version 3
```

9. Assign an IPv6 link-local address to the VRID for use in the local network.

```
device(config-if-e1000-1/1/5-vrid-2)# ipv6-address fe80::768e:f8ff:fe2a:0099
```

10. Assign a global IPv6 address to the VRID.

```
device(config-if-e1000-1/1/5-vrid-2)# ipv6-address fd2b::2
```

11. Enable the VRRP session.

```
device(config-if-e1000-1/1/5-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 1/1/5
device(config-if-e1000-1/1/5)# ipv6 address fd2b::2/64
device(config-if-e1000-1/1/5)# ipv6 vrrp vrid 2
device(config-if-e1000-1/1/5-vrid-2)# owner
device(config-if-e1000-1/1/5-vrid-2)# version 3
device(config-if-e1000-1/1/5-vrid-2)# ipv6-address fe80::768e:f8ff:fe2a:0099
device(config-if-e1000-1/1/5-vrid-2)# ipv6-address fd2b::2
device(config-if-e1000-1/1/5-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. Ruckus has implemented IPv6 VRRPv3 functionality to comply with RFC 5798 and will interoperate well with other vendors that support RFC 5798.

NOTE
Only 16 VRRP instances are configurable on the ICX 7150 device.

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 1/1/4
```

4. Configure the IPv6 address of the interface.

```
device(config-if-e1000-1/1/4)# ipv6 address fd2b::3/64
```

5. Assign the backup device to the virtual router ID (VRID) 2.

```
device(config-if-e1000-1/1/4)# 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-e1000-1/1/4-vrid-2)# backup priority 100
```

7. Configure the VRRP version.

```
device(config-if-e1000-1/1/4-vrid-2)# version 3
```

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-e1000-1/1/4-vrid-2)# advertise backup
```

9. Assign the IPv6 link-local address to the VRID for use in the local network.

```
device(config-if-e1000-1/1/4-vrid-2)# ipv6-address fe80::768e:f8ff:fe2a:0099
```

10. Assign the global IPv6 address to the VRID.

```
device(config-if-e1000-1/1/4-vrid-2)# ipv6-address fd2b::2
```

11. Enable the VRRP session.

```
device(config-if-e1000-1/1/4-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 1/1/4
device(config-if-e1000-1/1/4)# ipv6 address fd2b::3/64
device(config-if-e1000-1/1/4)# ipv6 vrrp vrid 2
device(config-if-e1000-1/1/4-vrid-2)# backup priority 100
device(config-if-e1000-1/1/4-vrid-2)# version 3
device(config-if-e1000-1/1/4-vrid-2)# advertise backup
device(config-if-e1000-1/1/4-vrid-2)# ipv6-address fe80::768e:f8ff:fe2a:0099
device(config-if-e1000-1/1/4-vrid-2)# ipv6-address fd2b::2
device(config-if-e1000-1/1/4-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.

NOTE
Only 16 VRRP instances are configurable on the ICX 7150 device.

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 1/1/6
```

4. Configure the IP address of the interface.

```
device(config-if-e1000-1/1/6)# ip address 10.53.5.1/24
```

5. Assign the virtual router ID (VRID) 1 to the interface.

```
device(config-if-e1000-1/1/6)# 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-e1000-1/1/6-vrid-1)# owner
```

7. Configure the VRRP version.

```
device(config-if-e1000-1/1/6-vrid-1)# version 3
```

In this step, VRRPv3 is selected.

8. Configure the IP address of the VRID.

```
device(config-if-e1000-1/1/6-vrid-1)# ip-address 10.53.5.1
```

9. Enable the VRRP session.

```
device(config-if-e1000-1/1/6-vrid-1)# activate
```

The following example configures an IPv4 VRRPv3 owner device.

```
device# configure terminal
device(config)# router vrrp
device(config)# interface ethernet 1/1/6
device(config-if-e1000-1/1/6)# ip address 10.53.5.1/24
device(config-if-e1000-1/1/6)# ip vrrp vrid 1
device(config-if-e1000-1/1/6-vrid-1)# owner
device(config-if-e1000-1/1/6-vrid-1)# version 3
device(config-if-e1000-1/1/6-vrid-1)# ip-address 10.53.5.1
device(config-if-e1000-1/1/6-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.

NOTE
Only 16 VRRP instances are configurable on the ICX 7150 device.

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 1/1/5
```

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-e1000-1/1/5)# ip address 10.53.5.3/24
```

5. Assign the same VRID as the VRID used by the owner device.

```
device(config-if-e1000-1/1/5)# 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-e1000-1/1/5-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. Set the VRRP version to 3 to indicate that this is VRRPv3 session for IPv4.

```
device(config-if-e1000-1/1/5-vrid-1)# version 3
```

8. Configure the IP address of the VRID.

```
device(config-if-e1000-1/1/5-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.

9. Enable the VRRP session.

```
device(config-if-e1000-1/1/5-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 1/1/5
device(config-if-e1000-1/1/5)# ip address 10.53.5.3/24
device(config-if-e1000-1/1/5)# ip vrrp vrid 1
device(config-if-e1000-1/1/5-vrid-1)# backup priority 110
device(config-if-e1000-1/1/5-vrid-1)# version 3
device(config-if-e1000-1/1/5-vrid-1)# ip-address 10.53.5.1
device(config-if-e1000-1/1/5-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.

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 1/1/6
   ```

4. Enter the IPv6 address for the interface to be used for the virtual router ID (VRID).

```
device(config-if-e1000-1/1/6)# ipv6 address fd2b::2/64
```

5. Enter the following command to enter the appropriate VRRPv3 virtual router ID (VRID) mode.

```
device(config-if-e1000-1/1/6)# ipv6 vrrp vrid 1
```

6. Enter the **track-port** command to set the tracked port and priority:

```
device(config-if-e1000-1/1/6-vrid-1)# track-port ethernet 1/2/4 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 interface Ethernet 1/2/4 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 1/1/6
device(config-if-e1000-1/1/6)# ipv6 address fd2b::2/64
device(config-if-e1000-1/1/6)# ipv6 vrrp vrid 1
device(config-if-e1000-1/1/6-vrid-1)# track-port ethernet 1/2/4 priority 20
```

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. When the device acting as the master device is not the IP address owner (the device with the interface whose actual IP address is used as the virtual device's IP address), the master device accepts only the ARP packets sent to the virtual IP address. When accept mode is configured, the master device responds to ping, TELNET, and traceroute packets sent to the virtual IP address even when the master device is not the IP address owner.

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.

```
device# configure terminal
device(config)# router vrrp
device(config)# interface ethernet 1/1/5
device(conf-if-e1000-1/1/5)# ip address 10.53.5.3/24
device(conf-if-e1000-1/1/5)# ip vrrp vrid 1
device(conf-if-e1000-1/1/5-vrid-1)# backup priority 110
device(conf-if-e1000-1/1/5-vrid-1)# accept-mode
```

Alternate VRRPv2 checksum for VRRPv3 IPv4 sessions

If VRRPv3 is configured on a Ruckus 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 Ruckus 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 Ruckus 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 Ruckus 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 1/2/4
```

4. To configure a VRRP virtual routing ID, use the **ip vrrp vrid** command with an associated ID number.

```
device(config-if-e1000-1/2/4)# ip vrrp vrid 14
```

5. To enable VRRP version 3 (VRRPv3), enter the **version** command with a version number of v3.

```
device(config-if-e1000-1/2/4-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-e1000-1/2/4-vrid-14)# use-v2-checksum
```

7. Enter the IP address for the interface using the **ip-address** command.

```
device(config-if-e1000-1/2/4-vrid-14)# ip-address 10.14.14.99
```

8. To activate the interface, enter the **activate** command.

```
device(config-if-e1000-1/2/4-vrid-14)# activate
```

The following example shows the v2 checksum computation method enabled for an VRRPv3 IPv4 session on a Ruckus device.

```
device# configure terminal
device(config)# router vrrp
device(config)# interface ethernet 1/2/4
device(config-if-e1000-1/2/4)# ip vrrp vrid 14
device(config-if-e1000-1/2/4-vrid-14)# version v3
device(config-if-e1000-1/2/4-vrid-14)# use-v2-checksum
device(config-if-e1000-1/2/4-vrid-14)# ip-address 10.14.14.99
device(config-if-e1000-1/2/4-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 1/2/4
 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 1/2/4
---------------
auth-type no authentication

VRID 14 (index 1)
 interface 1/2/4
 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, Ruckus 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 a Ruckus proprietary protocol, VRRP Extended version 3 (VRRP-Ev3) is not supported.

Assigning an auto-generated link-local IPv6 address for a VRRPv3 cluster

A virtual link-local IPv6 address can be auto-generated and assigned as the virtual IPv6 address of a VRRPv3 session.

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. To auto-generate and assign a virtual link-local IPv6 address as the virtual IPv6 address of a VRRPv3 cluster, use the following steps on either an IPv6 VRRPv3 owner or backup device.

1. From privileged EXEC mode, enter global configuration mode.

```
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)# interface ve 3
```

4. Configure the global IPv6 address of the interface.

```
device(config-vif-3)# ipv6 address fd3b::3/64
```

5. Assign the device to virtual router ID (VRID) 2.

```
device(config-vif-3)# 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 owner device.

```
device(config-vif-3-vrid-2)# owner
```

7. Automatically generate the IPv6 link-local address for the VRID for use in the local network.

```
device(config-vif-3-vrid-2)# ipv6-address auto-gen-link-local
```

8. Enable the VRRP session.

```
device(config-vif-3-vrid-2)# activate
```

The following example shows the auto-generation of a virtual link-local IPv6 address and its allocation as the virtual IPv6 address of a VRRPv3 session on an IPv6 VRRPv3 owner router.

```
device# configure terminal
device(config)# ipv6 router vrrp
device(config)# interface ve 3
device(config-vif-3)# ipv6 address fd3b::3/64
device(config-vif-3)# ipv6 vrrp vrid 2
device(config-vif-3-vrid-2)# owner
device(config-vif-3-vrid-2)# ipv6-address auto-gen-link-local
device(config-vif-3-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 1/1/3
----------------
auth-type no authentication
VRID 13 (index 2)
interface 1/1/3
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 1/1/3
----------------
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 Ruckus 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, Ruckus 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.

VRRP-Ev3 is not supported on non-Ruckus devices and does not interoperate with VRRPv2 or VRRPv3 sessions on Ruckus devices.

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.

NOTE
Only 16 VRRP instances are configurable on the ICX 7150 device.

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 1/1/7
```

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-e1000-1/1/7)# ipv6 address fd4b::4/64
```

5. Assign the device to VRID 4.

```
device(config-if-e1000-1/1/7)# 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-e1000-1/1/7-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 the VRRP version.

```
device(config-if-e1000-1/1/7-vrid-4)# version 3
```

8. Configure an IPv6 link-local address for the VRID.

```
device(config-if-e1000-1/1/7-vrid-4)# ipv6-address fe80::768e:f8ff:fe2a:0089
```

9. Configure a global IPv6 address for the VRID.

```
device(config-if-e1000-1/1/7-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.

10. Enable the VRRP session.

```
device(config-if-e1000-1/1/7-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 1/1/7
device(config-if-e1000-1/1/7)# ipv6 address fd4b::4/64
device(config-if-e1000-1/1/7)# ipv6 vrrp-extended vrid 4
device(config-if-e1000-1/1/7-vrid-4)# backup priority 50
device(config-if-e1000-1/1/7-vrid-4)# version 3
device(config-if-e1000-1/1/7-vrid-4)# ipv6-address fe80::768e:f8ff:fe2a:0089
device(config-if-e1000-1/1/7-vrid-4)# ipv6-address fd4b::99
device(config-if-e1000-1/1/7-vrid-4)# activate
VRRP-E router 4 for this interface is activating
```

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
-------------------------------------------------------------------
1/1/3   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 1/1/1
----------------
auth-type md5-authentication
VRID 1 (index 1)
interface 1/1/1
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
```

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.

NOTE
ICX 7150 devices do not support VRFs.

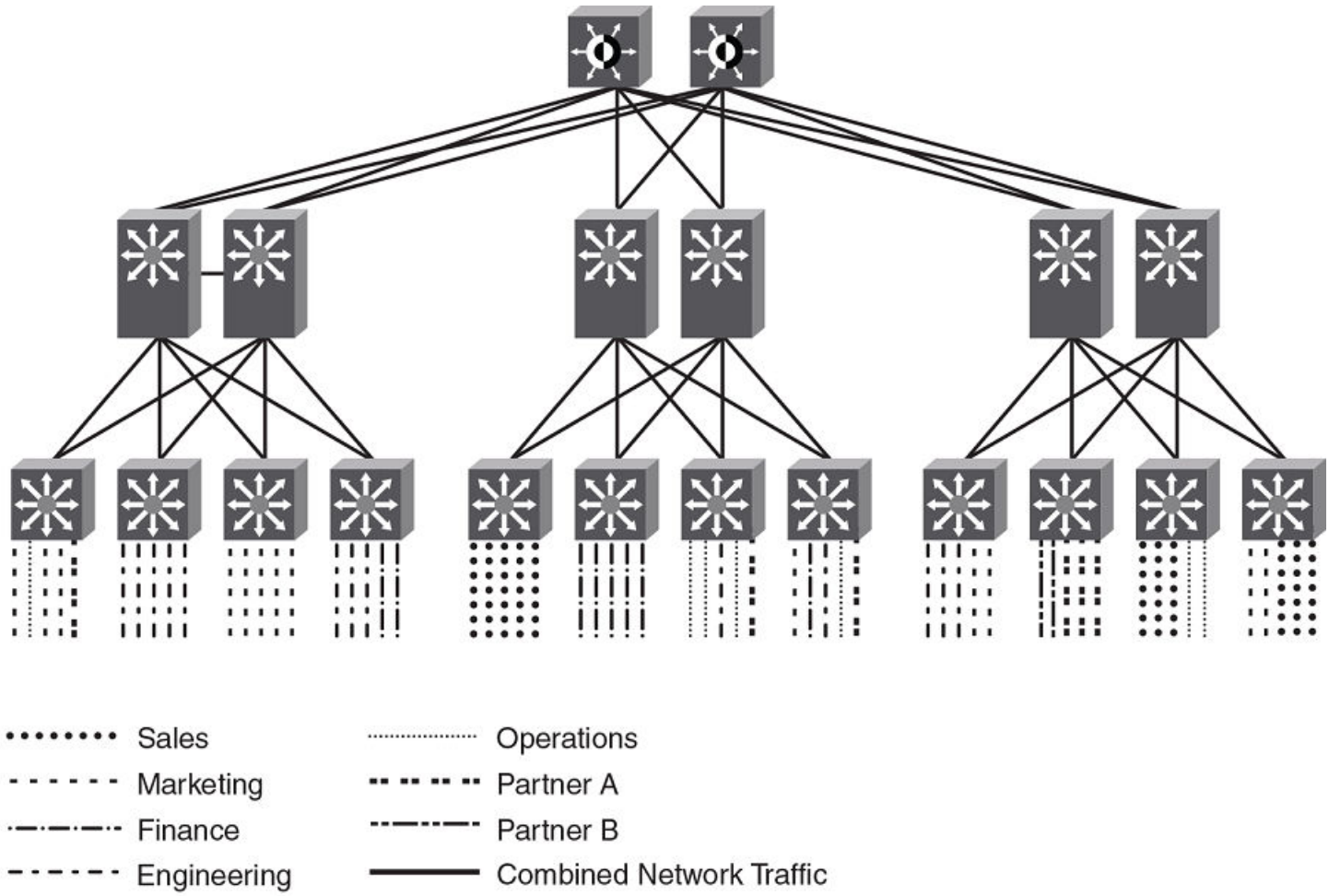
Central to VRF-Lite is the ability to maintain multiple VRF tables on the same Provider Edge (PE) Router. VRF-Lite uses multiple instances of a routing protocol such as OSPF or BGP to exchange route information for a VPN among peer PE routers. The VRF-Lite capable PE router maps an input customer interface 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 50 Example high-level Multi-VRF topology



A Multi-VRF instance can be configured on any of the following:

- Platforms that support untagged physical ports - Applies only to the ICX 7250,ICX 7250 ICX 7450, ICX 7650, ICX 7750 and Ruckus ICX 7850 . It is recommended that these ports be configured "route-only" to prevent the leaking of switching traffic if two interfaces in the same VLAN are configured with different VRFs.
- 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:

- Configure VRF-related system-max values.
- (Optional) Configure tagging on peer interfaces for security.

- Configure 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.

FastIron considerations for Multi-VRF

When a VRF is configured, a warning message specifies that any configuration existing on the interface is deleted.

NOTE
ICX 7150 devices do not support VRFs.

NOTE
ICX 7250 devices support VRF-lite from FI 08.0.50.

VRF-related system-max Values

The default FastIron configuration does not allow space for VRF routing tables. As a result, you must modify VRF-related system-max values before configuring a VRF instance. The following table lists commands that configure system-max values at the global level.

NOTE
For the ICX 7850, values are used from the forwarding profile and there is no option to change these values. When a new profile is selected, the values from the new profile are applied. Refer to the **forwarding-profile** command in the *Ruckus FastIron Command Reference* and the *Configuration Fundamentals* chapter in the *Ruckus FastIron Management Configuration Guide* for more information.

TABLE 20 Commands for Configuring system-max Values

Command	Description
ip-vrf	Configures maximum VRF instances supported by the software.
ip-route	Configures maximum IPv4 routes, used to initialize hardware during system init.
ip6-route	Configures maximum IPv6 routes, used to initialize hardware during system init.
ip-route-default-vrf	Configures maximum IPv4 routes to be allocated for the default VRF instance.
ip6-route-default-vrf	Configures maximum IPv6 routes to be allocated for the default VRF instance.
ip-route-vrf	Configures default maximum IPv4 routes to be allocated per user-defined VRF.
ip6-route-vrf	Configures default maximum IPv6 routes to be allocated per user-defined VRF.

For the IPv4 partition, the default value for IPv4 TCAM allocation is decreased to 10,000. IPv6 TCAM allocation can then be increased from the default value of 908 to 1408. Both IPv4 and IPv6 VRF instances are planned to allocate 500 routes each.

The following table lists the **ip-vrf** configuration limits for the **system-max** command, by line card and platform.

TABLE 21 Configuration Limits for ip-vrf With The system-max Command

Hardware	Minimum	Default	Maximum
ICX 7250	4	16	16
ICX 7450	4	32	32
ICX 7650	16	128	128
ICX 7750	16	128	128
ICX 7850	16	128	128

TABLE 22 Additional Configuration Limits For The system-max Command

Configuration	ICX 7450			ICX 7650 / ICX 7750		
	Min	Default	Max	Min	Default	Max
ip-route (system-max IPv4 routes that all VRFs in total can support)	4096	12000	15168	98304	98304	131072
ip6-route (system-max IPv6 routes that all VRFs in total can support)	68	5120	5120	5120	5120	7168
ip-route-default-vrf (system-max IPv4 routes configuration for default-VRF)	1024	12000	15168	256	65536	131072
ip-route-vrf (default system-max IPv4 routes per non-default-VRF instances)	128	1024	15168	64	4096	131072
ip6-route-default-vrf (system-max IPv6 routes configuration for default-VRF)	64	5120	5120	64	2048	7168
ip6-route-vrf (default system-max IPv6 routes per non-default-VRF instances, for 3rd generation line cards)	64	100	5120	16	1024	7168

The following table lists values for **ip-route** and **ip6-route** on the ICX 7250.

TABLE 23 Configuration Limits for ip-route and ip6-route on ICX 7250 Devices

Configuration	Min	Default	Max
ip-route	4096	12000	12000
ip6-route	68	730	2048

The following table lists values for **ip-route** and **ip6-route** on the Ruckus ICX 7150. Values vary depending on the IPv6 prefix length.

TABLE 24 Configuration Limits for ip-route and ip6-route on Ruckus ICX 7150 Devices

Configuration	Min	Default	Max
ip-route	20	800	1000
ip6-route (prefix length 0 – 64)	20	220	1000
ip6-route (prefix length 65 – 128)	20	256	256

The following table lists values for **ip-cache** and **ip6-cache** on the Ruckus ICX 7150.

TABLE 25 Configuration Limits for ip-cache and ip6-cache on Ruckus ICX 7150 Devices

Configuration	Min	Default	Max
ip-cache	20	2048	4076
ip6-cache	10	1024	2038

The following examples illustrate the **system-max** values to support two VRF instances for IPv4 and two instances for IPv6.

- To allocate 2 x 500 routes for IPv4 user-VRF, (10000 - (500+500) = 9000 routes):

```
device(config)# system-max ip-route-default-vrf 9000
Total max configured ipv4 routes are 12000
  - Max ipv4 routes configured for default VRF are 9000
  - Max ipv4 routes available for all non-default VRFs are 3000
Warning: Please revalidate these values to be valid for your configuration.
Reload required.  Please write memory and then reload or power cycle.
device#
```

- To modify the IPv4 partition after modifying the **ip-route-default-vrf** value:

```
device(config)# system-max ip-route 10000
ip-route and ip6-route values changed.
ip-route: 10000
ip6-route: 1408
Warning: Please reconfigure system-max for ip-route-default-vrf and ip-route-vrf (if required).
Reload required.  Please write memory and then reload or power cycle.
device#
```

NOTE
This example also modifies the **ip6-route system-max** parameter and is intended only for the ICX 7450.

- To allocate 2 x 500 routes for IPv6 user-VRF (1408 - (500+500) = 408):

```
device(config)# system-max ip6-route-default-vrf 408
Total max configured ipv6 routes are 1408
  - Max ipv6 routes configured for default VRF are 408
  - Max ipv6 routes available for all non-default VRFs are 1000
Warning: Please revalidate these values to be valid for your configuration.
Reload required.  Please write memory and then reload or power cycle.
device#
```

- To allocate 500 routes for IPv6 user-VRF:

```
device# system-max ip6-route-vrf 500
Reload required.  Please write memory and then reload or power cycle.
device# end
```

- To save the configuration changes:

```
device# write memory
Write startup-config done.
device# Flash Memory Write (8192 bytes per dot) .
Flash to Flash Done.
```

- After the system reloads, the system-max configuration appears as an active configuration.

```
!
system-max ip-route 12000
system-max ip6-route 5120
system-max ip-route-default-vrf 9000
system-max ip6-route-default-vrf 5120
system-max ip-route-vrf 500
```

```
system-max ip6-route-vrf 500
!
```

Additional features to support Multi-VRF

In addition to basic features, you can configure dynamic ARP inspection, DHCP snooping, and IP Source Guard to support Multi-VRF.

Static ARP

Static ARP entries help ensure Layer 2 to Layer 3 mappings. This removes some network overhead in the form of ARP requests and replies and can be helpful in managing Multi-VRF networks where devices must communicate on a regular basis. The interface associated with an ARP entry determines which VRF the ARP entry belongs to. However, the additional management involved in adding and maintaining static ARP cache entries must also be taken into account.

An ARP entry is defined by the following parameters:

- IP address
- MAC address
- Type
- Interface

The **arp** command is used to configure static ARP entries on a nondefault VRF interface. (An ARP index is not required before a static ARP is configured.) The **arp** command is available in the address-family mode for a particular VRF.

Proxy ARP

Proxy ARP allows a Layer 3 switch to answer ARP requests from devices on one subnet on behalf of devices in another network. Proxy ARP is configured globally and can be further configured per interface. Interface-level configuration overrides the global configuration.

With the **proxy-arp** command configured, a router does not respond to ARP requests for IP addresses in the same subnet as the incoming ports. The **local-proxy-arp** command permits the router to respond to ARP requests for IP addresses within the same subnet and to forward all traffic between hosts in the subnet. The **local-proxy-arp** command is an interface-level configuration that has no VRF-related impact.

ARP rate limiting

ARP rate limiting is configured globally and applies to all VRFs.

ARP age can be configured globally and on a Layer 3 interface. An ARP age timer configured on a Layer 3 interface overrides the global configuration for ARP aging. The aging timer ensures that the ARP cache does not retain learned entries that are no longer valid.

Dynamic ARP inspection

Dynamic ARP Inspection (DAI) enables the Ruckus device to intercept and examine all ARP request and response packets in a subnet and to discard packets with invalid IP-to-MAC address bindings. DAI can prevent common man-in-the-middle (MiM) attacks such as ARP cache poisoning and can prevent the misconfiguration of client IP addresses. DAI allows only valid ARP requests and responses to be forwarded, and supports Multi-VRFs with overlapping address spaces. For more information on DAI, refer to the *FastIron Ethernet Switch Security Configuration Guide*.

DHCP snooping

Dynamic Host Configuration Protocol (DHCP) snooping enables a Ruckus device to filter untrusted DHCP IPv4 or IPv6 packets in a subnet. DHCP snooping can ward off 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 resulting from the user misconfiguration of DHCP servers. DHCP snooping supports Multi-VRFs. For more information on configuring DHCP IPv4 or IPv6 snooping to support a Multi-VRF instance, refer to the *FastIron Ethernet Switch Security Configuration Guide*.

IP Source Guard

You can use IP Source Guard (IPSG) together with DAI on untrusted ports. The Ruckus implementation of the IP Source Guard feature supports configuration on a port, on specific VLAN memberships on a port (for Layer 2 devices only), and on specific ports on a virtual Ethernet (VE) interface (for Layer 3 devices only). For more information on IPSG, refer to the *FastIron Ethernet Switch Security Configuration Guide*.

Configuring Multi-VRF

Configuring VRF system-max values

Use this example procedure to modify the default system-max values to accommodate Multi-VRF on a Ruckus ICX 7450.

The default **system-max** value must be configured because the device does not have routing table space for user VRFs.

In this example, two user VRFs are configured with 512 maximum routes on each VRF. The *ip-route-default-vrf* and *ip-route-vrf* values must be modified. The **write memory** and **reload** commands are required after the modification.

Once the device has rebooted after the reload, enter the **show default values** command to display the **system-max** settings.

1. Verify the default values.

```
device(config)# show default values
sys log buffers:50         mac age time:300 sec       telnet sessions:5
ip arp age:10 min          bootp relay max hops:4     ip ttl:64 hops
ip addr per intf:24
               :
               :
System Parameters     Default    Maximum    Current    Configured
ip-arp                4000       64000      4000       4000
ip-static-arp         512        6000       512        512
pim-mcache            1024       4096       1024       1024
               :
               :
ip-route              12000      15168      12000      12000
ip-static-route       64         2048       64         64
               :
               :
ip-vrf                16         16         16         16
ip-route-default-vrf  12000      15168      12000      12000
ip6-route             5120       5120       5120       5120
ip6-route-default-vr  5120       5120       5120       5120
ip6-route-vrf         100        5120       100        100

device(config)#
```

2. Change the maximum number of routes, save the configuration, and reload the device.

```
device(config)# system-max ip-route-default-vrf 10000
Total max configured ipv4 routes are 12000
  - Max ipv4 routes configured for default VRF are 10000
  - Max ipv4 routes available for all non-default VRFs are 2000
Warning: Please revalidate these values to be valid for your configuration.
Reload required.  Please write memory and then reload or power cycle.
device(config)#
device(config)# system-max ip-route-vrf 512
Reload required.  Please write memory and then reload or power cycle.
device(config)#
device(config)# exit
device# write memory
Write startup-config done.
device# Flash Memory Write (8192 bytes per dot) .
Flash to Flash Done.
device# reload
Are you sure? (enter 'y' or 'n'): Rebooting(0)...
y
```

3. Confirm the modified values.

```
device(config)# show default values
sys log buffers:50         mac age time:300 sec       telnet sessions:5
ip arp age:10 min          bootp relay max hops:4     ip ttl:64 hops
ip addr per intf:24
               :
               :
System Parameters    Default    Maximum    Current    Configured
ip-arp               4000       64000      4000       4000
ip-static-arp        512        6000       512        512
pim-mcache           1024       4096       1024       1024
               :
               :
ip-route             12000      15168      12000      12000
ip-static-route      64         2048       64         64
               :
               :
ip-vrf               16         16         16         16
ip-route-default-vrf 12000      15168      10000      10000
ip6-route            5120       5120       5120       5120
ip6-route-default-vr 5120       5120       5120       5120
ip6-route-vrf        100        5120       100        100
device(config)#
```

Creating VLANs as links on a tagged port for security

Where Multi-VRF is used, for example, in an enterprise data center, trusted servers or devices should be allowed to communicate directly, and untrusted ones should not be allowed to communicate directly at all. This optional task configures tagged Layer 3 interfaces to support secure VRF instances.

1. In global configuration mode, create a VLAN.

```
device(config)# vlan 10
device(config-vlan-10)#
```

2. Use the **tagged** command to identify the interface as secure.

```
device(config-vlan-10)# tagged e 1/1/1
```

3. Repeat the previous step on the corresponding interface on the peer device.

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. Create a VRF instance.

```
device(config)# vrf corporate
```

2. 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)# 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 26 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**

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-e1000-1/7/1)# no vrf forwarding1
All existing IP and IPv6 address will be removed from port 1/7/1
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 static ARP for Multi-VRF

The interface associated with an ARP entry determines to which VRF the ARP entry belongs.

An ARP entry is defined by the following parameters:

- IP address
- MAC address
- Type
- Interface

1. The following example illustrates how to configure static ARP on default VRFs on an Ethernet interface.

```
device(config)# arp 192.168.1.100 0000.2344.2441 eth 1/7/1
```

2. The following example illustrates how to configure static ARP on nondefault VRFs.

NOTE
The **arp** command can be used to configure static-ARP entries on a nondefault VRF interface. The VRF command does not require an ARP index before a static-ARP is configured. The **arp** command is available in the address-family mode for a particular VRF.

```
device(config)#
device(config)# vrf customer-1
device(config-vrf-customer-1)# address-family ipv4
device(config-vrf-customer-1-ipv4)# arp 1.1.1.1 0004.8044.5566 ethernet 1/7/8
device(config-vrf-customer-1-ipv4)# exit-address-family
device(config-vrf-customer-1)# exit-vrf
device(config)#
```

Configuring additional ARP features for Multi-VRF

This section discusses options for configuring proxy ARP and ARP rate limiting.

Proxy ARP allows a Layer 3 switch to answer ARP requests from devices on one subnet on behalf of devices in another network. Proxy ARP is configured globally and can be further configured per interface. Interface-level configuration overrides the global configuration.

With the **proxy-arp** command configured, a router does not respond to ARP requests for IP addresses in the same subnet as the incoming ports. The **local-proxy-arp** command permits the router to respond to ARP requests for IP addresses within the same subnet and to forward all traffic between hosts in the subnet. The **local-proxy-arp** command is an interface-level configuration that has no VRF-related impact.

ARP rate limiting is configured globally and applies to all VRFs.

ARP age can be configured globally and on a Layer 3 interface. An ARP age timer configured on a Layer 3 interface overrides the global configuration for ARP aging. The aging timer ensures that the ARP cache does not retain learned entries that are no longer valid.

To configure proxy ARP globally:

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

To configure proxy ARP on a Layer 3 Ethernet interface:

```
device(config)# int e1000 1/7/1
device(config-if-e1000-1/7/1)# local-proxy-arp
```

To configure ARP rate limiting globally:

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

To configure ARP rate limiting on a Layer 3 Ethernet interface for an aging timeout of 20 minutes:

```
device(config)# int e1000 1/7/1
device(config-if-e1000-1/7/1)# ip arp-age 20
```

Unicast Reverse Path Forwarding

Unicast Reverse Path Forwarding

The unicast Reverse Path Forwarding check is used to avoid source IP-based spoofing and a malformed source IP address.

A number of common types of denial-of-service (DoS) attacks, including Smurf and Tribe Flood Network (TFN), can take advantage of forged or rapidly changing source IP addresses to allow attackers to thwart efforts to locate or filter the attacks. Reverse Path Forwarding (RPF) is designed to prevent such an attacker from spoofing a source IP address by checking that the source IP address specified for a packet is received from a network to which the device has access. Packets with invalid source IP addresses are not forwarded. RPF is supported for IPv4 and IPv6 packets. Differences in RPF support between IPv4 and IPv6 are noted within this section where necessary. RFC 3704, Ingress Filtering for Multihomed Networks, covers various aspects of the source IP address being spoofed in traffic being forwarded.

FastIron devices support two unicast Reverse Path Forwarding (uRPF) modes according to RFC 3704:

- Strict mode: In this mode, all incoming packets are tested against the forwarding information base (FIB). If the incoming interface is not the best reverse path, the packet check fails. Failed packets are discarded by default. Source IP (SIP) lookup and the SIP next hop layer interface information is used in this mode. This mode has options to include default route check or exclude default route check. Including the default route check is the default configuration mode. Use the **rpf-mode strict** command for this mode. To exclude the default route check, you must include the option to urpf-exclude-default after entering the **rpf-mode strict** command.
- Loose mode: In this mode, each incoming packet's source address is tested against the forwarding information base. As long as there is a match for the source IP address in the forwarding information base, the traffic is allowed. Next hop interface information is not used in this mode. The packet is dropped only if the source address is not reachable through any interface on that router. This mode has options to include or exclude the default route check. Including the default route check is the default configuration mode. Use the **rpf-mode loose** command for this mode. To exclude the default route check, you must include the option to **urpf-exclude-default** after entering the command **rpf-mode loose** explicitly.

Configuration considerations for uRPF

The following configuration considerations apply to unicast Reverse Path Forwarding (uRPF) on supported Ruckus devices.

The following are general considerations for uRPF:

- uRPF works on the Layer 3 interface level (Layer 3 physical interface or Layer 3 VE interface).
- uRPF is VRF-aware.

- If a VLAN has multiple ports, the uRPF check will not identify packets coming in from different ports within the same VLAN, because a VLAN is considered as having a single Layer 3 interface.
- uRPF can be configured along with PBR, ACLs, routing protocol configurations, and multicast configurations.
- uRPF is not supported on tunnel interfaces.
- Tunnel keep-alive packets will be dropped in the hardware if uRPF is configured.
- uRPF must not be configured on devices where group-VE, tunnel keep-alive packets, or OpenFlow is configured.
- Counters or logging information is unavailable for uRPF hits.
- After enabling reverse path check, you must reload the device for uRPF to be programmed.
- Tunnel over user VRF should not be configured on a device on which uRPF is enabled.

ICX 7850, ICX 7750, ICX 7650, ICX 7450, and ICX 7250 considerations

- ICX 7850, ICX 7750 and ICX 7650 devices support global configuration mode and interface configuration mode.
- Per-interface level configuration is available on VE interfaces and physical ports only.
- IPv4 and IPv6 unicast routed packets are subjected to uRPF check on ICX 7850, ICX 7750, and ICX 7650 devices.
- Scaling numbers are reduced by half for the following system values when uRPF is enabled: ip-route, ip6-route, ip-route-default-vrf, ip6-route-default-vrf, ip-route-vrf, ip6-route-vrf.
- uRPF and MCT should not be configured together.
- If the number of ECMP paths for a route is more than 8, the hardware automatically chooses to use loose mode check, despite the configuration on the incoming interface.
- If the interface is not uRPF-enabled, the traffic is not subjected to uRPF check.
- If the interface is uRPF-enabled, both IPv4 and IPv6 traffic is subjected to uRPF check.

Unicast Reverse Path Forwarding feasibility

The following table provides support information about uRPF.

NOTE
uRPF is not supported on the ICX 7150.

TABLE 27 uRPF Feasibility

Device	Configurable mode	ECMP route supported	Default route lookup control	Non-Tunneled		Tunneled	
				IPv4	IPv6	IPv4	IPv6
ICX 7850, ICX 7750, ICX 7650, ICX 7450, ICX 7250	Strict mode (Interface configuration)	Yes	Yes	Yes	Yes	No	No
	Loose mode (Interface configuration)	NA	Yes	Yes	Yes	No	No

NOTE
In strict mode (interface configuration), if the number of ECMP paths for a route is more than eight, the hardware will apply loose mode check for the SIP check, even if the interface is configured as strict mode.

System-max Changes and uRPF

The following tables describe the system-max values with and without uRPF configured on the device. Note that the values with uRPF configuration after reload are reduced by half.

NOTE
uRPF is not supported on the ICX 7150.

NOTE
For the ICX 7850, values for IP routes and IPv6 routes are used from the forwarding profile and there is no option to change these values. When a new profile is selected, the values from the new profile are applied. Refer to the **forwarding-profile** command in the *Ruckus FastIron Command Reference* and the *Configuration Fundamentals* chapter in the *Ruckus FastIron Management Configuration Guide* for more information.

TABLE 28 ICX 7750, and ICX 7650 system-max values without uRPF configuration

System parameter	Default	Maximum	Current	Configured
ip-route	98304	131072	98304	98304
ip6-route	5120	7168	5120	5120
ip-route-default-vrf	65536	131072	65536	65536
ip6-route-default-vrf	2048	7168	2048	2048
ip-route-vrf	4096	131072	4096	4096
ip6-route-vrf	1024	7168	1024	1024

TABLE 29 ICX 7750 system-max values with uRPF configuration after reload

System parameter	Default	Maximum	Current	Configured
ip-route	49152	65536	49152	49152
ip6-route	2560	3584	2560	2560
ip-route-default-vrf	32768	65536	32768	32768
ip6-route-default-vrf	1024	3584	1024	1024
ip-route-vrf	2048	65536	2048	2048
ip6-route-vrf	512	3584	512	512

TABLE 30 ICX 7250 system-max values without uRPF configuration

System parameter	Default	Maximum	Current	Configured
ip-route	12000	12000	12000	12000
ip6-route	730	2048	730	730

TABLE 31 ICX 7250 system-max values with uRPF configuration after reload

System parameter	Default	Maximum	Current	Configured
ip-route	6000	6000	6000	6000

TABLE 31 ICX 7250 system-max values with uRPF configuration after reload (continued)

System parameter	Default	Maximum	Current	Configured
ip6-route	365	1024	365	365

TABLE 32 ICX 7450 system-max values without uRPF configuration

System parameter	Default	Maximum	Current	Configured
ip-route	12000	15168	12000	12000
ip6-route	5120	5120	5120	5120
ip-route-default-vrf	12000	15168	12000	12000
ip6-route-default-vrf	5120	5120	5120	5120
ip-route-vrf	1024	15168	1024	1024
ip6-route-vrf	100	5120	100	100

TABLE 33 ICX 7450 system-max values with uRPF configuration after reload

System parameter	Default	Maximum	Current	Configured
ip-route	6000	7584	6000	6000
ip6-route	2584	3572	2584	2584
ip-route-default-vrf	6000	7584	6000	6000
ip6-route-default-vrf	2560	2560	2560	2560
ip-route-vrf	512	7584	512	512
ip6-route-vrf	50	2560	50	50

Enabling unicast Reverse Path Forwarding

Unicast Reverse Path Forwarding can be enabled in different modes.

Both strict or loose modes can be configured when you globally enable uRPF on FastIron devices. uRPF is not supported on tunnel interfaces. When uRPF is enabled on a VE interface or a physical interface with an IP address configured, the prefixes learned over these uRPF-enabled interfaces will be checked with the uRPF criteria. On FastIron ICX devices, the uRPF check enables the interface level CLI and hardware settings. You should reload the device after enabling reverse path check for this configuration to be captured in the system settings.

1. Enter global configuration mode.

```
device# configure terminal
```

2. Enter the **reverse-path-check** command.

```
device(config)# reverse-path-check
```

The following example enables uRPF at the global level.

```
device# configure terminal
device(config)# reverse-path-check
```

Configuring unicast Reverse Path Forwarding modes

You can configure the various uRPF modes on a Layer 3 VE or physical interface.

You must enable uRPF forwarding globally before you enable the required forwarding modes.

1. Enter global configuration mode.

```
device# configure terminal
```

2. Enter interface configuration mode.

```
device(config)# interface ethernet 1/1/9
```

3. Enter the **rpf-mode** command followed by the required mode (**strict** or **loose**) you want to configure on the device. You can optionally use the exclude default route check (urpf-exclude-default) on the physical interface.

The following example shows the uRPF strict mode enabled.

```
device# interface ethernet 1/1/9
device(interface ethernet 1/1/9)# rpf-mode strict
```

Enabling uRPF check on PE ports

To enable uRPF check, PE ports must be part of a VE interface. You cannot configure uRPF on physical PE ports.

You must enable uRPF globally using the **reverse-path-check** command before configuring uRPF on PE ports.

1. Enter global configuration mode.

```
device# configure terminal
```

2. Enter VLAN configuration mode.

```
device(config)# vlan 460
device(config-vlan-460)#
```

3. Add tagged port 17/1/33 to port-vlan 460.

```
device(config-vlan-460)# tagged ethernet 17/1/33
```

4. Enter the **router-interface** command to create virtual interface 460.

```
device(config-vlan-460)# router-interface ve 460
        device(config-vlan-460)# interface ve 460
```

5. Enter the **rpf-mode strict** command.

```
device(config-vif-460)# rpf-mode strict
```

The other RPF modes that you can configure are **rpf-mode loose**, **rpf-more strict exclude default**, and **rpf-mode loose exclude default**.

6. Enter the **show run interface ve** command to verify the RPF mode configured on the device.

```
device(config-vif-460)# show run interface ve 460
    interface ve 460
    rpf-mode strict
    !
```

Ruckus Wireless, Inc., a wholly owned subsidiary of ARRIS International plc.
350 West Java Dr., Sunnyvale, CA 94089 USA
www.ruckuswireless.com