Extreme OS ONE Switching and Routing v22.2.3.0 Layer 3 Configuration Guide

Routing, BGP, EVPN, and VxLAN Implementation

9041117-00 Rev AA
August 2026

Legal Notice

Extreme Networks, Inc. reserves the right to make changes in specifications and other information contained in this document and its website without prior notice. The reader should in all cases consult representatives of Extreme Networks to determine whether any such changes have been made.

The hardware, firmware, software or any specifications described or referred to in this document are subject to change without notice.

Trademarks

Extreme Networks® and the Extreme Networks logo are trademarks or registered trademarks of Extreme Networks, Inc. in the United States and/or other countries.

All other names (including any product names) mentioned in this document are the property of their respective owners and may be trademarks or registered trademarks of their respective companies/owners.

For additional information on Extreme Networks trademarks, see: https://www.extremenetworks.com/about-extreme-networks/company/legal/trademarks

Open Source Declarations

Some software files have been licensed under certain open source or third-party licenses. End-user license agreements and open source declarations can be found at: https://www.extremenetworks.com/support/policies/open-source-declaration/

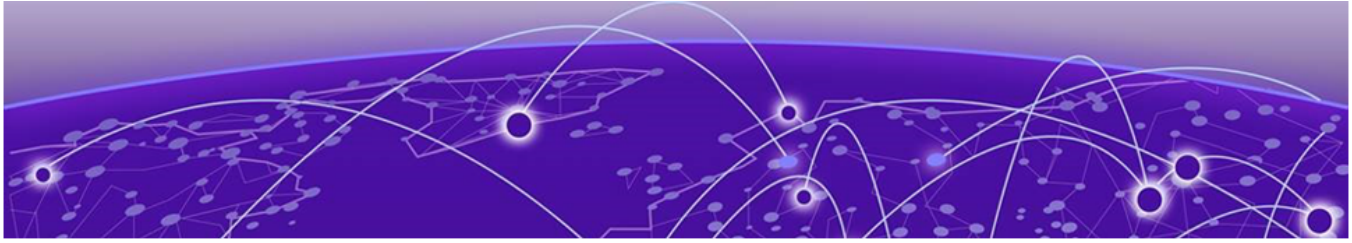
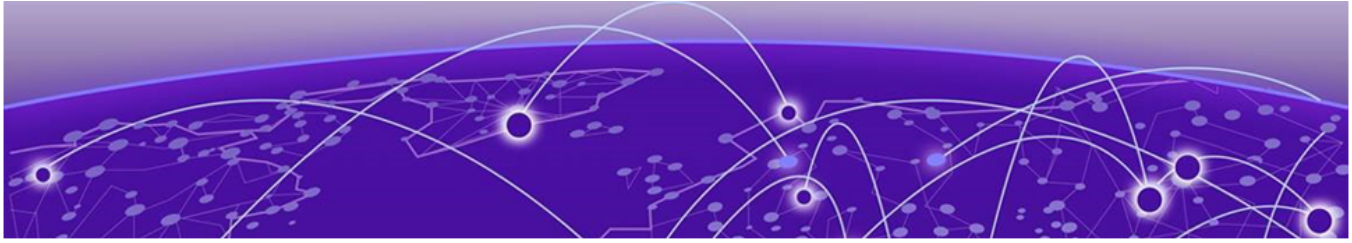


Table of Contents



Abstract

The *Extreme OS ONE Switching and Routing Layer 3 Configuration Guide* version 22.2.3.0provides comprehensive technical instructions for deploying advanced Layer 3 routing architectures. This guide covers IPv4 and IPv6 addressing, static and dynamic routing (BGP), policy-based routing, VRRP gateway redundancy, EVPN for IP fabrics, VxLAN, and resilient hashing mechanisms. Designed for experienced network engineers and architects, it provides practical guidance for implementing scalable and resilient routing solutions on Extreme OS ONE platforms.

Preface

Read the following topics to learn about:

- The meanings of text formats used in this document.
- Where you can find additional information and help.
- How to reach us with questions and comments.

Text Conventions

Unless otherwise noted, information in this document applies to all supported environments for the products in question. Exceptions, like command keywords associated with a specific software version, are identified in the text.

When a feature, function, or operation pertains to a specific hardware product, the product name is used. When features, functions, and operations are the same across an entire product family, such as Extreme Networks switches, the product is referred to as *the switch*.

Table 1: Notes and warnings

Icon	Notice type	Alerts you to...
	Tip	Helpful tips and notices for using the product
	Note	Useful information or instructions
	Important	Important features or instructions
	Caution	Risk of personal injury, system damage, or loss of data
	Warning	Risk of severe personal injury

Table 2: Text

Convention	Description
`screen displays`	This typeface indicates command syntax, or represents information as it is displayed on the screen.
The words *enter* and *type*	When you see the word *enter* in this guide, you must type something, and then press the Return or Enter key. Do not press the Return or Enter key when an instruction simply says *type*.
Key names	Key names are written in boldface, for example **Ctrl** or **Esc**. If you must press two or more keys simultaneously, the key names are linked with a plus sign (+). Example: Press **Ctrl**+**Alt**+**Del**
Words in italicized type	Italics emphasize a point or denote new terms at the place where they are defined in the text. Italics are also used when referring to publication titles.
NEW!	New information. In a PDF, this is searchable text.

Table 3: Command syntax

Convention	Description
bold text	Bold text indicates command names, keywords, and command options.
italic text	Italic text indicates variable content.
`[ ]`	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, such as passwords, are enclosed in angle brackets.
...	Repeat the previous element, for example, *member*[*member*...].
\	In command examples, the backslash indicates a “soft” line break. When a backslash separates two lines of a command input, enter the entire command at the prompt without the backslash.

Documentation and Training

Find Extreme Networks product information at the following locations:

Current Product Documentation
Release Notes
Hardware and Software Compatibility for Extreme Networks products
Extreme Optics Compatibility
Other Resources such as articles, white papers, and case studies

Open Source Declarations

Some software files have been licensed under certain open source licenses. Information is available on the Open Source Declaration page.

Training

Extreme Networks offers product training courses, both online and in person, as well as specialized certifications. For details, visit the Extreme Networks Training page.

Help and Support

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

Extreme Portal

Search the GTAC (Global Technical Assistance Center) knowledge base; manage support cases and service contracts; download software; and obtain product licensing, training, and certifications.

The Hub

A forum for Extreme Networks customers to connect with one another, answer questions, and share ideas and feedback. This community is monitored by Extreme Networks employees, but is not intended to replace specific guidance from GTAC.

Call GTAC

For immediate support: (800) 998 2408 (toll-free in U.S. and Canada) or 1 (408) 579 2800. For the support phone number in your country, visit www.extremenetworks.com/support/contact.

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

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

Subscribe to Product Announcements

You can subscribe to email notifications for product and software release announcements, Field Notices, and Vulnerability Notices.

1. Go to The Hub.
2. In the list of categories, expand the **Product Announcements** list.
3. Select a product for which you would like to receive notifications.
4. Select **Subscribe**.
5. To select additional products, return to the **Product Announcements** list and repeat steps 3 and 4.

You can modify your product selections or unsubscribe at any time.

Send Feedback

The User Enablement team at Extreme Networks has made every effort to ensure that this document is accurate, complete, and easy to use. We strive to improve our documentation to help you in your work, so we want to hear from you. We welcome all feedback, but we especially want to know about:

- Content errors, or confusing or conflicting information.
- Improvements that would help you find relevant information.
- Broken links or usability issues.

To send feedback, email us at Product-Documentation@extremenetworks.com.

Provide as much detail as possible including the publication title, topic heading, and page number (if applicable), along with your comments and suggestions for improvement.

About This Document

What's New in This Document

Feature	Description	Link
Configurable Internal IP Range	Added support for operator-configurable internal IPv4/IPv6 ranges to avoid conflicts with customer network subnets. Includes CLI utility for configuration management and boot-time validation.	Configurable Internal IP Range on page 19
VRRP Gateway Redundancy	Added support for Virtual Router Redundancy Protocol version 3 (VRRPv3) as defined in RFC 9568 to enable automatic failover of gateway routers and increase availability and reliability of routing paths. Features include sub-second failover capability (1–3 seconds) for IPv4 environments, virtual IP (VIP) and virtual MAC (VMAC) address support, automatic Active/Backup router election based on priority, OpenConfig YANG model-based configuration and management, gNMI operational state visibility, transparent failover with no end-host reconfiguration required, and line-rate traffic forwarding with instant MAC learning via gratuitous ARP.	VRRP Gateway Redundancy on page 86

Feature	Description	Link
BGP Large Communities	Updated the topic "Configuring Routing Policies" to add support for BGP large communities, which are a 96-bit BGP path attribute used to tag, filter, and manipulate routing policies across the Internet to solve the limitations of standard communities by fully supporting modern 4-byte Autonomous System Numbers (ASNs)	Configuring Routing Policies on page 93
BGP Route Aggregation	Added a "BGP Route Aggregation" chapter to describe how to aggregate routes from a range of networks into a single network prefix for a specific BGP address family, which combines multiple specific IP addresses into a single, broader route advertisement instead of announcing numerous individual routes across the Internet	BGP Route Aggregation on page 137

For more information, see the *Extreme OS ONE Switching and Routing Release Notes*.

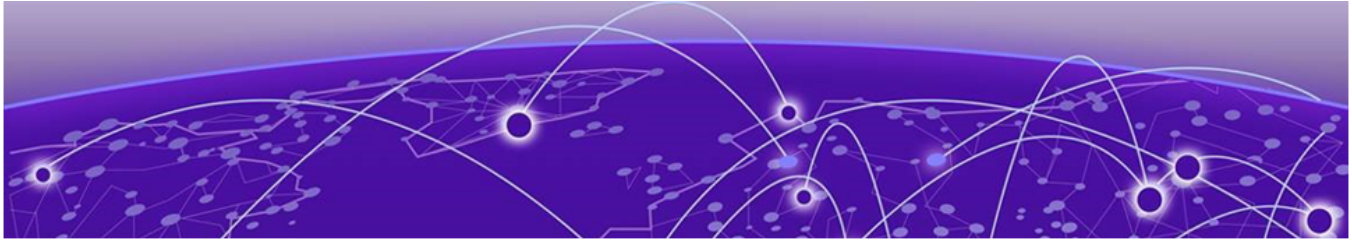
Supported Platforms

Extreme OS ONE Switching and Routing 22.2.3.0 and later releases support Extreme 8520, Extreme 8720, Extreme 8730, and Extreme 8820 hardware platforms.

Note

Although Extreme Networks® tests and supports many software and hardware configurations for this release, this document's scope does not include all possible configurations and scenarios.

For information about other releases, see the documentation for those releases.



IPv4 Addressing

The following topics describe how to configure IPv4 addressing on a device that is running Extreme OS ONE.

IPv4 Addressing Overview

IPv4 uses a 32-bit addressing system in packet switched networks. Extreme OS ONE enables IPv4 routing on devices that operate at Layer 3. You cannot disable IPv4 routing.

IPv4 is an Internet protocol that delivers packets of data from a source to a destination across an interconnected system of networks. IPv4 is a fixed length 32-bit addressing system that uses a 4-byte dotted decimal format: x.x.x.x.

IP uses four main mechanisms to provide service:

- **Type of Service (ToS)** — Quality of Service (QoS) required for a specific traffic type or network and configures a higher priority to be given to voice traffic, for example, that is more sensitive to dropped packets.
- **Time to Live (TTL)** — Time period for which a packet can exist before it reaches its final destination. If the TTL expires before the packet reaches its destination, the packet is destroyed. The period is set by the packet sender.

- **Options** — Control mechanisms such as timestamps, security, and other special routing functions that are optional.
- **Header Checksum** — Verification that the packet contents transmitted correctly. If the checksum algorithm fails, the packet is dropped immediately.

An IP address has two sections:

- **Network** — Network on which the device is configured.
- **Host** — Host device.

IPv4 does not provide a reliable communication function. No acknowledgments are sent, and the only error control is the header checksum. There are no flow control mechanisms or retransmissions. Internet Control Message Protocol (ICMP) can be used to report any errors.

IP interfaces

Extreme OS ONE devices that operate at Layer 3 allow IPv4 addresses to be configured on the following types of interfaces (and subinterfaces):

- Ethernet ports
- VE interfaces
- Port channel (LAG) interfaces
- Loopback interfaces

You can configure up to 128 IP addresses on each interface (or subinterface).

Note
After you configure a port as part of a bridge domain, you cannot configure Layer 3 interface parameters on that port. The parameters must be configured on the appropriate virtual routing interface.

ARP Cache

The ARP cache contains entries that map IP addresses to MAC addresses.

Entries to the Address Resolution Protocol (ARP) cache are added from:

- Devices that are directly attached to the Layer 3 device.
- An interface based static IP route that goes to a destination two or more router hops away. The MAC address is either of the destination device of the router interface answering an ARP request on behalf of the device, using proxy ARP.

The ARP cache can contain both dynamic (learned) entries and static (user-configured) entries. The software places an entry in the ARP cache:

- **Dynamic** — When the Layer 3 device learns a device MAC address from an ARP request or ARP reply from the device.
- **Static** — When the interface with a proper IP address comes up.

The ARP cache also contains the ARP entries learned from MLAG and EVPN.

For more information about ARP, see ARP Overview on page 30.

Configurable Internal IP Range

The Configurable Internal IP Range feature allows operators to customize the private IPv4 and IPv6 address space used by the embedded Kubernetes cluster to avoid conflicts with customer production networks.

Internal IP Range Overview

Extreme OS ONE runs its application workloads inside an embedded single-node K3S (lightweight Kubernetes) cluster that operates entirely within the device. The K3S cluster requires three private address ranges:

- **Cluster (pod) CIDR** — The address pool from which pod IP addresses are allocated by the Container Network Interface (CNI)
- **Service CIDR** — The virtual address pool for Kubernetes ClusterIP services
- **Node IP** — The address to which the control plane and kubelet bind. On Extreme OS ONE, this address is hosted on an internal `tap0` interface

Previously, these ranges were fixed in the device image (IPv4 cluster 10.42.0.0/16, service 10.43.0.0/16, node IP 10.52.1.52, and equivalent IPv6 space fd42::/46). If a customer's production network used any of these subnets, traffic collisions would occur, resulting in:

- Loss of in-band management connectivity
- Broken inter-pod communication
- Potential blackholing of internal traffic

The Configurable Internal IP Range feature resolves these conflicts by externalizing the internal addressing into an operator-editable YAML configuration file.

Key Benefits

- **Network Conflict Avoidance** — Relocate internal addressing to avoid collision with customer subnets
- **Centralized Configuration** — Single YAML file (/mnt/onl/config/IPs/internal.yaml) controls all internal addressing
- **Consistent Sub-range Allocation** — Component sub-ranges are expressed relative to the overall range, so moving the overall range coherently relocates all derived addresses
- **Boot-time Application** — Configuration is validated and applied during early device initialization
- **Dual-Stack Support** — Independent IPv4 and IPv6 overall ranges are supported

Feature Components

- **Configuration File** — Persistent, operator-editable YAML file that describes the overall internal range and how it is carved into component ranges
- **Apply Engine** — Boot-time process (`apply_internal_ips`) that reads the configuration, computes and validates the component ranges, and applies them to relevant components
- **Checksum-gated Apply** — Configuration is re-applied only when it has actually changed, avoiding unnecessary K3S runtime restarts
- **K3S Apply Component** — Dedicated process (`k3s_ip_apply`) that programs K3S service flags, regenerates the `tap0` interface, updates /etc/hosts, and propagates the node IP
- **CLI Utility** — Generic query and update utility (`internal_ip_range`) with validation and user-friendly error messages

Configuration Validation

The Configurable Internal IP Range feature validates all configuration to ensure consistency and prevent deployment errors. The system enforces the following rules:

- The overall range must be large enough to contain all configured component allocations
- The overall range must be aligned to its own prefix length
- IPv4 and IPv6 ranges must be specified in CIDR notation (for example, 10.40.0.0/14)
- Component allocations must not overlap within the overall range

Invalid configurations are rejected by the CLI and are not applied at boot time, ensuring the device operates with a consistent and valid addressing scheme.

System Impact and Limitations

When you change the internal IP range, the following behavior occurs:

- **K3S Cluster Reset** — Changing the internal range resets the embedded K3S cluster state and re-initializes it with the new addressing scheme
- **Reboot Required** — Changes to the internal range do not take effect until the next device reboot
- **Non-persisted State Loss** — Any non-persisted in-cluster state is rebuilt after the reboot that applies the change

Note
The Configurable Internal IP Range feature is not intended for frequent reconfiguration. Configure the internal range during initial device deployment or during planned maintenance windows to minimize service disruption.

Configure the Internal IP Range

Configure the overall IPv4 and IPv6 address ranges used by the embedded K3S cluster to avoid conflicts with customer production networks.

Before you begin, ensure that:

- You have root privileges on the device.
- You have identified IPv4 and IPv6 ranges that do not conflict with the customer network.
- Each range is specified in CIDR notation and is aligned to its prefix.
- Each range is large enough to contain the configured component allocations.
- You understand that the changes take effect after the next device reboot.

The internal addressing configuration uses one overall range for each address family. The system derives the cluster CIDR, service CIDR, and node IP from the configured overall ranges.

Use the **`internal_ip_range`** utility to query and update the configuration. The utility validates changes before writing them to the persistent configuration file.

Note
Changing an internal range resets the embedded K3S cluster state when the new configuration is applied during reboot. Non-persisted in-cluster state is rebuilt.

1. Display the current overall internal IP ranges.

   ```
   device# internal_ip_range show
   ```

   The command displays the current IPv4 and IPv6 ranges, their address spans, and the minimum block size required by the configured component allocations.
2. Configure the overall IPv4 internal range, if required.

   Replace *`NETWORK/PREFIX`* with the required IPv4 range in CIDR notation.

   ```
   device# internal_ip_range set --ipv4 NETWORK/PREFIX
   ```

   For example:

   ```
   device# internal_ip_range set --ipv4 10.128.0.0/14
   ```
3. Configure the overall IPv6 internal range, if required.

   Replace *`NETWORK/PREFIX`* with the required IPv6 range in CIDR notation.

   ```
   device# internal_ip_range set --ipv6 NETWORK/PREFIX
   ```

   For example:

   ```
   device# internal_ip_range set --ipv6 fd50::/46
   ```
4. Alternatively, configure both address families in one command.

   ```
   device# internal_ip_range set --ipv4 10.128.0.0/14 --ipv6 fd50::/46
   ```

5. Display the derived node IP addresses, if required.

   Run the command without an option to display both node IP addresses.

   ```
   device# internal_ip_range node-ip
   ```

   To display only one address family, use one of the following commands:

   ```
   device# internal_ip_range node-ip --ipv4
   device# internal_ip_range node-ip --ipv6
   ```

6. Reboot the device.

   The updated ranges are validated and applied during early device initialization. If the addressing changed, the embedded K3S cluster state is reset and re-initialized with the new addressing.

7. After the device reboots, verify the applied configuration.

   ```
   device# internal_ip_range show
   ```

   Confirm that the displayed overall IPv4 and IPv6 ranges match the configured values.

The device uses the configured overall IPv4 and IPv6 ranges for its embedded K3S cluster and derives the associated cluster CIDR, service CIDR, and node IP values.

Verify in-band management connectivity and confirm that embedded applications are operational after the reboot.

Internal IP Range CLI Commands and Error Messages

The `internal_ip_range` utility provides commands to query and configure the internal IPv4 and IPv6 address ranges used by the embedded Kubernetes cluster.

Command Syntax

```
internal_ip_range <subcommand> [<options>]
```

Subcommands:

- `show`—Display current internal IP range configuration
- `set`—Update internal IP range configuration
- `node-ip`—Display derived node IP addresses

show Subcommand

Display the current overall IPv4 and IPv6 internal IP ranges.

Syntax:

```
internal_ip_range show
```

Output Example:

```
IPv4 overall range: 10.40.0.0/14
IPv6 overall range: fd42::/46
```

Description:

The `show` command displays the current overall internal IP range configuration that is stored in the persistent configuration file. This command does not require root privileges and can be executed by any user.

set Subcommand

Update the overall IPv4 or IPv6 internal IP range. This command requires root privileges.

Syntax:

```
internal_ip_range set <family> <cidr_range>
```

Parameters:

Parameter	Description
`<family>`	Address family: `ipv4` or `ipv6`
`<cidr_range>`	CIDR notation for the overall range (for example, 10.40.0.0/14 or fd42::/46)

Example:

```
internal_ip_range set ipv4 10.50.0.0/14
internal_ip_range set ipv6 fd50::/46
```

Restrictions:

- Command requires root privileges
- Range must be specified in CIDR notation
- Range must be large enough for all configured component allocations (minimum /14 for default allocations)
- Range must be aligned to its prefix length
- Changes take effect only after a device reboot

node-ip Subcommand

Display the node IP addresses derived from the configured internal IP ranges.

Syntax:

```
internal_ip_range node-ip
```

Output Example:

```
IPv4 node IP: 10.40.0.1
IPv6 node IP: fd42:0:2::1
```

Description:

The `node-ip` command displays the IPv4 and IPv6 node IP addresses that are derived from the configured overall ranges. The node IP is the address to which the Kubernetes control plane and kubelet bind and is hosted on the internal `tap0` interface. This command does not require root privileges.

Error Messages and Validation

The `internal_ip_range` utility validates all input and provides clear error messages when invalid values are provided. The following table lists common error conditions and messages:

Error Condition	Example Message
Range too small for configured allocations	`ipv4 range 10.42.0.0/16 is too small: a /16 was given, but the configured component allocations need at least a /14 (a larger block). Use a prefix of /14 or wider.`
Network not aligned to its prefix	`'10.40.0.5/14' is not aligned to its prefix; did you mean 10.40.0.0/14?`
Wrong address family for the parameter	`'fd42::/46' is an IPv6 network but an ipv4 value was expected`
Malformed CIDR notation	`'10.40.0.0' must be given as NETWORK/PREFIX, e.g. 10.40.0.0/14`
Permission denied (for set command)	`must be run as root to modify the configuration`

Event Log Messages

When the device boots or applies internal IP range configuration changes, events are logged in the system event log. The following table describes the event log messages related to the Configurable Internal IP Range feature:

Event	Log Message
Configuration unchanged at boot	`apply_internal_ips: configuration unchanged since last apply, skipping`
Invalid configuration (missing overall range)	`apply_internal_ips: invalid config: missing 'overall' section`
Invalid overall range	`apply_internal_ips: invalid 'overall' range: <error_detail>`
Applying component configuration	`apply_internal_ips: applying component 'k3s' via k3s_ip_apply`
Partial failure (checksum not recorded)	`apply_internal_ips: apply incomplete, not recording checksum (will retry next run)`
K3S addressing changed	`k3s_ip_apply: k3s addressing changed, resetting k3s state`
K3S addressing unchanged	`k3s_ip_apply: k3s addressing unchanged, skipping state reset`

Configuration File

The internal IP range configuration is stored in the following persistent file:

```
/mnt/onl/config/IPs/internal.yaml
```

This YAML file describes the overall internal range and how it is carved into component sub-ranges. The file is seeded on first boot from an image-supplied default and is retained across firmware upgrades.

Although the configuration file can be edited directly, Extreme Networks recommends using the `internal_ip_range set` command to ensure proper validation and formatting. Hand-edited files are validated at boot time and are rejected if they contain syntax errors or invalid values.

Note
Do not edit the configuration file directly unless you are familiar with YAML syntax and understand the internal addressing scheme. Invalid hand-edited files can cause boot failures or connectivity issues.

Virtual Routing and Forwarding (VRF)

You use VRFs to set up multiple independent routing tables within a single Layer 3 device. VRFs route traffic between VLANs assigned to the VRF, simulating multiple networks on one router. This provides enhanced security and separation.

VRF configuration requires the following items:

- Unicast: Provides IPv4 VPN instance on the VRF (required for other features)
- Multicast: Provides Layer 3 VSN IP Multicast over Fabric Connect for the VRF (required for other features)
- Direct Route: Sets up route redistribution for the VRF
- Default Gateway: Configures a default gateway for the VRF
- DvR Redistribution: Controls route redistribution over DvR

Sample VRF Configuration

The following example shows a typical configuration for a VRF (in this case, the default VRF). In this example, the **`member`** (VRF configuration) command attaches 12 interfaces to the VRF. All IPv4 and IPv6 addressing becomes active only when interfaces are attached to a VRF:

```
device# show running-config vrf default-vrf

vrf default-vrf
  member ethernet 0/15
  member ethernet 0/15 vlan 4094
  member ethernet 0/21
  member ethernet 0/29 vlan 4094
  member port-channel 1
  member port-channel 16
  member port-channel 16 vlan 1,4094
  member port-channel 26
  member port-channel 26 vlan 4094
  member port-channel 46
  member  ve 600,800,7000
  member  loopback 1-2,1024
  ecmp-max-path 32
  resilient-hash
```

```
  table-connections
    src-protocol connected dst-protocol bgp address-family ipv4
  !
  evpn-instance tenant_1
    nve nve1
    ignore-as
    bridge-domain 11,1001,2000,4000,4500,4901,5000,6000,7000,8192
  static-route 192.0.2.163/32 192.0.2.1
  !
  router bgp
    local-as 65001
    router-id 198.51.100.254
    graceful-restart
    use-multiple-paths ebgp maximum-paths 10
    address-family ipv4 unicast
      graceful-restart
      network 192.0.2.0/24
      network 198.51.100.0/24
      network 203.0.113.6/32
      activate
    !
    address-family ipv6 unicast
      network 2001:db8:1::/64
      network 2001:db8:2::/64
      network 2001:db8:3::1/128
      activate
    !
    address-family l2vpn evpn
      graceful-restart
      activate
    !
    peer-group group1
      remote-as 65124
      enable-bfd
      address-family ipv6 unicast
        activate
      !
    !
    peer-group group2
      remote-as 65001
      nvo isp1_nvo
      enable-bfd
      address-family ipv4 unicast
        activate
      !
      address-family l2vpn evpn
        activate
      !
    !
    peer-group group3
      remote-as 65124
      nvo isp1_nvo
      enable-bfd
      address-family ipv4 unicast
        activate
      !
      address-family l2vpn evpn
        nexthop-unchanged
        activate
      !
    !
  !
!
device#
```

IP Addressing

IP addressing is scoped within each VRF as follows:

- Each VRF has its own IP address space and Layer 3 interfaces.
- IP addresses are assigned per VRF, allowing reuse of the same IP in different VRFs without conflict.
- IP interfaces are bound to a specific VRF (for example, `interface vlan 100 vrf Customer-A`).

Static Routing

Static routes are created and maintained separately per VRF. Each VRF has its own routing table, and routes are not visible across VRFs unless leaked.

ECMP Maximum Paths

You use ECMP to set up multiple paths to forward traffic to the same destination. ECMP support in VRF provides load balancing and redundancy within a VRF and distributes traffic across multiple paths to the same destination.

The following example shows the **`ecmp-max-path`** command, which configures the ECMP feature to set the number of paths:

```
device(config)# vrf default-vrf
device(config-vrf-default-vrf)# ecmp-max-path
  (2-128)  Specify the maximum ECMP paths (range: 2-128, power of 2, default: 8)
device(config-vrf-default-vrf)#
```

Resilient Hashing

Resilient hashing ensures consistent traffic distribution across ECMP paths, even when paths change. With VRF and ECMP, it ensures consistent traffic distribution across ECMP paths and minimizes traffic disruption and maintains predictable traffic redistribution.

Combining VRF, ECMP, and Resilient Hashing creates a scalable and resilient network infrastructure to support complex routing requirements.

Load Balancing (Hashing)

Load balancing (hashing) is a method of distributing traffic between interfaces that are tied to the same group to use all the links efficiently. The goal is to provide both redundancy and increased throughput while ensuring that packets within the same "flow" are not reordered.

Configure Ethernet Load Balancing

Ethernet load balancing is done via the Ethernet header fields such as SRC MAC, DST MAC, Ether Type, and VLAN. The Ethernet load balancing configuration is applied globally (device-wide).

All Ethernet load balancing options are enabled for Ethernet by default. To disable one or more options, use the **no** form of the **load-balance ethernet** command.

For details about the commands in this section, see the *Extreme OS ONE Switching and Routing Command Reference Guide*.

1. From privileged EXEC mode, access global configuration mode.

```
device# configure terminal
```

2. Configure Ethernet load balancing parameters on the device. You can specify one or more of the following parameters in any combination and in any order: { [**dst-mac**] [**etype**] [**src-mac**] [**vlan**] }

```
device(config)# load-balance ethernet dst-mac etype src-mac vlan
```

3. Verify the configuration.

```
device(config)# do show running-config load-balance

load-balance ethernet dst-mac src-mac vlan etype
load-balance ipv4 dst-ip dst-l4-port protocol src-ip src-l4-port
load-balance ipv6 dst-ip dst-l4-port nxt-hdr src-ip src-l4-port
!
device#
```

Configure Layer 3 Load Balancing

Layer 3 IPv4 load balancing is done via the Layer 3 IPv4 header fields such as SRC IP, DST IP, and Protocol. Layer 3 IPv6 load balancing is performed using the Layer 3 IPv6 header fields such as SRC IPV6, DST IPV6 and Nxt-hdr. The Layer 3 load balancing configuration is applied globally (device-wide).

All load balancing options are enabled for Layer 3 (IPv4 and IPv6) by default. To disable one or more options, use the **no** form of the **load-balance ipv4** command or the **load-balance ipv6** command respectively.

For details about the commands in this section, see the *Extreme OS ONE Switching and Routing Command Reference Guide*.

1. From privileged EXEC mode, access global configuration mode.

```
device# configure terminal
```

2. Configure IPv4 load balancing parameters on the device. You can specify one or more of the following parameters in any combination and in any order: { [**dst-ip**] [**protocol**] [**src-ip**] }

```
device(config)# load-balance ipv4 dst-ip protocol src-ip
```

3. Configure IPv6 load balancing parameters on the device. You can specify one or more of the following parameters in any combination and in any order: { [**dst-ip**] [**next-hdr**] [**src-ip**] }

```
device(config)# load-balance ipv6 dst-ip nxt-hdr src-ip
```

4. Verify the configuration.

```
device(config)# do show running-config load-balance

load-balance ethernet dst-mac src-mac vlan etype
load-balance ipv4 dst-ip dst-l4-port protocol src-ip src-l4-port
load-balance ipv6 dst-ip dst-l4-port nxt-hdr src-ip src-l4-port
!
device#
```

Configure Layer 4 Load Balancing

All load balancing options are enabled for Layer 4 (IPv4 and IPv6) by default. To disable one or more options, use the **no** form of the **load-balance ipv4** command or the **load-balance ipv6** command respectively.

For details about the commands in this section, see the *Extreme OS ONE Switching and Routing Command Reference Guide*.

1. From privileged EXEC mode, access global configuration mode.

```
device# configure terminal
```

2. Configure IPv4 load balancing parameters on the device. You can specify one or more of the following parameters in any combination and in any order: { [**dist-l4-port**] [**protocol**] [**src-l4-port**] }

```
device(config)# load-balance ipv4 dist-l4-port protocol src-l4-port
```

3. Configure IPv6 load balancing parameters on the device. You can specify one or more of the following parameters in any combination and in any order: { [**dist-l4-port**] [**next-hdr**] [**src-l4-port**] }

```
device(config)# load-balance ipv6 dist-l4-port nxt-hdr src-l4-port
```

4. Verify the configuration.

```
device(config)# do show running-config load-balance

load-balance ethernet dst-mac src-mac vlan etype
load-balance ipv4 dst-ip dst-l4-port protocol src-ip src-l4-port
load-balance ipv6 dst-ip dst-l4-port nxt-hdr src-ip src-l4-port
!
device#
```

IP Parameters and Protocols

Most IP parameters are dynamic. They take effect as soon as you run the CLI command. You can verify that a dynamic change has taken effect by displaying the running configuration. To do so, use the **show running-config** command.

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

The system disables the BGP4 route exchange protocol and multicast protocols (IGMP) by default.

Address Resolution Protocol

The following topics describe how to configure Address Resolution Protocol (ARP). These topics include instructions for setting how long a device considers another device to be reachable after successfully contacting it as well as instructions for creating static ARP entries to ensure reliable, secure, and controlled IP-to-MAC mapping (useful for security, troubleshooting, legacy support, or network enforcement).

ARP Overview

The Address Resolution Protocol (ARP) maps IPv4 network addresses to MAC hardware addresses.

When forwarding traffic, a device needs to know the destination MAC address because each IP packet is encapsulated in an Ethernet frame. The MAC address is needed for the packet's final destination and for a next hop toward the destination.

A device first searches its ARP cache, and a match for the IP address supplies the corresponding MAC address. Otherwise, the device broadcasts an ARP request. Network devices receive the requests, and the host with a matching IP address sends an ARP reply that includes its MAC address.

After the device receives a matching ARP reply, the packet is sent toward its destination, and the IP address/MAC address pair is added to the ARP cache as a dynamic ARP entry.

Configure ARP Reachable Time for Remote IPv4 Nodes

You can configure the duration (in seconds) that a router considers a remote IPv4 node to be reachable. You can configure ARP reachable time for Ethernet, port channel (LAG), and Virtual Ethernet (VE) interfaces as well as Ethernet and port channel subinterfaces.

To do so, you use the **`ipv4 neighbor reachable-time`** command to configure the ARP IPv4 neighbor reachable-time configuration on an interface. ARP establishes correspondences between pairs of network-layer (Layer 3) addresses and LAN hardware addresses (Layer 2 MAC addresses).

Router advertisement messages include a *reachable time* value. All nodes on a link use the same value.

Reachable time is how long a device considers another device to be reachable after successfully contacting it without re-verifying its presence. When a device sends traffic to a neighbor and receives a response, it marks that neighbor as reachable.

The reachable time is a timer that specifies how long the entry stays in the reachable state before the device must probe or refresh the neighbor information. If no

additional communication happens during the reachable time, the device transitions the neighbor entry to a stale state and eventually needs to verify it again using an ARP request.

Extreme Networks recommends that you configure a long duration, because a short duration causes IPv4 network devices to process the information at a greater frequency.

1. Access global configuration mode.

   ```
   device# configure terminal
   ```

2. Access the interface on which you are changing the reachable time.

   ```
   device(config)# interface ethernet 0/1
   ```

3. Specify the new reachable time value.

   ```
   device(config-if-eth-0/1)# ipv4 neighbor reachable-time 300
   ```

   Valid values range from 30 through 3600 seconds. The default is 1200 seconds.

The following examples show how to configure ARP reachable time for Ethernet, port channel (LAG), and Virtual Ethernet (VE) interfaces as well as Ethernet and port channel subinterfaces respectively:

```
device# configure terminal
device(config)# interface ethernet 0/1
device(config-if-eth-0/1)# ipv4 neighbor reachable-time 300
device(config-if-eth-0/1)#

device# configure terminal
device(config)# interface port-channel 10
device(config-if-po-10)# ipv4 neighbor reachable-time 600
device(config-if-po-10)#

device# configure terminal
device(config)# interface ve 100
device(config-if-ve-100)# ipv4 neighbor reachable-time 1500
device(config-if-ve-100)#

device# configure terminal
device(config)# interface ethernet 0/1
device(config-if-eth-0/1)# subinterface vlan 100
device(config-subif-eth-0/1.100)# ipv4 neighbor reachable-time 2500
device(config-subif-eth-0/1.100)#

device# configure terminal
device(config)# interface port-channel 10
device(config-if-po-10)# subinterface vlan 200
device(config-subif-po-10.200)# ipv4 neighbor reachable-time 3600
device(config-subif-po-10.200)#
```

The following example displays the configuration of Ethernet interface 0/1 that is running currently on the device. In this example, the reachable time is set to 300 seconds on the interface:

```
device# show running-config interface ethernet 0/1

interface ethernet 0/1
  mtu 1600
  ipv4 neighbor reachable-time 300
  no shutdown
!
device#
```

Configure a Static ARP Entry for an Interface

A static entry in the IPv4 Address Resolution Protocol (ARP) cache ensures that an IPv4 neighbor is always reachable. You can create a static ARP entry for a device that is not yet connected to the network or to prevent an entry from aging out. You can configure static ARP entries for Ethernet, port channel (LAG), and Virtual Ethernet (VE) interfaces as well as Ethernet and port channel subinterfaces.

1. Access global configuration mode.

   ```
   device# configure terminal
   ```

2. Specify the interface to contain the static ARP entry.

   ```
   device(config)# interface ethernet 0/1
   ```

3. Create the static ARP entry for the neighbor.

   ```
   device(config-int-eth-0/1)# ipv4 neighbor 192.0.2.0 3c:fd:fe:e4:xx:a8
   ```

   This example creates a static ARP entry for a neighbor with IPv4 address 1.1.1.10 and associates it with MAC address 3c:fd:fe:e4:37:a8, which is reachable through physical port Ethernet 0/1.

The following example displays the configuration of Ethernet interface 0/1 that is running currently on the device. In this example, the ARP IPv4 address and MAC address on the interface are set to 1.1.1.10 and 3c:fd:fe:e4:37:a8 respectively:

```
device# show running-config interface ethernet 0/1

interface ethernet 0/1
  ipv4 address 192.0.2.0/24
  ipv4 neighbor 192.0.2.1 3c:fd:fe:e4:yy:xx
  no shutdown
!
device#
```

The following examples show how to configure static ARP entries for Ethernet, port channel (LAG), and Virtual Ethernet (VE) interfaces as well as Ethernet and port channel subinterfaces respectively:

```
device# configure terminal
device(config)# interface ethernet 0/1
device(config-if-eth-0/1)# ipv4 neighbor 192.0.2.0 3c:fd:fe:ex:xx:a5
device(config-if-eth-0/1)#

device# configure terminal
device(config)# interface port-channel 10
device(config-if-po-10)# ipv4 neighbor 192.0.2.1 3c:fd:fe:cv:xx:a6
device(config-if-po-10)#

device# configure terminal
device(config)# interface ve 100
device(config-if-ve-100)# ipv4 neighbor 192.0.2.2 3c:fd:fe:e4:xx:ax
device(config-if-ve-100)#

device# configure terminal
device(config)# interface ethernet 0/1
device(config-if-eth-0/1)# subinterface vlan 100
device(config-subif-eth-0/1.100)# ipv4 neighbor 192.0.2.3 3c:fd:fe:ex:xx:a8
device(config-subif-eth-0/1.100)#

device# configure terminal
device(config)# interface port-channel 10
device(config-if-po-10)# subinterface vlan 200
```

```
device(config-subif-po-10.200)# ipv4 neighbor 192.0.2.4 3c:fd:ff:ee:xx:aa
device(config-subif-po-10.200)#
```

Assign an IPv4 Address to a Loopback Interface

IPv4 addresses can be assigned to a loopback interface via Classless Interdomain Routing (CIDR) network masks. Loopback interfaces add stability to a network, because they do not incur route flap problems due to unstable links between devices.

IPv4 routing is enabled by default on Extreme OS ONE devices that operate at Layer 3 and cannot be disabled. You must assign IP addresses to interfaces on the devices to allow IPv4 based protocols to work across the network.

1. From privileged EXEC mode, access global configuration mode.

   ```
   device# configure terminal
   ```

2. Access the interface to which you are assigning the IP addresses.

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

3. Assign an IP address to the interface.

   Note
   You can define only one IP address per loopback. The only valid mask value is /32.

   ```
   device(config-if-lo-1)# ipv4 address 192.0.2.0/32
   ```

4. Activate the interface.

   ```
   device(config-if-lo-1)# no shutdown
   ```

5. Add the interface to a VRF.

   ```
   device(config-if-lo-1)# exit
   device(config)# vrf default-vrf
   device(config-vrf-default-vrf)# member loopback 1
   ```

6. Verify that the IP address is assigned to the interface.

   ```
   device(config-vrf-default-vrf)# do show ipv4 interface loopback 1

   Interface: Lo 1 Admin-status:UP Oper-status:UP
   IP MTU: 1500
   Vrf:default-vrf
   Address     Status
   ========    ========
    192.0.2.0/32 PREFERRED
   device(config-vrf-default-vrf)#
   ```

Assign IPv4 addresses to Non-Loopback Interfaces

You can assign IPv4 addresses (primary or secondary) to interfaces or subinterfaces by using Classless Interdomain Routing (CIDR) network masks. You can assign IPv4 addresses tor Ethernet, port channel (LAG), and Virtual Ethernet (VE) interfaces as well as Ethernet and port channel subinterfaces.

1. Access global configuration mode.

   ```
   device# configure terminal
   ```

2. Access the interface to which you are assigning the IP addresses.

```
device(config)# interface ethernet 0/1
```

3. Assign one or more IP addresses with a CIDR network mask.

```
device(config-if-eth-0/1)# ipv4 address 192.0.2.0/24
device(config-if-eth-0/1)# ipv4 address 192.0.2.1/24
```

   This example assigns IPv4 address 192.0.2.0 and CIDR network mask /24 to interface Ethernet 0/1 and also assigns IPv4 address 192.0.2.1 and CIDR network mask /24 to the same interface.

4. To assign a secondary IPv4address with a CIDR network mask, include the **secondary** keyword.

   Note
   You can configure a secondary IPv4 address only if the primary IPv4 address is already configured in the same subnet.

```
device(config-if-eth-0/1)# ipv4 address 192.0.2.2/24 secondary
```

5. Activate the interface.

```
device(config-if-eth-0/1)# no shutdown
```

6. Add the interface to a VRF.

```
device(config-if-eth-0/1)# exit
device(config)# vrf red
device(config-vrf-red)# member loopback 1
```

7. Verify that the IPv4 addresses are assigned to the interface.

```
device(config-vrf-red)# do show ipv4 interface ethernet 0/1

Interface: Eth 0/1 Admin-status:UP Oper-status:UP
IP MTU: 1500
Vrf: red
Address                    Status
========                   ========
 192.0.2.0/24              PREFERRED
 192.0.2.1/24             PREFERRED
 192.0.2.2/24 (secondary) PREFERRED
device#
```

The following example displays the configuration of Ethernet interface 0/1 that is running currently on the device. In this example, IPv4 address 1.1.1.1 and CIDR network mask /24 are assigned to interface Ethernet 0/1:

```
device# show running-config interface ethernet 0/1

interface ethernet 0/1
  ipv4 address 192.0.2.0/24
  ipv4 neighbor 192.0.2.1 3c:fd:fe:e4:xx:a8
  no shutdown
!
device#
```

The following examples show how to configure IPv4 addresses with CIDR network masks to Ethernet, port channel (LAG), and Virtual Ethernet (VE) interfaces as well as Ethernet and port channel subinterfaces respectively:

```
device# configure terminal
device(config)# interface ethernet 0/1
```

```
device(config-if-eth-0/1)# ipv4 address 192.0.2.0/24
device(config-if-eth-0/1)#

device# configure terminal
device(config)# interface port-channel 10
device(config-if-po-10)# ipv4 address 192.0.2.1/24
device(config-if-po-10)#

device# configure terminal
device(config)# interface ve 100
device(config-if-ve-100)# ipv4 address 192.0.2.2/24
device(config-if-ve-100)#

device# configure terminal
device(config)# interface ethernet 0/1
device(config-if-eth-0/1)# subinterface vlan 100
device(config-subif-eth-0/1.100)# ipv4 address 192.0.2.3/24
device(config-subif-eth-0/1.100)#

device# configure terminal
device(config)# interface port-channel 10
device(config-if-po-10)# subinterface vlan 200
device(config-subif-po-10.200)# ipv4 address 192.0.2.4/24
device(config-subif-po-10.200)#
```

The following examples show how to use the secondary keyword to configure secondary IPv4 addresses with CIDR network masks to Ethernet, port channel (LAG), and Virtual Ethernet (VE) interfaces as well as Ethernet and port channel subinterfaces respectively:

```
device# configure terminal
device(config)# interface ethernet 1/1
device(config-if-eth-1/1)# ipv4 address 192.0.2.5/24 secondary
device(config-if-eth-1/1)#

device# configure terminal
device(config)# interface port-channel 5
device(config-if-po-5)# ipv4 address 192.0.2.6/24 secondary
device(config-if-po-5)#

device# configure terminal
device(config)# interface ve 50
device(config-if-ve-50)# ipv4 address 192.0.2.7/24 secondary
device(config-if-ve-50)#

device# configure terminal
device(config)# interface ethernet 0/2
device(config-if-eth-0/2)# subinterface vlan 50
device(config-subif-eth-0/2.50)# ipv4 address 192.0.2.8/24 secondary
device(config-subif-eth-0/2.50)#

device# configure terminal
device(config)# interface port-channel 4
device(config-if-po-4)# subinterface vlan 75
device(config-subif-po-4.75)# ipv4 address 192.0.2.9/24 secondary
device(config-subif-po-4.75)#
```

IPv4 Management

This section describes the Ping and Traceroute tools. These are both network diagnostic tools, but they have different purposes and provide different insights into network connectivity. You use Ping to check if a host is up, quickly test latency, or verify DNS name resolution. You use Traceroute to troubleshoot slow or dropped connections, determine if a specific hop or route is causing issues, or map the path from source to destination.

IPv4 Ping

The **ping** command verifies connectivity from an Extreme OS ONE device to an IPv4 destination on a TCP/IP network. This command is a diagnostic tool used to test connectivity between your device and another network host. It tells you whether the target is reachable and how long it takes to respond.

This command sends a specified number of pings with configured parameters to the specified IPv4 destination device. Optional parameters include the datagram size, interface name, source address, time (in seconds) to wait for a response, and VRF instance name.

A ping displays information for the device as soon as the information is received. This includes the number of packets transmitted and received, percentage of packets lost, and elapsed time.

You can specify that the device display up to 1000 transmissions (pings). The range is 1 through 1000.

For more information about the command, including examples, see the *Extreme OS ONE Switching and Routing Command Reference Guide*.

The following example pings an IPv4 destination address:

```
device# ping 192.0.2.0 vrf mgmt-vrf

PING 192.0.2.0 (192.0.2.0) 56(84) bytes of data.
64 bytes from 192.0.2.0 icmp_seq=1 ttl=54 time=304 ms
64 bytes from 192.0.2.0 icmp_seq=2 ttl=54 time=304 ms
64 bytes from 192.0.2.0 icmp_seq=3 ttl=54 time=303 ms
device#
```

The following example pings an IPv4 destination address in quiet mode:

```
device# ping 192.0.2.0 vrf mgmt-vrf quiet

PING 192.0.2.0 (192.0.2.0) from 192.0.2.1 default-vrf: 56(84) bytes of data.

--- 192.0.2.0 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4089ms
device#
```

IPv4 Traceroute

The **traceroute** command displays the path of network packets from a Extreme OS ONE device to an IPv4 host. Traceroute is a network diagnostic tool used to track the path that packets take from your device to a destination IP address or hostname. It shows each hop (router or gateway) that the packet goes through to help identify where delays or failures occur in the network.

A traceroute displays information for each hop as soon as the information is received. Optional parameters include an interface name, minimum and maximum Time to Live (TTL) values in a number of hops, source address, time (in seconds) to wait for a response, and VRF instance name.

You can use the **traceroute** command to configure a minimum TTL setting between 1 and 255 hops (the default is 1 hop). You can also use it to configure a maximum TTL setting between 1 and 255 hops (the default is 30 hops).

The device displays up to three responses when there are multiple equal-cost routes to the destination.

The following example executes an IPv4 traceroute:

```
device# traceroute 192.0.2.0

traceroute to 192.0.2.0 (192.0.2.0), 30 hops max, 60 byte packets
 1  192.0.2.0 (192.0.2.0)  0.075 ms  0.019 ms  0.017 ms
device#
```

The following example executes an IPv4 traceroute for a VRF named vrf1. The output shows two hops:

```
device# traceroute 192.0.2.0 vrf vrf1

traceroute to 192.0.2.0 (192.0.2.0), 30 hops max, 60 byte packets
 1  192.0.2.1 (192.0.2.1)  1.033 ms  0.950 ms  0.889 ms
 2  192.0.2.0 (192.0.2.0)  0.851 ms  0.809 ms  0.766 ms
device#
```

Delete an IP Address from an Interface

You can delete a specified IP address or all IP addresses from an interface or subinterface.

1. From privileged EXEC mode, access global configuration mode.

   ```
   device# configure terminal
   ```

2. Access the interface from which you are deleting the IP address.

   ```
   device(config)# interface ethernet 0/2
   ```

3. To delete a specific IP address from the interface, run the **no ipv4 address** command with the IP address and the mask.

   ```
   device(config-if-eth-0/2)# no ipv4 address 192.0.2.0/24
   ```

4. To delete all IP addresses from the interface, run the **no ipv4 address** command.

   ```
   device(config-if-eth-0/2)# no ipv4 address
   ```

The Domain Name System

The Domain Name System (DNS) is a hierarchical naming system that assigns a name (such as a company name) to an Internet entity to represent the real IP address of the entity. An entity can be a gateway router and is referred to as a domain.

A domain name (for example, extreme.router.com) can be used in place of an IP address for certain operations such as IP pings, traceroutes, and Telnet management connections to the router. A domain name is easier to remember than all of the numbers in an IP address. DNS has several components:

- DNS server: A DNS server stores the information about a DNS domain. DNS servers are a key element of DNS because they respond to queries against its database.

When a DNS domain is defined on this device to recognize all hosts in that domain, this device automatically appends the appropriate domain to the host address and forwards it to the DNS server.

- DNS resolver: In a Layer 2 or Layer 3 device, the DNS resolver 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 performs a DNS query, all hosts in 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 device automatically appends the appropriate domain to a host and forwards it to the DNS servers for resolution.
- DNS gateway addresses: Gateway IP addresses that are assigned to the device enable clients that are attached to the device to reach DNS.

Configure a DNS Domain and Gateway Addresses

You can configure a DNS domain and DNS gateway addresses to resolve host names to IP addresses.

1. Access global configuration mode.

```
device# configure terminal
```

2. Access system configuration mode.

```
device(config)# system
```

3. Access DNS configuration mode.

```
device(config-system)# dns
```

4. Configure one or more domain names.

```
device(config-system-dns)# domain-name mydevice1.com
device(config-system-dns)# domain-name mydevice2.com
```

5. Configure one or more DNS gateway addresses for DNS servers.

```
device(config-system-dns)# name-server 192.0.2.0
device(config-system-dns)# name-server 192.0.2.0
```

The first DNS server IP address added to the domain name configuration will be considered the primary DNS server. You can add up to five additional DNS servers for the domain name. Any combination of IPv4 or IPv6 DNS name servers can be configured. For example, you could add two IPv6 name servers alongside four IPv4 name servers. However, you cannot add more than six name servers for the domain.

If you add more than six name servers for the domain, the following error message appears:

```
too many 'ip dns name-server', 7 configured, at most 6 must be configured
```

6. Return to privileged EXEC mode.

```
device(config-system-dns)# end
```

7. (Optional) Verify the DNS configuration.

```
device# traceroute mydevice1.com

Sending DNS Query to 192.0.2.0
Tracing Route to IP node 192.0.2.1
To ABORT Trace Route, Please use stop-traceroute command.
```

```
Traced route to target IP node 192.0.2.0:
IP Address    Round Trip Time1    Round Trip Time2
192.0.2.1    93 msec             121 msec
```

The output shows that 192.0.2.0 is the IP address of the DNS server (default DNS gateway address), and 192.0.2.1 represents the IP address of the mydevice1.com host.

The following example configures six DNS name servers for the domain *www.mydevice1.com*. Of these six domain names, two are IPv6 DNS resolvers:

```
device# configure terminal
device(config)# system
device(config-system)# dns
device(config-system-dns)# domain-name www.mydevice1.com
device(config-system-dns)# name-server 192.0.2.0
device(config-system-dns)# name-server 192.0.2.1
device(config-system-dns)# name-server 192.0.2.2
device(config-system-dns)# name-server 2001:db8:0:53::1
device(config-system-dns)# name-server 2001:db8:0:53::2
device(config-system-dns)# name-server 192.0.2.3
device(config-system-dns)#
```

Configure the Source IP Address for Various Packet Types

When a device originates a packet of one of the following types, the default source address of the packet is the lowest-numbered IP address on the interface that sends the packet:

- Telnet
- TACACS or TACACS+
- TFTP
- RADIUS
- Syslog
- SMTP

You can configure the device to always use the lowest-numbered IP address on a specific Ethernet, loopback, or Virtual Ethernet (VE) interface as the source addresses for these packets. When configured, the device uses the same IP address as the source for all packets of the specified type regardless of the ports that actually sends the packets.

Designating a source IP address provides the following benefits:

- If your server is configured to accept packets only from specific IP addresses, you can configure the device to always send the packets from the same link or source address.
- If you specify a loopback interface as the single source for specified packets, servers can receive the packets regardless of the states of individual links. Thus, if a link to the server becomes unavailable but 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.

IPv4 ICMP Rate Limiting on the Management Interface

IPv4 Internet Control Message Protocol (ICMP) rate limiting lets you control and monitor IPv4 ICMP echo-request (ping) traffic on the management interface. This protects the device from ping flooding attacks and excessive IPv4 ICMP traffic that could impact device performance.

This feature ensures device manageability by allowing necessary diagnostics while dropping excessive traffic. It specifically throttles IPv4 ICMP traffic to allow safe troubleshooting without affecting management traffic such as SSH or HTTPS.

ICMP rate limiting on the management interface in IPv4 functions the same as it does in IPv6. For details about configuring and monitoring IPv4 ICMP rate limiting on the management interface, see the *Extreme OS ONE Switching and Routing Security Configuration Guide*.

IPv4 Addressing Show and Clear Commands

You can display and delete information related to IPv4 interfaces and routes.

Table 4 shows a list of **show** and **clear** commands that you can use to display or delete information about IPv4 interfaces and routes.

Table 4: IPv4 addressing show and clear commands

Command	Description
show ipv4 interface [**ethernet** *interface-name* | **loopback** *port-number* | **port-channel** *port-channel-number* | **tunnel** *interface-name* | **ve** *interface-name* | **internal** *interface-name*] [**brief**]	Displays the IPv4 address, status, and configuration for a specified Ethernet, loopback, or Virtual Ethernet (VE) interface. You can also display a brief summary of this information for all interfaces.
show ipv4 neighbor vrf { **all** | *vrf-name* } [*ipv4-address*]	Displays the address resolution protocol (ARP) entries in the neighbor caches for Virtual Routing and Forwarding (VRF) instances.
show ipv4 route vrf *vrf-name* [*ipv4-address* **/** *prefix-length*] [**connected** | **static** | **arp** | **bgp** | **brief**]	Displays IPv4 route information.
show counters icmp interface management *interface-name*	Displays the aggregated numbers of IPv4 ICMP packets that the system accepted or dropped because of ICMP rate limiting on the management interface. The only supported value for *interface-name* is 0 in this release. For details about configuring and monitoring IPv4 ICMP rate limiting on the management interface, see the *Extreme OS ONE Switching and Routing Security Configuration Guide*.

Table 4: IPv4 addressing show and clear commands (continued)

Command	Description
clear ipv4 route vrf *vrf-name* [*ipv4-address* **/** *prefix-length*]	Clears a specified IPv4 route or all IPv4 routes in the IP routing tables.
clear counters interface management *interface-name*	Clears statistics for the management interface. These include the aggregated numbers of IPv4 ICMP packets that the system accepted or dropped because of ICMP rate limiting. The only supported value for *interface-name* is 0 in this release. For details about configuring and monitoring IPv4 ICMP rate limiting on the management interface, see the *Extreme OS ONE Switching and Routing Security Configuration Guide*.

IPv6 Addressing

The following topics describe how to configure 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 more network devices.

An IPv6 address consists of 8 fields of 16-bit hexadecimal values separated by colons (:). Figure 1 shows the format for IPv6 addresses.

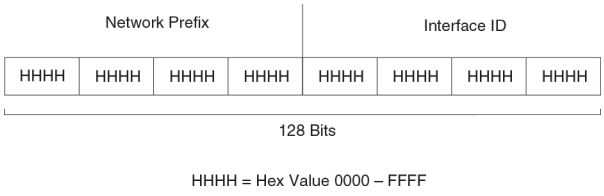


Figure 1: IPv6 address format

As shown in Figure 1, HHHH is a 16-bit hexadecimal value. H is a 4-bit hexadecimal value. An example of an IPv6 address is 2001:db8:0:200:2d:d0ff:fe48:4672.

An IPv6 address can include hexadecimal fields of zeros. To make the address manageable, you can:

- Omit the leading zeros. For example, 2001:db8:0:200:2d: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:db8::200:2d:d0ff:fe48:4672.

When specifying an IPv6 address in a command syntax, consider the following rules:

- 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 1 on page 42, the IPv6 network prefix consists of the leftmost bits of the address. As with an IPv4 address, you can specify the IPv6 prefix using the prefix/prefix-length format, for which the following rules apply:

- You specify the prefix parameter as 16-bit hexadecimal values separated by a colon.
- You specify the prefix-length parameter as a decimal value that indicates the network portion of the IPv6 address.

An example of an IPv6 prefix is 2001:db8:49ea:d088::/64.

IP interfaces

Extreme OS ONE devices that operate at Layer 3 can use IPv6 addresses that you configure on the following types of interfaces (and subinterfaces):

- Ethernet
- VE
- Port channel (LAG)
- Loopback

You can configure up to 128 IP addresses on each interface (or subinterface).

Note
After you configure a port as part of a bridge-domain, you cannot configure Layer 3 interface parameters on that port. The parameters must be configured on the appropriate virtual routing interface.

Assign an IPv6 Address to a Loopback Interface

IPv6 addresses can be assigned to a loopback interface via Classless Interdomain Routing (CIDR) network masks. Loopback interfaces add stability to a network, because they do not incur route flap problems due to unstable links between devices.

IPv6 routing is enabled by default on Extreme OS ONE devices that operate at Layer 3 and cannot be disabled. IP addresses must be assigned to interfaces on the devices to allow IPv6 based protocols to operate across the network.

1. From privileged EXEC mode, access global configuration mode.

```
device# configure terminal
```

2. Access the interface to which you are assigning the IP addresses.

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

3. Assign an IP address to the interface.

Note
You can define only one IP address per loopback. The only valid mask value is /128.

```
device(config-if-lo-1)# ipv6 2001:db8::100/128
```

4. Activate the interface.

```
device(config-if-lo-1)# no shutdown
```

5. Add the interface to a VRF.

```
device(config-if-lo-1)# exit
device(config)# vrf default-vrf
device(config-vrf-default-vrf)# member loopback 1
```

6. Verify that the IP address is assigned to the interface.

```
device(config-vrf-default-vrf)# do show ipv6 interface loopback 1

Interface: Lo 1 Admin-status:UP Oper-status:UP
IP MTU: 1500
Vrf:default-vrf
Address                 Status
========                ========
ipv6 2001:db8::100/128 PREFERRED
device#
```

Assign IPv6 Addresses to Non-Loopback Interfaces

IPv6 addresses (primary or secondary) can be assigned to interfaces or subinterfaces, using Classless Interdomain Routing (CIDR) network masks. You can assign interfaces tor Ethernet, port channel (LAG), and Virtual Ethernet (VE) interfaces as well as Ethernet and port channel subinterfaces.

Configuration of IPv6 addresses on an interface (or subinterface) has the following conditions:

- You can have multiple primary IPv6 addresses from different subnets, but not from the same subnet.
- Secondary IPv6 addresses must have the same subnet as the primary addresses.
- You cannot delete primary IPv6 addresses when you have configured a secondary address.
- You cannot add secondary IPv6 addresses when you have configured a primary address.

Perform the following steps to assign IPv6 addresses. The following example is for an Ethernet interface.

1. Enter global configuration mode.

```
device# configure terminal
```

2. Access the interface to which you are assigning the IP addresses.

```
device(config)# interface ethernet 0/2
```

3. Assign one or more primary IP addresses, including the CIDR network mask.

```
device(config-if-eth-0/2)# ipv6 address 2001:db8:102:102::1/64
device(config-if-eth-0/2)# ipv6 address 2001:db8:202:102::1/64
```

4. Assign one or more secondary IP addresses, including the CIDR network mask and the secondary keyword.

```
device(config-if-eth-0/2)# ipv6 address 2001:db8:102:102::2/64 secondary
device(config-if-eth-0/2)# ipv6 address 2001:db8:202:102::2/64 secondary
```

5. Activate the interface.

```
device(config-if-eth-0/2)# no shutdown
```

6. Add the interface to a VRF.

```
device(config-if-eth-0/2)# exit
device(config)# vrf red
device(config-vrf-red)# member loopback 1
```

7. Verify that the IP addresses are assigned to the interface.

```
device(config-vrf-red)# do show ipv6 interface ethernet 0/2

Interface: Eth 0/2 Admin-status:UP Oper-status:UP
IP MTU: 1500
Vrf: red
Address Status
======== ========
2001:db8:102:102::1/64 PREFERRED
2001:db8:202:102::1/64 PREFERRED
2001:db8:102:102::2/64 (secondary) PREFERRED
2001:db8:202:102::2/64 (secondary) PREFERRED
device#
```

The following example displays the configuration of Ethernet interface 0/1 that is running currently on the device. In this example, IPv6 address 2001:db8:102:102::1/64 and CIDR network mask /64 are assigned to interface Ethernet 0/1:

```
device# show running-config interface ethernet 0/1

interface ethernet 0/1
 ipv6 address 2001:db8:102:102::1/64
 ipv6 neighbor 2001:db8:102:102::9 3c:fd:fe:e4:37:a0
 no shutdown
!
device#
```

The following examples show how to configure IPv6 addresses with CIDR network masks to Ethernet, port channel (LAG), and Virtual Ethernet (VE) interfaces as well as Ethernet and port channel subinterfaces respectively:

```
device# configure terminal
device(config)# interface ethernet 0/1
device(config-if-eth-0/1)# ipv6 address 2001:db8:102:0::1/64
device(config-if-eth-0/1)#

device# configure terminal
device(config)# interface port-channel 10
device(config-if-po-10)# ipv6 address 2001:db8:102:1::1/64
device(config-if-po-10)#

device# configure terminal
device(config)# interface ve 100
device(config-if-ve-100)# ipv6 address 2001:db8:102:2::1/64
device(config-if-ve-100)#

device# configure terminal
device(config)# interface ethernet 0/1
device(config-if-eth-0/1)# subinterface vlan 100
device(config-subif-eth-0/1.100)# ipv6 address 2001:db8:102:3::1/64
device(config-subif-eth-0/1.100)#

device# configure terminal
device(config)# interface port-channel 10
device(config-if-po-10)# subinterface vlan 200
device(config-subif-po-10.200)# ipv6 address 2001:db8:102:4::1/64
device(config-subif-po-10.200)#
```

The following examples show how to use the secondary keyword to configure secondary IPv6 addresses with CIDR network masks to Ethernet, port channel (LAG), and Virtual Ethernet (VE) interfaces as well as Ethernet and port channel subinterfaces respectively:

```
device# configure terminal
device(config)# interface ethernet 0/2
device(config-if-eth-0/2)# ipv6 address 2001:db8:102:10::1/64 secondary
device(config-if-eth-0/2)#

device# configure terminal
device(config)# interface port-channel 5
device(config-if-po-5)# ipv6 address 2001:db8:102:11::1/64 secondary
device(config-if-po-5)#

device# configure terminal
device(config)# interface ve 50
device(config-if-ve-50)# ipv6 address 2001:db8:102:12::1/64 secondary
device(config-if-ve-50)#

device# configure terminal
device(config)# interface ethernet 0/2
device(config-if-eth-0/2)# subinterface vlan 50
device(config-subif-eth-0/2.50)# ipv6 address 2001:db8:102:13::1/64 secondary
device(config-subif-eth-0/2.50)#

device# configure terminal
device(config)# interface port-channel 4
device(config-if-po-4)# subinterface vlan 75
device(config-subif-po-4.75)# ipv6 address 2001:db8:102:14::1/64 secondary
device(config-subif-po-4.75)#
```

IPv6 Management

Ping and Traceroute are both network diagnostic tools, but they serve different purposes and provide different insights into network connectivity. You use Ping to check if a host is up, quickly test latency, or verify DNS name resolution. You use Traceroute to troubleshoot slow or dropped connections, determine if a specific hop or route is causing issues, or map the path from source to destination.

Note

On the management interface of an Extreme OS ONE device, the IPv6 routing functionality is not enabled.

IPv6 Ping

The **`ping ipv6`** command verifies connectivity from an Extreme OS ONE device to an IPv6 destination on a TCP/IP network. This command is a diagnostic tool used to test connectivity between your device and another network host. It tells you whether the target is reachable and how long it takes to respond.

This command sends a specified number of pings with configured parameters to the specified IPv6 destination device. Optional parameters include the datagram size, interface name, source address, time (in seconds) to wait for a response, and VRF instance name.

A ping displays information for the device as soon as the information is received. This includes the number of packets transmitted and received, percentage of packets lost, and elapsed time.

You can specify that the device display up to 1000 transmissions (pings). The range is 1 through 1000.

For more information about the command, including examples, see the *Extreme OS ONE Switching and Routing Command Reference Guide*.

The following example pings an IPv6 destination address:

```
device# ping ipv6 2001:db8:60:69bc:92:218:8bff:fe40:1470 vrf mgmt-vrf

ping 2001:db8:60:69bc:92:218:8bff:fe40:1470
PING 2001:db8:60:69bc:92:218:8bff:fe40:1470 (2001:db8:60:69bc:92:218:8bff:fe40:1470) from
2001:db8:60:69bc:92:218:8bff:fe40:1470 default-vrf: 54(82) bytes of data.

--- 2001:db8:60:69bc:92:218:8bff:fe40:1470 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 4192ms
device#
```

IPv6 Traceroute

The **`traceroute ipv6`** command displays the path of network packets from a Extreme OS ONE device to an IPv6 host. Traceroute is a network diagnostic tool used to track the path that packets take from your device to a destination IP address or hostname. It shows each hop (router or gateway) that the packet goes through to help identify where delays or failures occur in the network.

A traceroute displays information for each hop as soon as the information is received. Optional parameters include the interface name, minimum and maximum Time to Live (TTL) values in a number of hops, source address, time (in seconds) to wait for a response, and VRF instance name.

The device displays up to three responses when there are multiple equal-cost routes to the destination.

For more information about the command, including examples, see the *Extreme OS ONE Switching and Routing Command Reference Guide*.

The following example executes an IPv6 traceroute with minimum and maximum TTL values, a source IPv6 address, and a timeout value:

```
device# traceroute ipv6 2001:db8:60:69bc:92:218:8bff:fe40:1470
maxttl 128 minttl 30 source 2001:db8:60:69bc:92:205:33ff:fe9e:3f20 timeout 3

traceroute to 2001:db8:60:69bc:92:218:8bff:fe40:1470
(2001:db8:60:69bc:92:218:8bff:fe40:1470),
128 hops max, 80 byte packets
30  2001:db8:60:69bc:92:218:8bff:fe40:1470
(2001:db8:60:69bc:92:218:8bff:fe40:1470)  2.145 ms  2.118 ms  2.085 ms
device#
```

The following example executes an IPv6 traceroute for a VRF named vrf1. The output shows two hops:

```
device# traceroute ipv6 2001:db8::1 vrf vrf1

traceroute to 2001:db8::1 (2001:db8::1), 30 hops max, 80 byte packets
 1  2001:db8:10:1:3::2 (2001:db8:10:1:3::2)  1.117 ms  1.082 ms  1.056 ms
 2  2001:db8::1 (2001:db8::1)  2.103 ms  2.108 ms  2.102 ms
device#
```

IPv6 Neighbor Discovery

The Neighbor Discovery feature for IPv6 uses IPv6 ICMP messages to perform the following tasks:

- Determines the link-layer address of a neighbor on the same link
- Verifies that a neighbor is reachable
- Tracks neighbor routers

An IPv6 host must 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

IPv6 Neighbor Discovery features that you can adjust include the following:

- Neighbor solicitation messages for duplicate address detection
- Router advertisement messages, which include the following configuration:
  - Interval between router advertisement messages
  - Value that indicates a router is advertised as a default router (for use by all nodes on a link)
  - Prefixes advertised in router advertisement messages
  - Flags for host stateful autoconfiguration

Note
For all solicitation and advertisement messages, Extreme OS ONE uses seconds as the unit of measure instead of milliseconds.

Note
Neighbor Discovery is not supported on tunnel interfaces.

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) lets the host 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.

Configure IPv6 Router Advertisement

You can configure the interval for sending router advertisements, the router advertisement lifetime, the hop limit, and several other settings. As a best practice, ensure that the interval between router advertisement transmission is less than or equal to the router lifetime value if the router is advertised as a default route.

IPv6 router advertisement has the following limitations or unsupported features:

- *RFC 6104 Rogue IPv6 Router Advertisement Problem Statement*
- *RFC 6105 IPv6 Router Advertisement Guard*
- *RFC 6106 IPv6 Router Advertisement Options for DNS Configuration*
- Origination of router solicitation
- IPV4 router advertisement

1. Access global configuration mode.

```
device# configure terminal
```

2. Access interface configuration mode.

```
device# interface ethernet 0/1
```

3. Access IPv6 router advertisement configuration mode.

```
device(config-if-eth-0/1)# ipv6-router-advertisement
```

4. (Optional) Configure the allowed maximum number of hops for the IPv6 router advertisement.

```
device(config-if-eth-0/1-ipv6-router-advertisement)# hop-limit 133
```

   The default hop limit is 64 hops.

5. (Optional) Configure the lifetime for the IPv6 router advertisement.

```
device(config-if-eth-0/1-ipv6-router-advertisement)# lifetime 200
```

   The default lifetime is 1800 seconds. If you set the lifetime to 0, the router is not advertised as a default router.
6. (Optional) Enable the managed address configuration flag for IPv6 router advertisement.

```
device(config-if-eth-0/1-ipv6-router-advertisement)# managed-config-flag
```

   The managed address configuration flag is disabled by default. When this feature is enabled, the managed address configuration (M) flag is set in the router advertisement. This flag indicates that there are addresses available via DHCPv6.
7. (Optional) Disable the transmission of unsolicited messages for IPv6 router advertisement.

```
device(config-if-eth-0/1-ipv6-router-advertisement)# mode disable-unsolicited-ra
```

   Transmission of unsolicited router advertisement messages is enabled by default.
8. (Optional) Enable the other configuration (O) flag for IPv6 router advertisement.

```
device(config-if-eth-0/1-ipv6-router-advertisement)# other-config
```

   The O flag is disabled by default. When this feature is enabled, the O flag is set in the advertised router advertisement. The O flag indicates that there is other configuration available via DHCPv6 (such as DNS servers).
9. (Optional) Configure the maximum and minimum intervals for IPv6 router advertisement.

```
device(config-if-eth-0/1-ipv6-router-advertisement)# interval 100 min 50
```

   These are the intervals during which router advertisement messages are sent randomly.

   The maximum interval is 600 seconds by default. The minimum interval is .33 x the maximum interval, if the maximum interval is nine seconds or more, by default (otherwise, the default equals the maximum interval).
10. Enable IPv6 router advertisement on the interface.

```
device(config-if-eth-0/1-ipv6-router-advertisement)# enable
```

11. Configure a prefix for IPv6 router advertisement.

```
device(config-if-eth-0/1-ipv6-router-advertisement)# prefix 2001:db8:100::1/64
device(config-if-ipv6-router-advertisement-prefix-2001:db8:100::1/64)# advertisement
device(config-if-ipv6-router-advertisement-prefix-2001:db8:100::1/64)#
autoconfiguration
device(config-if-ipv6-router-advertisement-prefix-2001:db8:100::1/64)# onlink
device(config-if-ipv6-router-advertisement-prefix-2001:db8:100::1/64)# preferred-
lifetime 5000
device(config-if-ipv6-router-advertisement-prefix-2001:db8:100::1/64)# valid-lifetime
1000
```

    Advertisement is enabled by default. This makes the prefix get advertised.

Autoconfiguration is optional and is enabled by default. When autoconfiguration is disabled, the prefix will not be used for stateless address configuration. This is achieved by setting the autonomous address configuration bit for the prefix.

Onlink is optional and is enabled by default. When onlink is enabled, the prefix is marked as being "on link" via the L flag bit for the prefix within a router advertisement. This flag tells hosts that the prefix is reachable directly on the local link. This means that packets to destinations within that prefix will be sent directly (without going through a router).

The default values for preferred lifetime and valid lifetime are 604,800 seconds and 2,592,000 seconds respectively. The preferred lifetime value must not exceed the valid lifetime value. The preferred lifetime is the length of time that the address within the prefix remains in the preferred state, which means that unrestricted use is allowed by upper-layer protocols. The valid lifetime is the length of time that the prefix is valid relative to the time the packet was sent.

The following example displays the configuration of Ethernet interface 0/1 that is running currently on the device. In this example, IPv6 router advertisement is configured on the interface:

```
device# show running-config interface ethernet 0/1

!
interface ethernet 0/1
  ipv6 address 2001:db8:1111::10/64
  ipv6 neighbor 2001:db8:1111::1 00:04:96:eb:c4:51
  ipv6-router-advertisement
    hop-limit 133
    lifetime 200
    managed-config-flag
    mode disable-unsolicited-ra
    other-config
    enable
    interval 100 min 50
    prefix 2001:db8:100::1/63
    prefix 2001:db8:100::1/64
      advertisement
      autoconfiguration
      onlink
      preferred-lifetime 5000
      valid-lifetime 10000
  !
  no shutdown
!
device#
```

The following example displays details about IPv6 router advertisement on Ethernet interface 0/1:

```
device# show ipv6 router-advertisement interface ethernet 0/1

If Name: ethernet 0/1.0
ICMPv6 active timers:
  Last Router-Advertisement sent :0s
  Next Router-Advertisement sent in : 2s
Router-Advertisement parameters:
  Ra Enable flag : true
  Router lifetime field : 1800
```

```
  Periodic interval : 20 to 10 seconds
  Managed Address Configuration flag : true
  Other config flag : true
  RA Mode : disable_unsolicited_ra
  Current Hop Limit field : 0
  MTU option value : 1500
  Reachable Time field : 1200
device#
```

Duplicate Address Detection

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 (DAD) 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.

In the DAD NS message, the source address field in the IPv6 header is set to the unspecified address (::). The address being queried for duplication cannot be used until it is determined that there are no duplicates. In the neighbor advertisement (NA) reply to a DAD NS message, the destination address in the IPv6 header is set to the link local all-nodes multicast address (FF02::1). The Solicited flag in the NA message is set to 0. Because the sender of the DAD NS message is not using the desired IP address, it cannot receive unicast NA messages. Therefore, the NA message is multicast.

Upon receipt of the multicast NA message with the target address field set to the IP address for which duplication is being detected, the node disables the use of the duplicate IP address on the interface. If the node does not receive an NA message that defends the use of the address, it initializes the address on the interface.

Configure a Static Neighbor Entry for an Interface

A static entry in the IPv6 Neighbor Discovery (ND) cache ensures that an IPv6 neighbor is always reachable. You can create a static ND entry for a device that is not yet connected to the network or to prevent an entry from aging out. You can configure static ND entries for Ethernet, port channel (LAG), and Virtual Ethernet (VE) interfaces as well as Ethernet and port channel subinterfaces.

1. Access global configuration mode.

   ```
   device# configuration terminal
   ```

2. Specify the interface to contain the static ND entry.

   ```
   device(config)# interface ethernet 0/1
   ```

3. Create the static ND entry for the neighbor.

```
device(config-if-eth-0/1)# ipv6 neighbor 2001:db8:2678::2 0000.002b.8641
```

   This example adds a static ND entry for a neighbor with IPv6 address 2001:db8:2678::2 and MAC address aa:aa:aa:aa:aa:aa, reachable through physical port ethernet 0/1.

The following example displays the configuration of Ethernet interface 0/1 that is running currently on the device. In this example, the ND IPv6 address and MAC address on the interface are set to 2001:db8:2678::2 and aa:aa:aa:aa:aa:aa respectively:

```
device# show running-config interface ethernet 0/1

interface ethernet 0/1
  ipv6 address 2001:DB8:1::1
  ipv6 neighbor 2001:db8:2678::2 aa:aa:aa:aa:aa:aa
  no shutdown
!
device#
```

The following examples show how to configure static ND entries for Ethernet, port channel (LAG), and Virtual Ethernet (VE) interfaces as well as Ethernet and port channel subinterfaces respectively:

```
device# configure terminal
device(config)# interface ethernet 0/1
device(config-if-eth-0/1)# ipv6 neighbor 2001:db8:2678::2 aa:aa:aa:aa:aa:aa
device(config-if-eth-0/1)#

device# configure terminal
device(config)# interface port-channel 10
device(config-if-po-10)# ipv6 neighbor 2001:db8:2678::3 aa:aa:aa:aa:aa:ab
device(config-if-po-10)#

device# configure terminal
device(config)# interface ve 100
device(config-if-ve-100)# ipv6 neighbor 2001:db8:2678::4 aa:aa:aa:aa:aa:ac
device(config-if-ve-100)#

device# configure terminal
device(config)# interface ethernet 0/1
device(config-if-eth-0/1)# subinterface vlan 100
device(config-subif-eth-0/1.100)# ipv6 neighbor 2001:db8:2678::5 aa:aa:aa:aa:aa:ad
device(config-subif-eth-0/1.100)#

device# configure terminal
device(config)# interface port-channel 10
device(config-if-po-10)# subinterface vlan 200
device(config-subif-po-10.200)# ipv6 neighbor 2001:db8:2678::6 aa:aa:aa:aa:aa:ae
device(config-subif-po-10.200)#
```

Configure Reachable Time for Remote IPv6 Nodes

You can configure the duration (in seconds) that a router considers a remote IPv6 node to be reachable. You can configure neighbor discovery (ND) reachable time for Ethernet, port channel (LAG), and Virtual Ethernet (VE) interfaces as well as Ethernet and port channel subinterfaces.

To do so, you use the **ipv6 neighbor reachable-time** command to configure the ND IPv6 neighbor reachable-time configuration on an interface. ND establishes

correspondences between pairs of network-layer (Layer 3) addresses and LAN hardware addresses (Layer 2 MAC addresses).

Router advertisement messages include a *reachable time* value. All nodes on a link use the same value.

Reachable time is how long a device considers another device to be reachable after successfully contacting it without re-verifying its presence. When a device sends traffic to a neighbor and receives a response, it marks that neighbor as reachable.

The reachable time is a timer that specifies how long the entry stays in the reachable state before the device must probe or refresh the neighbor information. If no additional communication happens during the reachable time, the device transitions the neighbor entry to a stale state and eventually needs to verify it again using an ND request.

You should configure a long duration, because a short duration causes IPv6 network devices to process the information at a greater frequency.

1. Access global configuration mode.
   ```
   device# configuration terminal
   ```
2. Access interface configuration mode.
   ```
   device(config)# interface ethernet 0/1
   ```
3. Configure the reachable time.

   Note
   The actual reachable time ranges from 0.5 to 1.5 times the configured or default value.

   ```
   device(config-int-eth-0/1)# ipv6 neighbor reachable-time 300
   ```
   The range is 30 to 3600 seconds. The default is 1200 seconds.

The following examples show how to configure IPv6 neighbor reachable time on Ethernet, port channel (LAG), and Virtual Ethernet (VE) interfaces as well as on Ethernet and port channel subinterfaces respectively:

```
device# configure terminal
device(config)# interface ethernet 0/1
device(config-if-eth-0/1)# ipv6 neighbor reachable-time 300
device(config-if-eth-0/1)#

device# configure terminal
device(config)# interface port-channel 10
device(config-if-po-10)# ipv6 neighbor reachable-time 600
device(config-if-po-10)#

device# configure terminal
device(config)# interface ve 100
device(config-if-ve-100)# ipv6 neighbor reachable-time 1500
device(config-if-ve-100)#

device# configure terminal
device(config)# interface ethernet 0/1
device(config-if-eth-0/1)# subinterface vlan 100
device(config-subif-eth-0/1.100)# ipv6 neighbor reachable-time 2500
device(config-subif-eth-0/1.100)#

device# configure terminal
device(config)# interface port-channel 10
```

```
device(config-if-po-10)# subinterface vlan 200
device(config-subif-po-10.200)# ipv6 neighbor reachable-time 3600
device(config-subif-po-10.200)#
```

The following example displays the configuration of Ethernet interface 0/1 that is running currently on the device. In this example, the IPv6 neighbor reachable time is set to 300 seconds on the interface:

```
device# show running-config interface ethernet 0/1

interface ethernet 0/1
  mtu 1600
  ipv6 neighbor reachable-time 300
  no shutdown
!
device#
```

IPv6 ICMP Rate Limiting on the Management Interface

IPv6 Internet Control Message Protocol (ICMP) rate limiting lets you control and monitor IPv6 ICMP echo-request (ping) traffic on the management interface. This protects the interface from ping flooding attacks and excessive IPv6 ICMP traffic that could impact device performance.

This feature ensures device manageability by allowing necessary diagnostics while dropping excessive traffic. It specifically throttles IPv6 ICMP traffic to allow safe troubleshooting without affecting management traffic such as SSH or HTTPS.

ICMP rate limiting on the management interface in IPv6 functions the same as it does in IPv4. For details about configuring and monitoring IPv6 ICMP rate limiting on the management interface, see the *Extreme OS ONE Switching and Routing Security Configuration Guide*.

Display Global IPv6 Information

You can use **show** commands to display information about IPv6 interfaces, neighbors, and route tables.

1. Display IPv6 interface information.

```
device# show ipv6 interface brief

L3 Interface     IPv6-Address        Vrf             Admin status Oper status Address
status
================ ================== ============== ============ ===========
==============
Eth 0/9:3.1332   2001:db8:535::2     default-vrf     UP           UP          PREFERRED
Eth 0/9:3.2399   2001:db8:960::2     default-vrf     UP           UP          PREFERRED
Eth 0/9:3.3054   2001:db8:bef::2     default-vrf     UP           UP          PREFERRED
Eth 0/9:3.3254   2001:db8:cb7::2     default-vrf     UP           UP          PREFERRED
Eth 0/9:3.523    2001:db8:20c::2     default-vrf     UP           UP          PREFERRED
Eth 0/9:3.2044   2001:db8:7fd::2     default-vrf     UP           UP          PREFERRED
Eth 0/9:3.2563   2001:db8:a04::2     default-vrf     UP           UP          PREFERRED
Eth 0/9:3.520    2001:db8:209::2     default-vrf     UP           UP          PREFERRED
Eth 0/9:3.2109   2001:db8:83e::2     default-vrf     UP           UP          PREFERRED
Eth 0/9:3.3328   2001:db8:d01::2     default-vrf     UP           UP          PREFERRED
Eth 0/9:3.3798   2001:db8:ed7::2     default-vrf     UP           UP          PREFERRED
Eth 0/9:3.2148   2001:db8:865::2     default-vrf     UP           UP          PREFERRED
```

```
Eth 0/9:3.312    2001:db8:139::2     default-vrf    UP           UP           PREFERRED
Eth 0/9:3.695    2001:db8:2b8::2     default-vrf    UP           UP           PREFERRED
Eth 0/9:3.720    2001:db8:2d1::2     default-vrf    UP           UP           PREFERRED
Eth 0/9:3.760    2001:db8:2f9::2     default-vrf    UP           UP           PREFERRED
Eth 0/9:3.1166   2001:db8:48f::2     default-vrf    UP           UP           PREFERRED
Eth 0/9:3.1641   2001:db8:66a::2     default-vrf    UP           UP           PREFERRED
Eth 0/9:3.2086   2001:db8:827::2     default-vrf    UP           UP           PREFERRED
device#
```

2. Display IPv6 neighbor information for an interface.

```
device# show ipv6 neighbor interface ve 822

-----------

Interface Name:  _bridge_domain 822

IP Address                  MAC Address         Type      Interface  L2
Interface        Age
========================== ================== ========= ==========
=================== =========

2001:db8:337::1           00:04:96:eb:c4:51   Local     ve 822     -
2h17m3s

2001:db8:337::10          00:10:94:00:07:1d   Dynamic   ve 822     port-channel 1.822
15m19s

2001:db8:337::40          00:10:94:00:46:b3   MLAG      ve 822     port-channel 1.822
2h7m1s

2001:db8:337::60          00:10:94:00:56:b1   MLAG      ve 822     port-channel 1.822
2h6m45s

2001:db8:337::90          00:10:94:00:66:af   MLAG      ve 822     port-channel 1.822
1h22m9s

fe80::204:96ff:feeb:c451  00:04:96:eb:c4:51   Local     ve 822     -
2h17m3s
device#
```

3. Display the IPv6 neighbor discovery protocol (NDP) entries in the neighbor caches.

```
device# show ipv6 neighbor vrf default-vrf

vrf :default-vrf
---------------
Total number of neighbor entries: 4

Ipv6 address        Mac-address         Type       Interface        L2 Interface
Age
================    =================   =======    ==============   ==============
=====
2001:db8:1111::1    aa:aa:aa:aa:aa:aa   Dynamic    ethernet 0/1:1   ethernet 0/1:1
17m35s
2001:db8:1112::2    bb:bb:bb:bb:bb:bb   Static     ethernet 0/1:1   ethernet 0/1:1
17m37s
2001:db8:1113::3    cc:cc:cc:cc:cc:cc   Dynamic    ethernet 0/1:3   ethernet 0/1:3
17m40s
2001:db8:1114::4    dd:dd:dd:dd:dd:dd   Static     ve 100           ethernet 0/1:4
17m42s
device#
```

4. Display the IPv6 route table.

```
device# show ipv6 route vrf default-vrf
```

```
 Resilient Hash: Disabled
Ecmp Max Path: 128
Total number of IPv6 routes: 26, Max Routes: Not Set
'[x/y]' denotes [preference/metric]
2001:db8:1001::/64, attached, [0/0], tag 0, 1m12s
  via direct, ethernet 0/1:1
2001:db8:1001::1/128, local, [0/0], tag 0, 1m12s
  via direct, ethernet 0/1:1
2001:db8:2001::/64, attached, [0/0], tag 0, 1m4s
  via direct, ethernet 0/1:2.200
2001:db8:2001::1/128, local, [0/0], tag 0, 1m4s
  via direct, ethernet 0/1:2.200
2001:db8:3001::/64, attached, [0/0], tag 0, 17s
  via direct, ve 100
2001:db8:3001::1/128, local, [0/0], tag 0, 17s
device#
```

5. Display the aggregated numbers of IPv6 ICMP packets that were filtered because of ICMP rate limiting on the management interface.

```
device# show counters icmp interface management 0

ICMP Packet Summary:
  ICMPv4 Accepted Packets: 8
  ICMPV4 Dropped Packets:  11
  ICMPv6 Accepted Packets: 0
  ICMPV6 Dropped Packets:  0
device#
```

   In this release, the only supported management interface name is 0. For details about configuring and monitoring IPv6 ICMP rate limiting on the management interface, see the *Extreme OS ONE Switching and Routing Security Configuration Guide*.

Clear Global IPv6 Information

You can use **clear** commands to remove IPv6 neighbor discovery information and IPv6 routes.

1. Clear the IPv6 neighbor discovery cache on an interface.

```
device# clear ipv6 neighbor interface ethernet 0/1
```

2. Clear the IPv6 neighbor discovery cache on a VRF.

```
device# clear ipv6 neighbor vrf default-vrf
```

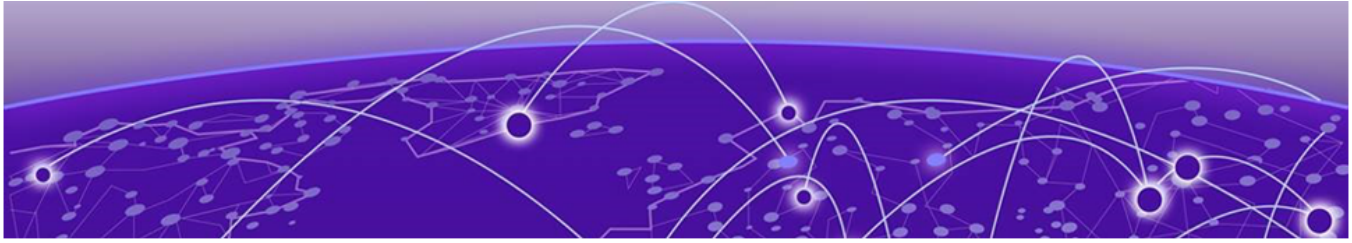
3. Clear the IPv6 routing tables on a VRF.

```
device# clear ipv6 route vrf default-vrf
```

4. Clear statistics for the management interface. This includes the aggregated numbers of IPv6 ICMP packets that the system accepted or dropped because of ICMP rate limiting.

```
device# clear counters interface management 0
device#
```

   In this release, the only supported management interface name is 0. For details about configuring and monitoring IPv6 ICMP rate limiting on the management interface, see the *Extreme OS ONE Switching and Routing Security Configuration Guide*.



IPv4 Static Routing

The following topics describe how to configure IPv4 static routing.

IPv4 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 .

The IPv4 route table can receive routes from several sources, such as static routes, directly connected networks, and BGP4.

Static routes are manually configured entries in the IPv4 routing table. Next hop functionality supports IP addresses and interfaces such as Ethernet, Port Channel (PO), or Virtual Ethernet (VE) interfaces.

You can influence the preference that a route is given by configuring it as follows:

- Configuring a route metric higher than the default metric.
- Giving the route an administrative distance.
- Specifying a route tag for use with a route map.

Static routes can be configured to serve as any of the following route types:

- Default routes
- Primary routes
- Backup routes
- Null routes (for dropping traffic intentionally when the desired connection fails)
- Alternate routes to the same destination (to help load balance traffic)

IPv4 Static Route Availability

IPv4 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.

If the interface is not available and the next hop IP address is invalid, the software removes the static route from the route table. When the port or VE interface becomes available and the next hop is valid, the software adds the route to the route table.

You use this feature to configure the router to adjust to changes in network topology. The router does not continue trying to use routes on unavailable paths. Instead, it uses routes only when their paths are available.

Figure 2 shows an example of a static route that is configured on Switch A:

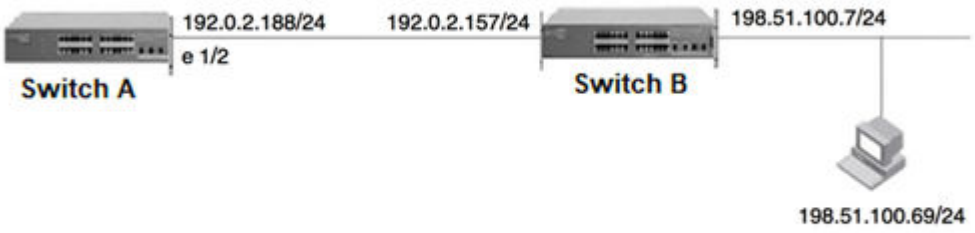


Figure 2: Example of an IPv4 static route

In Figure 2, the static route to the 192.0.2.188 destination is configured as follows. 192.0.2.157 is used as the next hop gateway:

```
device(config-vrf-default-vrf)# static-route 192.0.2.188/24 192.0.2.157
device(config-vrf-default-vrf)#
```

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 192.0.2.157 is reachable through port 1/2 and also assumes that local interfaces in that subnet are on the same port. Switch A deduces that IP interface 198.51.100.69 is also on port 1/2.

Default VRF and User-Defined VRFs

You can use two types of VRF: the default VRF and user-defined VRFs.

Default VRF: The default VRF (always named default-vrf in the system) is used for general routing and is the primary VRF for most network traffic. It exists automatically on the device without any specific configuration. It acts as the device's global routing table. Interfaces are not assigned to the default VRF and must be explicitly moved to a user-defined VRF. Routes and interfaces in the default VRF are separate from those in user-defined VRFs, which prevents route leakage unless explicitly configured otherwise.

User-Defined VRFs: You create user-defined VRFs to isolate specific network traffic or services. Each VRF has an independent routing table and forwarding rules and segregates and manage traffic for different departments, customers, or services in the same physical network infrastructure. They offer granular control over routing policies, interfaces, and security. They also offer flexibility and isolation for specialized network configurations when you use them for the following purposes:

- Creating virtual networks to separate tenants or virtual networks on the same physical infrastructure.
- Isolating services or applications (for example, web servers and database servers) onto their VRFs.
- Providing a sandboxed environment for testing and development.

The VRF handles general network operations, while user-defined VRFs provide customized isolation and routing for specific needs.

Following is an example of the default VRF after configuration:

```
device# show running-config vrf default-vrf

  member loopback 1-2
  static-route 203.0.113.0/24 192.0.2.1 enable-bfd profile default
  static-route 203.0.113.0/24 192.0.2.1 enable-bfd profile default
  static-route 203.0.113.0/24 192.0.2.1 enable-bfd profile default
  static-route 203.0.113.0/24 192.0.2.1 enable-bfd profile default
  static-route 198.51.100.0/24 192.0.2.1 enable-bfd profile default
  static-route 198.51.100.0/24 192.0.2.1 enable-bfd profile default
  static-route 198.51.100.0/24 192.0.2.1 enable-bfd profile default
  static-route 192.0.2.0/24 192.0.2.1 enable-bfd profile default
device#
```

Following is an example of a user-defined VRF named uservrf1 after configuration:

```
device# show running-config vrf uservrf1

  member ethernet 0/1:1
  member ethernet 0/1:2
  member ethernet 0/1:3
  member ethernet 0/1:4
  member port-channel 1 vlan 1-60
  member port-channel 2 vlan 61-120
  member port-channel 3 vlan 121-180
  member port-channel 4 vlan 181-240
  static-route 203.0.113.0/24 192.0.2.1 enable-bfd profile default
  static-route 203.0.113.0/24 192.0.2.1 enable-bfd profile default
  static-route 203.0.113.0/24 192.0.2.1 enable-bfd profile default
```

```
  static-route 203.0.113.0/24 192.0.2.1 enable-bfd profile default
  static-route 203.0.113.0/24 198.51.100.1 enable-bfd profile default
  static-route 203.0.113.0/24 198.51.100.1 enable-bfd profile default
  static-route 198.51.100.0/24 192.0.2.1 enable-bfd profile default
  static-route 198.51.100.0/24 192.0.2.1 enable-bfd profile default
  static-route 192.0.2.0/24 192.0.2.1 enable-bfd profile default
  static-route 203.0.113.0/24 192.0.2.1 enable-bfd profile default
  static-route 2001:db8:2003:1::/64 2001:db8:1004:1::1 enable-bfd profile default
  static-route 2001:db8:2003:13:1::/64 2001:db8:1004:13:1::1 enable-bfd profile default
  static-route 2001:db8:2000::/64 2001:db8:1001:23:1::1 enable-bfd profile default
  static-route 2001:db8:2002::/64 2001:db8:1003:21:1::1 enable-bfd profile default
  static-route 2001:db8:2001::/64 2001:db8:1002:26:1::1 enable-bfd profile default
  static-route 2001:db8:2002::/64 2001:db8:1003:2d:1::1 enable-bfd profile default
  static-route 2001:db8:2003:1f:1::/64 2001:db8:1004:1f:1::1 enable-bfd profile default
  static-route 2001:db8:2002::/64 2001:db8:1003:12:1::1 enable-bfd profile default
  static-route 2001:db8:2002::/64 2001:db8:1003:39:1::1 enable-bfd profile default
  static-route 2001:db8:2003:33:1::/64 2001:db8:1004:33:1::1 enable-bfd profile default
  static-route 2001:db8:2003:38:1::/64 2001:db8:1004:38:1::1 enable-bfd profile default
  static-route 2001:db8:2003:32:0/64 2001:db8:1004:33:1::1 enable-bfd profile default
  static-route 2001:db8:2000:a:1::/64 2001:db8:1001:a:1::1 enable-bfd profile default
  static-route 2001:db8:2000:b:1::/64 2001:db8:1001:b:1::1 enable-bfd profile default
  static-route 2001:db8:2003:a:1::/64 2001:db8:1004:a:1::1 enable-bfd profile default
  static-route 198.51.100.0/24 192.0.2.1 enable-bfd profile default
  static-route 203.0.113.0/24 192.0.2.1 enable-bfd profile default
  static-route 198.51.100.0/24 192.0.2.1 enable-bfd profile default
  static-route 198.51.100.0/24 192.0.2.1 enable-bfd profile default
  static-route 203.0.113.0/24 192.0.2.1 enable-bfd profile default
  static-route 198.51.100.0/24 192.0.2.1 enable-bfd profile default
device#
```

BFD for Layer 3 Protocols

Layer 3 protocols can use Bidirectional Forwarding Detection (BFD) for rapid failure detection in the forwarding path between two adjacent routers, including the interfaces, data links, and forwarding planes.

BFD can be configured for use with the following protocols:

- BGP4
- BGP4+
- Static Route

BFD must be enabled at the interface and routing protocol levels. BFD asynchronous mode depends on the sending of BFD control packets between two systems to activate and maintain BFD neighbor sessions between routers. Therefore, BFD must be configured on both BFD peers.

A BFD session is created after BFD is enabled on the interfaces and at the router level for the appropriate routing protocols. BFD timers are then negotiated, and the BFD peers begin to send BFD control packets to each other at the negotiated interval.

BFD provides one point of forwarding path monitoring when more than one Layer 3 application wants to monitor a host. BFD runs a session for that host and provides the status to multiple applications, instead of multiple applications running individual sessions to the host.

By sending rapid failure detection notices to the routing protocols in the local device to initiate the routing table recalculation process, BFD contributes to greatly reducing overall network convergence time.

Configure a Basic IPv4 Static Route

To configure a basic IPv4 static route, you specify the IPv4 destination address, the address mask, and the IPv4 address of the next hop.

1. Access global configuration mode.

```
device# configure terminal
```

2. Specify a VRF name and enter VRF configuration mode.

```
device(config)# vrf default-vrf
```

3. Enter the IP address and prefix length for the route destination network and the IP address for the next hop.

Note
You must use a prefix length as part of the address. You cannot use network masks. The prefix length of /8 equals a network mask of 255.0.0.0. The prefix length of /24 (equivalent to the mask 255.255.255.0) matches all hosts in the Class C subnet address in the destination IP address.

```
device(config-vrf-default-vrf)# static-route 192.0.2.0/24 192.0.2.1
```

Configure an IPv4 Static Route with an Interface Next Hop

To configure an IPv4 static route that uses an interface as a next hop, you specify the IPv4 destination address, the address mask, and the interface of the next hop.

1. Access global configuration mode.

```
device# configure terminal
```

2. Specify a VRF name and enter VRF configuration mode.

```
device(config)# vrf default-vrf
```

3. Enter the IP address and prefix length for the route destination network and the interface for the next hop.

Note
You must use a prefix length as part of the address. You cannot use network masks. The prefix length of /8 equals a network mask of 255.0.0.0. The prefix length of /24 (equivalent to the mask 255.255.255.0) matches all hosts in the Class C subnet address in the destination IP address.

```
device(config-vrf-default-vrf)# static-route 192.0.2.0/24 interface ethernet 0/1
device(config-vrf-default-vrf)# static-route 192.0.2.1/24 interface port-channel 25
device(config-vrf-default-vrf)# static-route 192.0.2.2/24 interface ve 1
```

Configure a BFD session for an IPv4 static route

An IPv4 static route is associated with a static Bidirectional Forwarding Detection (BFD) session when the next hop for the static route matches the neighbor address of the static BFD neighbor, and BFD monitoring is enabled for the static route.

When static BFD is configured, the static route manager checks the routing table for a route to the BFD neighbor. If a route exists and the next hop is connected directly, a single-hop session is created. If the next hop is not connected directly, a multihop BFD session is created.

When the BFD session is up, a corresponding static route is added to the routing table. When the BFD session that monitors the static route goes down because the BFD neighbor is not reachable, static routes are removed from the routing table. The device replaces these removed routes in the routing table when the BFD neighbor is reachable.

To use BFD for an IPv4 static route, you configure the static route and the corresponding static BFD separately.

1. Access global configuration mode.

```
device# configure terminal
```

2. Access BFD configuration mode.

```
device(config)# bfd
```

3. Create a BFD profile.

   A BFD profile contains the BFD settings that are applied (or requested to be applied) to a routing protocol that is configured (registered) as a BGP client (meaning that it uses BFD for link failure detection instead of its own mechanism).

   Note
   If no BFD profile is configured for a static route, the profile configured for the BFD member interface is used. If BFD profiles are configured for both the static route and the member interface, the one with the shorter detection time takes precedence.

```
device(config-bfd)# profile profile1

device(config-bfd)# member
  ethernet      Ethernet
  loopback      Interface Loopback Port
  port-channel  Port-channel
  ve            Ve
device(config-bfd)# member ethernet 0/1
  profile   Add default profile for sessions under this interface
  shutdown  Administratively shutdown the BFD session
device(config-bfd)#
```

4. Configure the interval (the desired rate at which to send BFD control packets to the neighboring system).

   This includes the desired minimum transmit interval, minimum receive interval for BFD control packets at the local end point, and a multiplier value that helps to calculate the detection timeout of BFD sessions.

```
device(config-bfd-profile-profile1)# interval 5000 min-rx 10000 multiplier 4
```

5. Return to global configuration mode.

```
device(config-bfd-profile-profile1)# exit
device(config-bfd)# exit
```

6. Specify a VRF name and enter VRF configuration mode.

```
device(config)# vrf default-vrf
```

7. Enter the IP address, prefix length for the route destination network, the IP address for the next hop, and the BFD profile.

Note
Prefix lengths must be used as part of the address. Network masks cannot be used. The prefix length of /8 is equivalent to a network mask of 255.0.0.0. The prefix length of /24 (equivalent to the mask 255.255.255.0) matches all hosts in the Class C subnet address in the destination IP address.

```
device(config-vrf-default-vrf)# static-route 192.0.2.0/24 192.0.2.1 enable-bfd profile
profile1
```

The following is a configuration example for the **bfd-source-interface** option, which is used for multihop BFD:

```
device(config-vrf-default-vrf)# static-route 198.51.100.0/24 192.0.2.1 enable-bfd bfd-
source-interface loopback 1 profile default
device(config-vrf-default-vrf)#
```

Disable Recursive Lookup for an IPv4 Static Route

A device uses recursive lookup to search for another route if the original next hop is not reachable. This feature is enabled by default.

If the next hop for a basic static route is reachable directly, it is added to the routing table. If that next hop is not reachable, the device can perform a lookup for another route (a recursive route). You can disable the recursive lookup feature in the default VRF or in a non default VRF.

Note
An original next hop is not considered resolved if it is reachable through a default route such as 0.0.0.0/0.

1. Access global configuration mode.

```
device# configure terminal
```

2. Specify a VRF name and enter VRF configuration mode.

```
device(config)# vrf red
```

3. To disable recursive lookup, enter the following command.

```
device(config-vrf-red)# static-route 192.0.2.0/24 192.0.2.1 no-recurse
```

This example disables recursive lookup in a VRF named red.

Add a Cost Metric or Administrative Distance to an IPv4 Static Route

You can influence route preference by adding a cost metric or an administrative distance to a static route.

The device replaces a static route if it receives a route to the same destination with a lower administrative distance.

1. Access global configuration mode.

```
device# configure terminal
```

2. Specify a VRF name and enter VRF configuration mode.

```
device(config)# vrf default-vrf
```

3. Designate the route destination, the next hop, and the route priority.

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 will share traffic to the destination. The value can be 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. By default, static routes 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 be from 1 to 254. The default is 1. A value of 255 is considered unreachable.

The following example configures a static route with an administrative distance of 10:

```
device(config-vrf-default-vrf)# static-route 198.51.100.0/24 198.51.100.1 admin-
distance 10
device(config-vrf-default-vrf)#
```

The following example configures a static route with an administrative distance of 3:

```
device(config-vrf-default-vrf)# static-route 192.0.2.0/24 192.0.2.1 admin-distance 3
device(config-vrf-default-vrf)#
```

The following example configures a static route with a cost metric of 2:

```
device(config-vrf-default-vrf)# static-route 192.0.2.0/24 192.0.2.1 metric 2
device(config-vrf-default-vrf)#
```

Configure an IPv4 Static Route to Use with a Route Map

You can configure a static route with a tag that can be referenced in a route map.

1. Access global configuration mode.

```
device# configure terminal
```

2. Specify a VRF name and enter VRF configuration mode.

```
device(config)# vrf default-vrf
```

3. Enter the destination network IP address and prefix length, the set-tag keyword, a tag number, and the next hop IP address.

```
device(config-vrf-default-vrf)# static-route 198.51.100.0/24 set-tag 9999 192.0.2.1
```

The tag "9999" in this example can be used in a route map.

Configure an IPv4 Null Static 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 a static route of any type exists with the same metric you specify for the null or interface based route.

Figure 3 depicts how a null static route works with a standard route to the same destination.

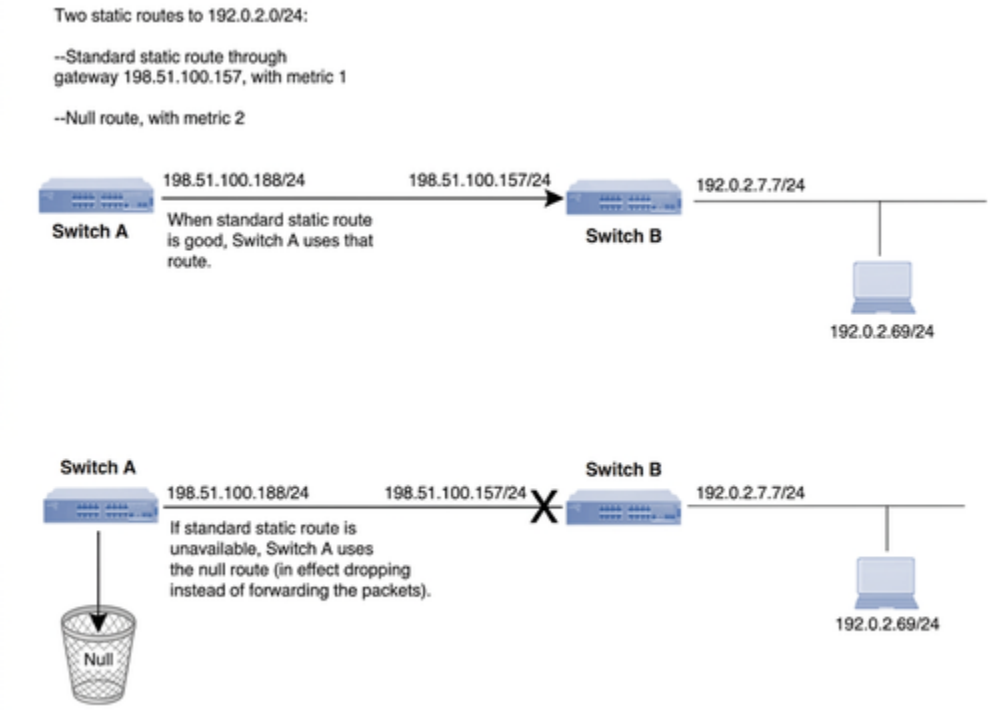


Figure 3: Null route and standard route to same destination

To configure a null static route, complete the following steps.

1. Access global configuration mode.

   ```
   device# configure terminal
   ```

2. Specify a VRF name and enter VRF configuration mode.

   ```
   device(config)# vrf default-vrf
   ```

3. Configure the preferred route to a destination.

   ```
   device(config-vrf-default-vrf)# static-route 192.0.2.0/24 192.0.2.1
   ```

   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.

4. Configure the null route to the same destination, followed by the keyword null, a space, and a zero. Also specify a metric value of 2 (which is higher than the default metric of 1 as described previously).

   ```
   device(config-vrf-default-vrf)# static-route 192.0.2.0/24 null 0 metric 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 summarizes the commands in this procedure:

```
device# configure terminal
device(config)# vrf default-vrf
device(config-vrf-default-vrf)# static-route 192.0.2.0/24 192.0.2.1
device(config-vrf-default-vrf)# static-route 192.0.2.0/24 null 0 metric 2
device(config-vrf-default-vrf)#
```

Configure a Default IPv4 Static Route

A router uses a default static route when there are no other default routes to a destination.

You cannot create a default route to a Virtual Ethernet (VE) or physical interface.

1. Access global configuration mode.

   ```
   device# configure terminal
   ```

2. Specify a VRF name and enter VRF configuration mode.

   ```
   device(config)# vrf default-vrf
   ```

3. Enter the destination route and prefix length (0.0.0.0/0) followed by a valid next hop IP address.

   ```
   device(config-vrf-default-vrf)# static-route 0.0.0.0/0 192.0.2.1
   ```

The following example configures a default route that is a null route:

```
device# configure terminal
device(config)# vrf default-vrf
device(config-vrf-default-vrf)# static-route 0.0.0.0/0 null 0
device(config-vrf-default-vrf)#
```

Configure IPv4 Static Routes for Load Sharing and Redundancy

You can configure multiple static routes to the same destination as load sharing or backup routes.

If you configure multiple static routes to the same destination with different next hop gateways but the same metrics, the device load balances among the routes by 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 device uses the route with the lowest metric. If this route becomes unavailable, the device fails over to the static route with the next lowest metric.

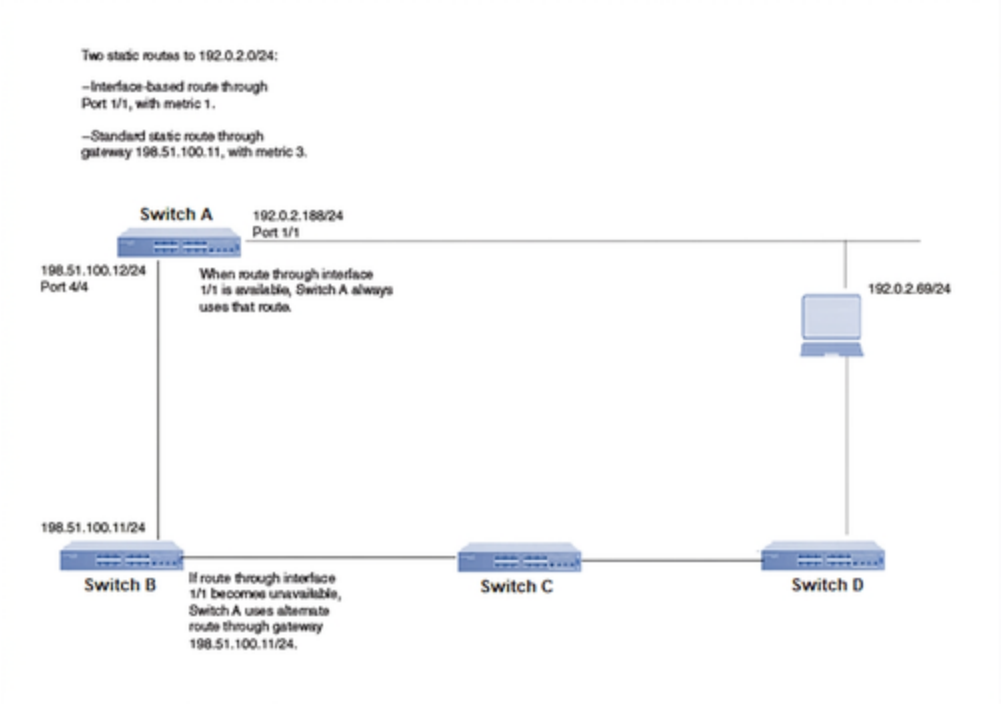


Figure 4: Two static routes to the same destination

Note
You can also use administrative distance to set route priority. Assign the static route a lower administrative distance than other types of routes, unless you want the other route types to be preferred over the static route.

1. Access global configuration mode.

```
device# configure terminal
```

2. Specify a VRF name and enter VRF configuration mode.

```
device(config)# vrf default-vrf
```

3. Enter multiple routes to the same destination using different next hops.

```
device(config-vrf-default-vrf)# static-route 198.51.100.0/24 192.0.2.1
device(config-vrf-default-vrf)# static-route 198.51.100.0/24 192.0.2.2
device(config-vrf-default-vrf)# static-route 198.51.100.0/24 192.0.2.3
```

This example creates three next hop gateways to the destination. Traffic alternates among the three paths through next-hop 192.0.2.1, next hop 192.0.2.2, and next hop 192.0.2.3.

4. To prioritize the routes, use different metrics for each possible next hop.

```
device(config-vrf-default-vrf)# static-route 198.51.100.0/24 192.0.2.1
device(config-vrf-default-vrf)# static-route 198.51.100.0/24 192.0.2.2 metric 2
device(config-vrf-default-vrf)# static-route 198.51.100.0/24 192.0.2.3 metric 3
```

This example creates three alternate routes to the destination. The primary next hop is 192.0.2.1, 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 192.0.2.2, which has the next lowest metric of 2. If the second path fails, traffic is directed to 192.0.2.3, which has a metric of 3.

Remove an IPv4 Static Route

Use the no form of the **static-route** command to remove an IPv4 static route.

1. (Optional) View configured routes and confirm parameters.

```
device# show ipv4 route vrf default-vrf

Resilient Hash: Enabled
Ecmp Max Path: 128
Total number of IPv4 routes: 13, Max Routes: Not Set
'[x/y]' denotes [preference/metric]
192.0.2.0/24, attached, [0/0], tag 0, 5m58s
  via direct, ethernet 0/11:1
192.0.2.1/32, , [0/0], tag 0, 5m58s
  via direct, ethernet 0/11:1
192.0.2.2/32, ARP, [3/0], tag 0, 5m58s
  via , ethernet 0/11:1, [00:16:3e:58:fb:20, ethernet 0/11:1.0]
198.51.100.0/24, attached, [0/0], tag 0, 5m58s
  via direct, ethernet 0/11:2
198.51.100.1/32, , [0/0], tag 0, 5m58s
  via direct, ethernet 0/11:2
198.51.100.2/32, static, [1/1], tag 0, 2m34s
  via 192.0.2.254, ethernet 0/13, [e4:db:ae:5c:24:16, ethernet 0/13]
203.0.113.0/24, attached, [0/0], tag 0, 5m58s
  via direct, ethernet 0/11:3
203.0.113.1/32, , [0/0], tag 0, 5m58s
  via direct, ethernet 0/11:3
203.0.113.3/32, static, [1/1], tag 0, 2m34s
  via 192.0.2.254, ethernet 0/13, [e4:db:ae:5c:24:16, ethernet 0/13]
203.0.113.10/32, ebgp, [20/0], tag 0, 5m32s
  via 192.0.2.2, ethernet 0/11:1, [00:16:3e:58:fb:20, ethernet 0/11:1]
  via 198.51.100.2, ethernet 0/11:2, [00:16:3e:58:fb:21, ethernet 0/11:2]
  via 203.0.113.2, ethernet 0/11:3, [00:16:3e:58:fb:22, ethernet 0/11:3]
198.51.100.20/24, attached, [0/0], tag 0, 5m57s
  via direct, ethernet 0/32:1
198.51.100.20/32, , [0/0], tag 0, 5m57s
  via direct, ethernet 0/32:1
203.0.113.30/24, , [1/1], tag 0, 5m58s
  via 192.0.2.2, ethernet 0/11:1, [00:16:3e:58:fb:20, ethernet 0/11:1]
  via 198.51.100.2, ethernet 0/11:2, [00:16:3e:58:fb:21, ethernet 0/11:2]
  via 203.0.113.2, ethernet 0/11:3, [00:16:3e:58:fb:22, ethernet 0/11:3]
device#
```

2. (Optional) Narrow the output to static routes only.

```
device# show ipv4 route vrf default-vrf static

Resilient Hash: Enabled
Ecmp Max Path: 128
Total number of IPv4 routes: 8, Max Routes: Not Set
'[x/y]' denotes [preference/metric]
198.51.100.2/32, static, [1/1], tag 0, 2m34s
  via 192.0.2.254, ethernet 0/13, [e4:db:ae:5c:24:16, ethernet 0/13]
203.0.113.3/32, static, [1/1], tag 0, 2m34s
  via 192.0.2.254, ethernet 0/13, [e4:db:ae:5c:24:16, ethernet 0/13]
device#
```

3. Access global configuration mode.

```
device# configure terminal
```

4. Specify a VRF name and enter VRF configuration mode.

```
device(config)# vrf default-vrf
```

5. Remove the static route, including the destination and next hop.

```
device(config-vrf-default-vrf)# no static-route 198.51.100.1/32 192.0.2.1
```

You do not need to include cost metric, distance, or tag parameters.

Display IPv4 Static Route Information

You can use **show** commands to display information about configured IPv4 routes, static routes, directly connected routes, routes configured for different protocols, the cost associated with each route, and the time that the route has been available.

1. Display the configured static routes.

```
device# show running-config vrf

vrf default-vrf
  static-route 192.0.2.0/24 192.0.2.2
  static-route 2001:db8:94::/64 set-tag 9999 interface ethernet 0/1 description this
is a connected   static route
  static-route 2001:db8:101::/64 set-tag 9999 2001:db8::55 admin-distance 5 metric 10
no-recurse description all parameters
  static-route 2001:db8:187::/64 2001:db8::55
  static-route 198.51.100.0/24 set-tag 9999 192.0.2.5 admin-distance 5 metric 10 no-
recurse description all parameters
  static-route 2001:db8:1::/64 set-tag 100 null 0 description this is a drop route
  static-route 192.0.2.0/24 set-tag 9999 interface ethernet 0/1 description this is a
connected   static route
  static-route 192.0.2.0/24 set-tag 100 null 0 description this is drop route
  static-route 2001:db8:d7::/64 set-tag 9999 2001:db8::29 interface port-channel 25
admin-distance 5   metric 10 description link-local nexthop
device#
```

2. Display a list of active static routes and their connection times.

```
device# show ipv4 route vrf default-vrf static

Resilient Hash: Enabled
Ecmp Max Path: 128
Total number of IPv4 routes: 8, Max Routes: Not Set
'[x/y]' denotes [preference/metric]
198.51.100.2/32, static, [1/1], tag 0, 2m34s
  via 192.0.2.1, ethernet 0/13, [e4:db:ae:5c:24:16, ethernet 0/13]
203.0.113.3/32, static, [1/1], tag 0, 2m34s
  via 192.0.2.1, ethernet 0/13, [e4:db:ae:5c:24:16, ethernet 0/13]
device#
```

3. Display all active IP routes and their connection times.

```
device# show ipv4 route vrf default-vrf

Resilient Hash: Enabled
Ecmp Max Path: 128
Total number of IPv4 routes: 13, Max Routes: Not Set
'[x/y]' denotes [preference/metric]
192.0.2.0/24, attached, [0/0], tag 0, 5m58s
  via direct, ethernet 0/11:1
192.0.2.1/32, , [0/0], tag 0, 5m58s
  via direct, ethernet 0/11:1
192.0.2.2/32, ARP, [3/0], tag 0, 5m58s
  via , ethernet 0/11:1, [00:16:3e:58:fb:20, ethernet 0/11:1.0]
198.51.100.0/24, attached, [0/0], tag 0, 5m58s
  via direct, ethernet 0/11:2
198.51.100.1/32, , [0/0], tag 0, 5m58s
  via direct, ethernet 0/11:2
```

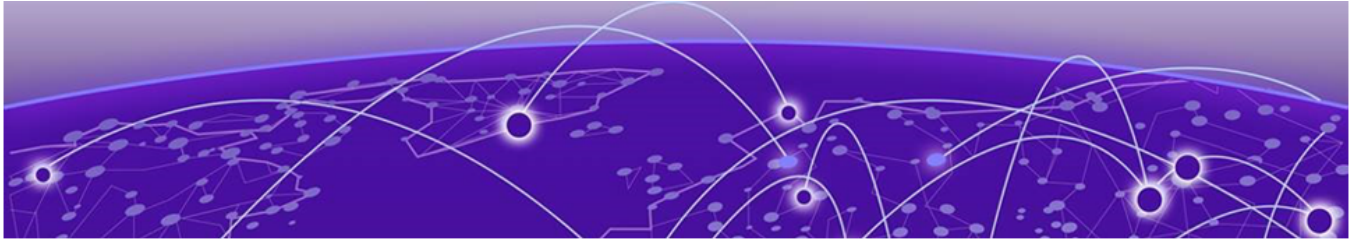
```
198.51.100.2/32, ARP, [3/0], tag 0, 5m57s
  via , ethernet 0/11:2, [00:16:3e:58:fb:21, ethernet 0/11:2.0]
203.0.113.0/24, attached, [0/0], tag 0, 5m58s
  via direct, ethernet 0/11:3
203.0.113.1/32, , [0/0], tag 0, 5m58s
  via direct, ethernet 0/11:3
203.0.113.2/32, ARP, [3/0], tag 0, 5m58s
  via , ethernet 0/11:3, [00:16:3e:58:fb:22, ethernet 0/11:3.0]
192.0.2.10/32, ebgp, [20/0], tag 0, 5m32s
  via 192.0.2.2, ethernet 0/11:1
  via 198.51.100.2, ethernet 0/11:2
  via 203.0.113.2, ethernet 0/11:3
198.51.100.20/24, attached, [0/0], tag 0, 5m57s
  via direct, ethernet 0/32:1
198.51.100.20/32, , [0/0], tag 0, 5m57s
  via direct, ethernet 0/32:1
203.0.113.30/24, , [1/1], tag 0, 5m58s
  via 192.0.2.2, ethernet 0/11:1
  via 198.51.100.2, ethernet 0/11:2
  via 203.0.113.2, ethernet 0/11:3
device#
```

4. Display abbreviated information in the IPv4 routing table for all entries for the VRF instance.

```
device# show ipv4 route vrf default-vrf brief

Total number of IPv4 routes: 5, Max Routes: Not Set
Type Codes - B:BGP L:Local D:Direct/Connected S:Static A: Arp
BGP  Codes - i:iBGP e:eBGP E:evpn
Hardware Status Codes - #:Failed

IP Prefix            Next Hop        Interface              Pref/Metric  Type
-------------------  --------------  ---------------------  -----------  ------
198.51.100.0/24      DIRECT          tunnel testtunnel      0 0/0        D
198.51.100.0/24      DIRECT          tunnel testtunnel      1 0/0        D
192.0.2.0/24         DIRECT          ethernet 0/1          0/0          D
192.0.2.1/32         DIRECT          ethernet 0/1          0/0          L
192.0.2.0/24         198.51.100.1     tunnel testtunnel     0 20/0       Be
device#
```



IPv6 Static Routing

The following topics describe how to configure IPv6 static routing.

IPv6 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 IPv6 routing table. Next hop functionality supports only IP addresses, not interfaces such as Ethernet, Port Channel (PO), or Virtual Ethernet (VE) interfaces.

You can influence the preference that a route is given by configuring a route metric higher than the default metric or giving the route an administrative distance.

You can configure static routes to serve as any of the following types of route:

- Default routes
- Primary routes
- Backup routes
- Null routes for intentionally dropping traffic when the desired connection fails
- Alternate routes to the same destination (to help load balance traffic)

IPv6 Static Route Availability

IPv6 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.

If the interface is not available and the next hop IP address is invalid, the software removes the static route from the route table. When the port or VE interface becomes available and the next hop is valid, the software adds the route to the route table.

You configure this feature so that the device adjusts to changes in network topology. The device does not continue trying to use routes on unavailable paths. Instead, it uses routes only when their paths are available.

Figure 5 shows an example of a static route that is configured on Switch A.

Figure 5: Example of an IPv6 static route

In Figure 5, the static route to 2001:db8:1::/64 was configured as follows, using 2001:db8:1::2 as the next hop gateway.

```
device(config-vrf-default-vrf)# static-route 2001:db8:1::/64 2001:db8:1::2
device(config-vrf-default-vrf)#
```

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:db8:1::2 is reachable through port 1/2 and also assumes that local interfaces in that subnet are on the same port. Switch A deduces that IP interface 2001:db8:1::/64 is also on port 1/2.

Default VRF and User-Defined VRFs

You can use two types of VRF: the default VRF and user-defined VRFs.

Default VRF: The default VRF (always named default-vrf in the system) is used for general routing and is the primary VRF for most network traffic. It exists automatically on the device without any specific configuration. It acts as the device's global routing table. Interfaces are not assigned to the default VRF and must be explicitly moved to a user-defined VRF. Routes and interfaces in the default VRF are separate from those in user-defined VRFs, which prevents route leakage unless explicitly configured otherwise.

User-Defined VRFs: You create user-defined VRFs to isolate specific network traffic or services. Each VRF has an independent routing table and forwarding rules and segregates and manage traffic for different departments, customers, or services in the same physical network infrastructure. They offer granular control over routing policies, interfaces, and security. They also offer flexibility and isolation for specialized network configurations when you use them for the following purposes:

- Creating virtual networks to separate tenants or virtual networks on the same physical infrastructure.
- Isolating services or applications (for example, web servers and database servers) onto their VRFs.
- Providing a sandboxed environment for testing and development.

The VRF handles general network operations, while user-defined VRFs provide customized isolation and routing for specific needs.

Following is an example of the default VRF after configuration:

```
device# show running-config vrf default-vrf

  member loopback 1-2
  static-route 203.0.113.0/24 192.0.2.1 enable-bfd profile default
  static-route 203.0.113.0/24 192.0.2.1 enable-bfd profile default
  static-route 203.0.113.0/24 192.0.2.1 enable-bfd profile default
  static-route 203.0.113.0/24 192.0.2.1 enable-bfd profile default
  static-route 198.51.100.0/24 192.0.2.1 enable-bfd profile default
  static-route 198.51.100.0/24 192.0.2.1 enable-bfd profile default
  static-route 198.51.100.0/24 192.0.2.1 enable-bfd profile default
  static-route 192.0.2.0/24 192.0.2.1 enable-bfd profile default
device#
```

Following is an example of a user-defined VRF named uservrf1 after configuration:

```
device# show running-config vrf uservrf1

  member ethernet 0/1:1
  member ethernet 0/1:2
  member ethernet 0/1:3
  member ethernet 0/1:4
  member port-channel 1 vlan 1-60
  member port-channel 2 vlan 61-120
  member port-channel 3 vlan 121-180
  member port-channel 4 vlan 181-240
  static-route 203.0.113.0/24 192.0.2.1 enable-bfd profile default
  static-route 203.0.113.0/24 192.0.2.1 enable-bfd profile default
  static-route 203.0.113.0/24 192.0.2.1 enable-bfd profile default
  static-route 203.0.113.0/24 192.0.2.1 enable-bfd profile default
  static-route 203.0.113.0/24 198.51.100.1 enable-bfd profile default
  static-route 203.0.113.0/24 198.51.100.1 enable-bfd profile default
  static-route 198.51.100.0/24 192.0.2.1 enable-bfd profile default
  static-route 198.51.100.0/24 192.0.2.1 enable-bfd profile default
  static-route 192.0.2.0/24 192.0.2.1 enable-bfd profile default
  static-route 203.0.113.0/24 192.0.2.1 enable-bfd profile default
  static-route 2001:db8:2003:1::/64 2001:db8:1004:1::1 enable-bfd profile default
  static-route 2001:db8:2003:13:1::/64 2001:db8:1004:13:1::1 enable-bfd profile default
  static-route 2001:db8:2000::/64 2001:db8:1001:23:1::1 enable-bfd profile default
  static-route 2001:db8:2002::/64 2001:db8:1003:21:1::1 enable-bfd profile default
  static-route 2001:db8:2001::/64 2001:db8:1002:26:1::1 enable-bfd profile default
  static-route 2001:db8:2002::/64 2001:db8:1003:2d:1::1 enable-bfd profile default
  static-route 2001:db8:2003:1f:1::/64 2001:db8:1004:1f:1::1 enable-bfd profile default
```

```
  static-route 2001:db8:2002::/64 2001:db8:1003:12:1::1 enable-bfd profile default
  static-route 2001:db8:2002::/64 2001:db8:1003:39:1::1 enable-bfd profile default
  static-route 2001:db8:2003:33:1::/64 2001:db8:1004:33:1::1 enable-bfd profile default
  static-route 2001:db8:2003:38:1::/64 2001:db8:1004:38:1::1 enable-bfd profile default
  static-route 2001:db8:2003:32:0/64 2001:db8:1004:33:1::1 enable-bfd profile default
  static-route 2001:db8:2000:a:1::/64 2001:db8:1001:a:1::1 enable-bfd profile default
  static-route 2001:db8:2000:b:1::/64 2001:db8:1001:b:1::1 enable-bfd profile default
  static-route 2001:db8:2003:a:1::/64 2001:db8:1004:a:1::1 enable-bfd profile default
  static-route 198.51.100.0/24 192.0.2.1 enable-bfd profile default
  static-route 203.0.113.0/24 192.0.2.1 enable-bfd profile default
  static-route 198.51.100.0/24 192.0.2.1 enable-bfd profile default
  static-route 198.51.100.0/24 192.0.2.1 enable-bfd profile default
  static-route 203.0.113.0/24 192.0.2.1 enable-bfd profile default
  static-route 198.51.100.0/24 192.0.2.1 enable-bfd profile default
device#
```

Configure a Basic IPv6 Static Route

Specify the IPv6 destination address, the address mask, and the IPv6 address of the next hop.

Before you configure a basic IPv6 static route, you must enable IPv6 on at least one interface by configuring an IPv6 address or enabling IPv6 on that interface explicitly.

1. Access global configuration mode.

   ```
   device# configure terminal
   ```

2. Specify a VRF name and enter VRF configuration mode.

   ```
   device(config)# vrf default-vrf
   ```

3. Enter the route destination IPv6 address in hexadecimal with 16-bit values between colons, the address prefix length preceded by a slash, and the IPv6 address of the next hop gateway.

   Note
   The IPv6 address architecture is defined in RFC 2373.

   ```
   device(config-vrf-default-vrf)# static-route 2001:db8::/32 2001:db8:0:ee44::1
   ```

Configure an IPv6 Static Route with an Interface Next Hop

To configure an IPv6 static route that uses an interface as a next hop, you specify the IPv6 destination address, the address mask, and the interface of the next hop.

Before you configure an IPv6 static route that uses an interface as a next hop, you must enable IPv6 on at least one interface by configuring an IPv6 address or explicitly enabling IPv6 on that interface.

1. Access global configuration mode.

   ```
   device# configure terminal
   ```

2. Specify a VRF name and enter VRF configuration mode.

   ```
   device(config)# vrf default-vrf
   ```

3. Enter the route destination IPv6 address in hexadecimal with 16-bit values between colons, the address prefix length preceded by a slash, and the interface of the next hop.

   Note
   The IPv6 address architecture is defined in RFC 2373.

```
device(config-vrf-default-vrf)# static-route 2001:db8:94::/64 interface ethernet 0/1
device(config-vrf-default-vrf)# static-route 2001:db8:95::/64 interface port-channel 25
device(config-vrf-default-vrf)# static-route 2001:db8:96::/64 interface ve 1
```

Configure a BFD session for an IPv6 static route

An IPv6 static route is associated with a static Bidirectional Forwarding Detection (BFD) session when the next hop for the static route matches the neighbor address of the static BFD neighbor, and BFD monitoring is enabled for the static route.

When static BFD is configured, the static route manager checks the routing table for a route to the BFD neighbor. If a route exists and the next hop is directly connected, a single-hop session is created. If the next hop is not directly connected, a multihop BFD session is created.

When the BFD session is up, a corresponding static route is added to the routing table. When the BFD session that monitors the static route goes down because the BFD neighbor is not reachable, static routes are removed from the routing table. These removed routes are replaced in the routing table when the BFD neighbor is reachable.

To use BFD for an IPv6 static route, you configure the static route and the corresponding static BFD separately.

1. Access global configuration mode.

```
device# configure terminal
```

2. Access BFD configuration mode.

```
device(config)# bfd
```

3. Create a BFD profile.

   A BFD profile contains the BFD settings that are applied (or requested to be applied) to a routing protocol that is configured (registered) as a BGP client (meaning that it uses BFD for link failure detection instead of its own mechanism).

```
device(config-bfd)# profile profile1
```

4. Configure the interval (the desired rate at which to send BFD control packets to the neighboring system).

   This includes the desired minimum transmit interval, minimum receive interval for BFD control packets at the local end point, and a multiplier value that helps to calculate the detection timeout of BFD sessions.

```
device(config-bfd-profile-profile1)# interval 5000 min-rx 10000 multiplier 4
```

5. Return to global configuration mode.

```
device(config-bfd-profile-profile1)# exit
device(config-bfd)# exit
```

6. Specify a VRF name and enter VRF configuration mode.

```
device(config)# vrf default-vrf
```

7. Enter the IP address, prefix length for the route destination network, the IP address for the next hop, and the BFD profile.

Note
Prefix lengths must be used as part of the address. For example, /64 is the prefix length in the 10:1::/64 address used in this step.

```
device(config-vrf-default-vrf)# static-route 2001:db8:1::/64 set-tag 100 null 0
description This is a drop route
```

Disable Recursive Lookup for an IPv6 Static Route

A device use the recursive lookup feature to search for another route if the original next-hop is not reachable. This feature is enabled by default.

If the next hop for a basic static route is directly reachable, it is added to the routing table. If that next hop is not reachable, the device can perform a lookup for another route (a recursive route). You can enable the recursive lookup feature in the default VRF or in a non-default VRF.

Note
An original next-hop is not considered resolved if it is reachable through a default route, such as ::/0.

1. Access global configuration mode.

```
device# configure terminal
```

2. Specify a VRF name and enter VRF configuration mode.

```
device(config)# vrf red
```

3. To disable recursive lookup, enter the following command.

```
device(config-vrf-red)# static-route 2001:db8:101::/64 2001:db8:55::55 no-recurse
```

Add a Cost Metric or Administrative Distance to an IPv6 Static Route

You can influence route preference by adding a cost metric or an administrative distance to a static route.

Before you add a cost metric or an administrative distance to a static route, you must enable IPv6 on at least one interface by configuring an IPv6 address or explicitly enabling IPv6 on that interface.

1. Access global configuration mode.

```
device# configure terminal
```

2. Designate the route destination, the next hop, and the route priority.

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

Option	Description
	value can be 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. By default, static routes 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 admin-distance and can be from 1 to 254. The default is 1. A value of 255 is considered unreachable.

This example configures a static route with an administrative distance of 3:

```
device(config-vrf-default-vrf)# static-route 2001:db8::/64 2001:db8:0:ee44::1 admin-
distance 3
device(config-vrf-default-vrf)#
```

This example configures a static route with an administrative distance of 254:

```
device(config-vrf-default-vrf)# static-route 2001:db8::/64 2001:db8:0:ee44::1 admin-
distance 254
device(config-vrf-default-vrf)#
```

This example configures a static route with a cost metric of 2:

```
device(config-vrf-default-vrf)# static-route 2001:db8::/64 2001:db8:0:ee44::1 metric 2
device(config-vrf-default-vrf)#
```

Configure an IPv6 Static Route to Use with a Route Map

You can configure a static route with a tag that can be referenced in a route map.

Before you configure a static route with a tag that can be referenced in a route map, you must enable IPv6 on at least one interface by configuring an IPv6 address or explicitly enabling IPv6 on that interface.

1. Access global configuration mode.

   ```
   device# configure terminal
   ```

2. Specify a VRF name and enter VRF configuration mode.

   ```
   device(config)# vrf default-vrf
   ```

3. Enter the destination IP address and prefix length, the set-tag keyword, a tag number, and the next hop address.

   ```
   device(config-vrf-default-vrf)# static-route 2001:db8:d7::/64 set-tag 9999 fe80::29
   ```

   The tag "9999" in this example can be used in a route map.

Configure an IPv6 Null Static 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 a static route of any type exists with the same metric you specify for the null or interface based route.

Figure 6 shows how an IPv6 null static route works with a standard route to the same destination.

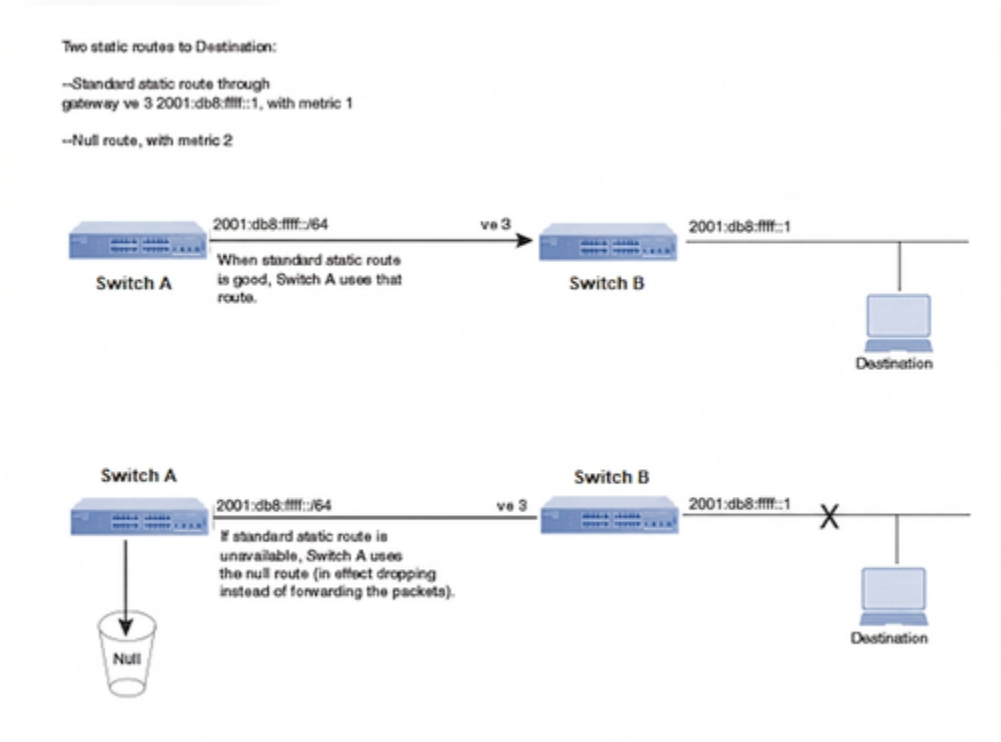


Figure 6: IPv6 null route and standard route to same destination

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. Specify a VRF name and enter VRF configuration mode.

   ```
   device(config)# vrf default-vrf
   ```

3. Configure the preferred route to a destination.

   ```
   device(config-vrf-default-vrf)# static-route 2001:db8::/64 fe80::1 interface ve 3
   ```

   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 Ethernet interface 3 (ve 3). The route uses the default cost metric of 1.

4. Configure the null route to the same destination, followed by the keyword null, a space, and a zero. Also specify a metric value of 2 (which is higher than the default metric of 1 as described previously).

   ```
   device(config-vrf-default-vrf)# static-route 2001:db8::/64 null 0 metric 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 summarizes the commands in this procedure:

```
device# configure terminal
device(config)# vrf default-vrf
device(config-vrf-default-vrf)# static-route 2001:db8::/64 fe80::1 interface ve 3
device(config-vrf-default-vrf)# static-route 2001:db8::/64 null 0 metric 2
device(config-vrf-default-vrf)#
```

Configure a Default IPv6 Static Route

A router uses a default static route when there are no other default routes to a destination.

You cannot create a default route to a Virtual Ethernet (VE) or physical interface.

1. Access global configuration mode.

   ```
   device# configure terminal
   ```

2. Specify a VRF name and enter VRF configuration mode.

   ```
   device(config)# vrf default-vrf
   ```

3. Enter the destination route and network mask (::/0) followed by a valid next hop IP address.

   ```
   device(config-vrf-default-vrf)# static-route ::/0 2001:db8:0:ee44::1
   ```

The following example summarizes the commands in this procedure:

```
device# configure terminal
device(config)# vrf default-vrf
ddevice(config-vrf-default-vrf)# static-route ::/0 2001:db8:0:ee44::1
device(config-vrf-default-vrf)#
```

Configure IPv6 Static Routes for Load Sharing and Redundancy

You can configure multiple static routes to the same destination as 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 device 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 device always uses the route with the lowest metric. If this route becomes unavailable, the device fails over to the static route with the next lowest metric.

Figure 7 shows two static routes to the same destination.

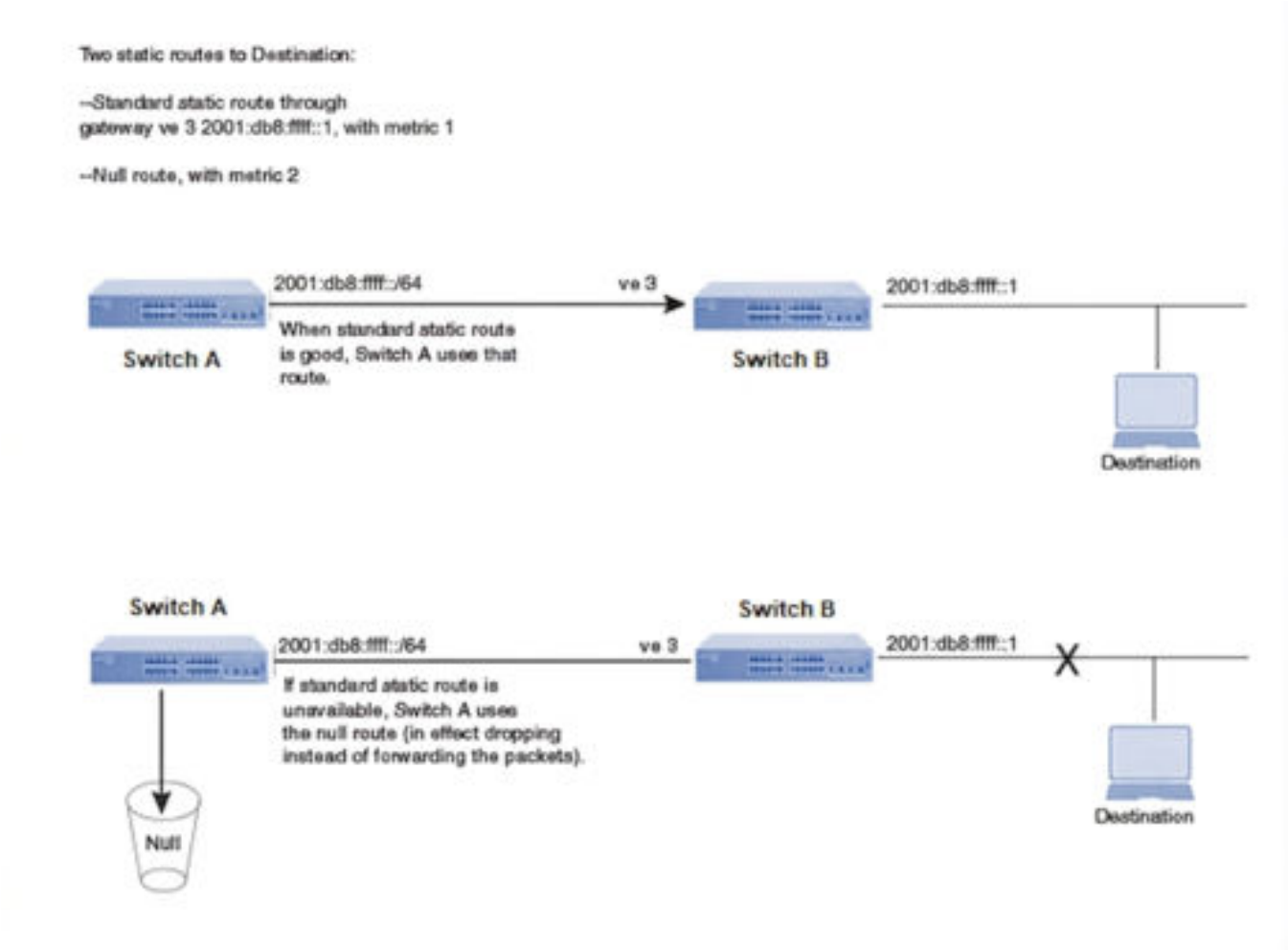


Figure 7: Two static routes to same destination

Note
You can also use administrative distance to set route priority. Assign the static route a lower administrative distance than other types of routes, unless you want the other route types to be preferred over the static route.

1. Access global configuration mode.

```
device# configure terminal
```

2. Specify a VRF name and enter VRF configuration mode.

```
device(config)# vrf default-vrf
```

3. Enter multiple routes to the same destination using different next hops.

```
device(config-vrf-default-vrf)# static-route 2001:db8::/64 2001:db8:2343:0:ee44::1
device(config-vrf-default-vrf)# static-route 2001:db8::/64 2001:db8:2344:0:ee44::2
```

This example creates two next-hop gateways for all 2001:DB8::0/64 destinations. Traffic alternates between the two paths.

4. To prioritize multiple routes, use different metrics for each possible next hop.

```
device(config-vrf-default-vrf))# static-route 2001:db8::/64 2001:db8:2343:0:ee44::1
device(config-vrf-default-vrf))# static-route 2001:db8::/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 (the default metric is not entered in the CLI). If this path is not available, traffic is directed through 2001:db8:2344:0:ee44::2, which has the next lowest metric of 2.

Remove an IPv6 Static Route

Use the no form of the **`static-route`** command to remove an IPv6 static route.

1. (Optional) View configured routes and confirm parameters.

```
device# show ipv6 route vrf default-vrf

Total number of IPv6 routes: 110, Max Routes: Not Set
'[x/y]' denotes [preference/metric]
2001:db8:152:1::2/128, static, [21/1], tag 0, 2m0s
  via 2001:db8:211:11:2::2, ethernet 0/5:2
  via 2001:db8:211:2:1::2, ethernet 0/5:1
2001:db8:152:2::1/128, attached, [0/0], tag 0, 3m11s
  via direct, loopback 522
2001:db8:152:2::2/128, static, [1/1], tag 0, 2m0s
  via 2001:db8:211:2:2::2, ethernet 0/5:2.2
2001:db8:152:3::1/128, attached, [0/0], tag 0, 3m11s
  via direct, loopback 523
2001:db8:211:2:4::2/128, ARP-local, [3/0], tag 0, 2m0s
  via , port-channel 122.3
2001:db8:211:11:2::1/128, local, [0/0], tag 0, 2m0s
  via direct, ethernet 0/5:2
2001:db8:211:11:2::2/128, local, [254/0], tag 0, 2m0s
  via 2001:db8:211:11:2::2, ethernet 0/5:2
```

2. (Optional) Narrow the output to static routes only.

```
device# show ipv6 route vrf default-vrf static

Total number of IPv6 routes: 110, Max Routes: Not Set
'[x/y]' denotes [preference/metric]

2001:db8:152:1::2/128, static, [21/1], tag 0, 2m0s
  via 2001:db8:211:11:2::2, ethernet 0/5:2
  via 2001:db8:211:2:1::2, ethernet 0/5:1
2001:db8:152:2::2/128, static, [1/1], tag 0, 2m0s
  via 2001:db8:211:2:2::2, ethernet 0/5:2.2
```

3. Access global configuration mode.

```
device# configure terminal
```

4. Specify a VRF name and enter VRF configuration mode.

```
device(config)# vrf default-vrf
```

5. Remove the static route, including the destination and next hop.

```
device(config-vrf-default-vrf)# no static-route 2001:db8:152:1::2/128
2001:db8:211:11:2::2
```

You do not need to include cost metric, distance, or tag parameters.

Display IPv6 Static Route Information

You can use show commands to display information about connected, static, and protocol routes.

1. Display the configured static routes.

```
vrf default-vrf
  static-route 192.0.2.0/24 198.51.100.2
  static-route 2001:db8:10:94::/64 set-tag 9999 interface ethernet 0/1 description
this is a connected static route
  static-route 2001:db8:10:101::/64 set-tag 9999 2001:db8:55::55 admin-distance 5
metric 10 no-recurse description all parameters
  static-route 2001:db8:10:200::/64 2001:db8:55::55
  static-route 198.51.100.0/24 set-tag 9999 203.0.113.5 admin-distance 5 metric 10 no-
recurse description all parameters
  static-route 2001:db8:10:1::/64 set-tag 100 null 0 description this is a drop route
  static-route 192.0.2.0/24 set-tag 9999 interface ethernet 0/1 description this is a
connected static route
  static-route 192.0.2.0/24 set-tag 100 null 0 description this is drop route
  static-route 2001:db8:10:d7::/64 set-tag 9999 fe80::29 interface port-channel 25
admin-distance 5 metric 10 description link-local nexthop
```

2. Display a list of active static routes and their connection times.

```
device# show ipv6 route vrf default-vrf static

Resilient Hash: Enabled
Ecmp Max Path: 128
Total number of IPv6 routes: 6, Max Routes: Not Set
'[x/y]' denotes [preference/metric]
2001:db8:2222::2/128, static, [1/1], tag 0, 2m36s
  via 2001:db8:1003::3, ethernet 0/13
2001:db8:3333::3/128, static, [1/1], tag 0, 2m36s
  via 2001:db8:1003::3, ethernet 0/13
device#
```

3. Display all active IP routes and their connection times.

```
device# show ipv6 route vrf default-vrf

Resilient Hash: Disabled
Ecmp Max Path: 128
Total number of IPv6 routes: 26, Max Routes: Not Set
'[x/y]' denotes [preference/metric]
2001:db8:1001::/64, attached, [0/0], tag 0, 1m12s
  via direct, ethernet 0/1:1
2001:db8:1001::1/128, local, [0/0], tag 0, 1m12s
  via direct, ethernet 0/1:1
2001:db8:2001::/64, attached, [0/0], tag 0, 1m4s
  via direct, ethernet 0/1:2.200
2001:db8:2001::1/128, local, [0/0], tag 0, 1m4s
  via direct, ethernet 0/1:2.200
2001:db8:3001::/64, attached, [0/0], tag 0, 17s
  via direct, ve 100
2001:db8:3001::1/128, local, [0/0], tag 0, 17s
device#
```

4. Display abbreviated information in the IPv6 routing table for all entries for the VRF instance.

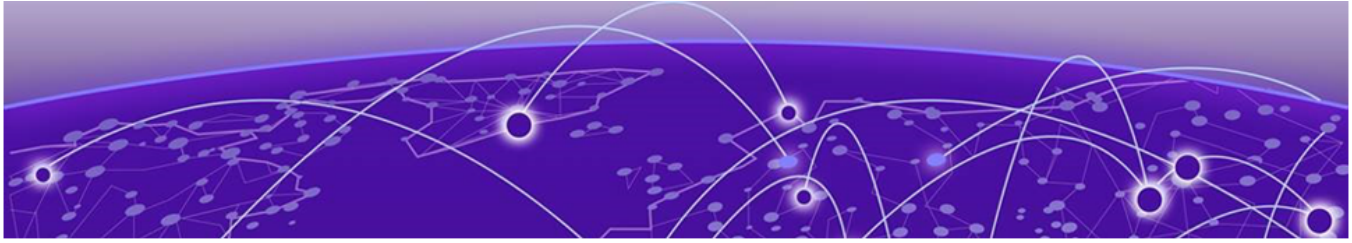
```
device# show ipv6 route vrf default-vrf brief

Total number of IPv6 routes: 40, Max Routes: Not Set

IP Prefix              Next Hop                    Interface             Pref/
```

```
Metric  Type
-------------------------------------------------------------------------------------
-----
2001:db8:1521::/64      DIRECT                       ve 1521
0/0           D
2001:db8:1522::/64      DIRECT                       ve 1522
0/0           D
2001:db8:1523::/64      DIRECT                       ve 1523
0/0           D
2001:db8:1524::/64      DIRECT                       ve 1524
0/0           D
2001:db8:1621::/64      DIRECT                       ve 1621
0/0           D
2001:db8:5:3:1::/64     DIRECT                       ethernet 0/1:3
0/0           D
2001:db8:6:3:1::/64     DIRECT                       ethernet 0/1:4.6
0/0           D
2001:db8:152:1::1/128   DIRECT                       loopback 521
0/0           D
2001:db8:152:1::2/128   2001:db8:211:2:3::           port-channel 121
1/1           S
                        2001:db8:211:2:2::           ethernet 0/5:2.2
                        2001:db8:211:2:1::           ethernet 0/5:1
2001:db8:152:2::1/128   DIRECT                       loopback 522
0/0           D
2001:db8:152:2::2/128   2001:db8:11:2:2::            ethernet 0/5:2.2
21/1          S
2001:db8:152:3::1/128   DIRECT                       loopback 523
0/0           D
2001:db8:152:3::2/128   2001:db8:211:2:3::           port-channel 121
21/1          S
2001:db8:152:4::1/128   DIRECT                       loopback 524
0/0           D
2001:db8:152:4::2/128   2001:db8:5:3:1::2            ethernet 0/1:3
21/1          S
2001:db8:2:1:4::1/128   2001:db8:6:3:1::2            ethernet 0/1:4.6
21/1          S
2001:db8:2:3:2::2/128   2001:db8:211:2:2::           ethernet 0/5:2.2
21/1          S
2001:db8:2:3:3::2/128   2001:db8:211:2:3::           port-channel 121
21/1          S
2001:db8:2:3:4::2/128                               254/0        L
                        2001:db8:211:2:4::           port-channel 122.3
2001:db8:5:3:1::1/128   DIRECT                       ethernet 0/1:3
0/0           L
2001:db8:5:3:1::2/128   MAC: 00:05:03:01:00:02       ethernet 0/1:3
3/0           A
2001:db8:6:3:1::1/128   DIRECT                       ethernet 0/1:4.6
0/0           L
2001:db8:6:3:1::2/128   MAC: 00:06:03:01:00:02       ethernet 0/1:4.6
3/0           A
2001:db8:211:2:3::/127  DIRECT                       port-channel 121
0/0           D
2001:db8:211:2:4::/127  DIRECT                       port-channel 122.3
0/0           D
2001:db8:211:2:1::/128  MAC: f0:64:26:f7:91:32       ethernet 0/5:1
3/0           A
2001:db8:211:2:2::/128  MAC: f0:64:26:f7:91:33       ethernet 0/5:2.2
3/0           A
2001:db8:211:2:3::/128  MAC: f0:64:26:f7:91:04       port-channel 121
3/0           A
2001:db8:211:2:4::/128  MAC: f0:64:26:f7:91:05       port-channel 122.3
3/0           A
2001:db8:100:1:3::1/128 DIRECT                       loopback 2
```

```
0/0          D
2001:db8:100:1:3::2/128 2001:db8:5:3:1::2           ethernet 0/1:3
21/1         S
2001:db8:160:1:3::1/128 DIRECT                      loopback 601
0/0          D
2001:db8:160:1:3::2/128 2001:db8:6:3:1::2           ethernet 0/1:4.6
21/1         S
2001:db8:211:2:1::1/128 DIRECT                      ethernet 0/5:1
0/0          L
2001:db8:211:2:2::1/128 DIRECT                      ethernet 0/5:2.2
0/0          L
2001:db8:211:2:3::1/128 DIRECT                      port-channel 121
0/0          L
2001:db8:211:2:4::1/128 DIRECT                      port-channel 122.3
0/0          L
```



VRRP Gateway Redundancy

VRRP Gateway Redundancy

About VRRP

Virtual Router Redundancy Protocol version 3 (VRRPv3) is a network protocol that enables automatic failover of gateway routers, increasing availability and reliability of routing paths. VRRPv3 is defined in RFC 9568 and provides sub-second failover capability for IPv4 environments.

Prerequisites

To configure and manage VRRP, you must have the following:

- Administrator access to Extreme OS ONE devices.
- Understanding of basic Layer 3 routing concepts (IP addressing, routing tables, default gateways).
- Two or more Extreme OS ONE devices on the same Layer 2 network segment or VLAN.
- At least one IP address assigned to each physical interface where VRRP will run.
- A virtual IP address (VIP) for the VRRP group.

Supported Platforms

VRRPv3 is supported on TD3 and TD4 platforms.

Key Terminology

Table 5: VRRP Terminology

Term	Definition
Active Router	The physical router currently forwarding traffic for a VRRP virtual router. There is exactly one Active router per VRRP group at any time.
Backup Router	A physical router standing by to take over Active responsibility if the current Active router fails. Multiple Backup routers can exist per VRRP group.
Virtual IP (VIP)	A shared IP address that represents the VRRP virtual router. End hosts configure the VIP as their default gateway.
Virtual MAC (VMAC)	A shared MAC address associated with the VIP. The Active router responds to ARP (IPv4) or Neighbor Discovery (IPv6) requests for the VIP using the VMAC.
VRRP Group	A set of physical routers (minimum two) that cooperate to manage a virtual router instance. Each group has a unique Virtual Router ID (VRID).
Virtual Router ID (VRID)	Numeric identifier used to uniquely identify a VRRP group on a network segment.
Preemption	A VRRP behavior in which a higher-priority Backup router takes over the Active role when it becomes available, even if the current Active router is functioning normally.
Failover	The event during which a Backup router transitions to Active, typically due to Active router failure or administrative action.

VRRP Overview and Concepts

The Default Gateway Problem

In traditional single-router network designs, all hosts on a subnet configure a static default gateway — the first-hop router that forwards packets destined for other subnets. This architecture creates a single point of failure:

- If the gateway router fails, all hosts on that subnet lose connectivity to other networks.
- Hosts cannot dynamically discover a new gateway without manual reconfiguration or deployment of dynamic routing protocols.
- Even if an alternate physical path exists in the network infrastructure, hosts continue to send traffic to a MAC address that is no longer reachable.
- Failover can take minutes or longer if manual intervention is required.

This architecture is unacceptable for mission-critical networks where high availability and continuous service delivery are required.

How VRRP Solves Gateway Redundancy

VRRPv3 solves the default gateway problem by presenting a virtual router — represented by a virtual IP address (VIP) and virtual MAC address (VMAC) — that is shared across multiple physical routers. End hosts configure the VIP as their default gateway instead of a single physical router's IP address.

At any given time, one physical router (the Active router) owns and operates the virtual router. It responds to all packets addressed to the VIP and forwards traffic on behalf of hosts using that VIP as their default gateway. Other physical routers (Backup routers) monitor the Active router and automatically take over if it fails.

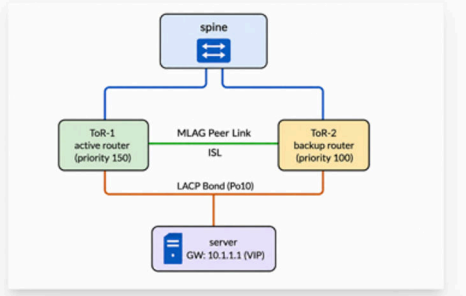


Figure 8: VRRP Architecture: Active and Backup Routers

Key benefits:

- **Sub-second failover:** Backup routers detect Active router failure within 1–3 seconds and assume the Active role automatically.
- **Transparent to end hosts:** Hosts do not need to reconfigure their default gateway or run dynamic routing protocols.
- **Line-rate forwarding:** Traffic forwarding occurs via hardware lookup tables with no performance penalty.
- **Instant MAC learning:** When failover occurs, the new Active router sends gratuitous ARP (IPv4) or unsolicited neighbor discovery (IPv6) messages. Switches learn the VIP-to-VMAC mapping instantly.
- **No client reconfiguration:** Hosts' ARP tables remain valid because the VMAC is owned by whichever router is Active.

VRRP Operation Overview

VRRP operates as follows:

1. Multiple physical routers are configured with the same VRRP group ID and virtual IP address.
2. Each router is assigned a priority value (0 – 255). Routers with the same priority are resolved by IP address.
3. During initialization, routers elect an Active router — the router with the highest priority (or highest IP address if priorities are equal).
4. The Active router periodically sends VRRP advertisement packets to notify Backup routers of its operational status. The advertisement interval is typically 1 second but is configurable.

5. If the Active router fails, Backup routers stop receiving advertisements and wait for the *Active Down Interval* (typically 3 seconds) before assuming the Active router is unavailable.
6. The highest-priority Backup router transitions to the Active state and begins forwarding traffic for the VIP.
7. If the original Active router recovers and preemption is enabled, it may reclaim the Active role based on its priority.

IPv4 Support

VRRP supports IPv4 networks with independent virtual router instances per protocol:

- **IPv4 VRRP:** A VRRP group protects an IPv4 virtual IP (VIP) on an interface. Hosts send ARP requests for the VIP, which the Active router answers using the IPv4 virtual MAC (format: 00-00-5E-00-01-*VRID*).

Common Use Cases

Table 6: VRRP Use Cases

Use Case	Description
Gateway Redundancy	Two routers (active-passive) protect a subnet's default gateway. If the Active router fails, the Backup takes over immediately, and hosts maintain connectivity.

Standards Compliance

VRRP implementation in Extreme OS ONE conforms to RFC 9568 (VRRPv3). The implementation supports the following RFC 9568 features:

- Dynamic election of Active router based on configured priority
- Sub-second failover capability

Configure VRRP

Configure VRRP for an IPv4 Interface

Configure a VRRP group on an IPv4 interface to provide default gateway redundancy.

- Administrator access to the device.
- Two or more devices on the same Layer 2 network segment or VLAN.
- An IPv4 address configured on the interface where VRRP will run.
- An IPv4 virtual IP address for the VRRP group. The virtual IP address must be in the same subnet as the physical IPv4 address.

The following example configures VRRP on Ethernet 0/1. The physical IPv4 address is `1.1.1.1/24`, the VRRP group ID is `1`, and the virtual IPv4 address is `1.1.1.6`.

1. Enter interface configuration mode.

```
device# configure terminal
device(config)# interface ethernet 0/1
```

2. Enable VRRP configuration for the IPv4 address on the interface.

```
device(config-if-eth-0/1)# ipv4 address 1.1.1.1/24 vrrp
```

3. Create the VRRP group.

```
device(config-if-eth-0/1-vrrp)# vrrp-group 1
```

4. Configure the advertisement interval in centiseconds.

```
device(config-if-vrrp-group-1)# advertisement-interval 150
```

5. Configure preemption and the preempt delay.

```
device(config-if-vrrp-group-1)# preempt-mode
device(config-if-vrrp-group-1)# preempt-delay 30
```

6. Configure the VRRP priority.

```
device(config-if-vrrp-group-1)# priority 200
```

7. Configure the virtual IPv4 address.

```
device(config-if-vrrp-group-1)# virtual-ipv4-address 1.1.1.6
```

8. Optional: Configure interface tracking.

```
device(config-if-vrrp-group-1)# interface-tracking
device(config-vrrp-group-interface-tracking)# track-interface ethernet 0/2
device(config-vrrp-group-interface-tracking)# priority-decrement 30
```

The VRRP group is configured for the IPv4 interface. When the device is Active for the group, it forwards traffic for the configured virtual IPv4 address.

Running configuration example

```
device# show running-config interface ethernet 0/1
interface ethernet 0/1
  ipv4 address 1.1.1.1/24 vrrp
    vrrp-group 1
      advertisement-interval 150
      preempt-mode
      preempt-delay 30
      priority 200
      virtual-ipv4-address 1.1.1.6
      interface-tracking
        track-interface ethernet 0/2
        priority-decrement 30
```

Use `show vrrp brief` or `show vrrp interface` to verify VRRP state and configuration.

VRRP gNMI and OpenConfig YANG Reference

Configure and query VRRP using gNMI and the OpenConfig YANG hierarchy documented for VRRP.

OpenConfig YANG Hierarchy

VRRP configuration and state are organized under the following OpenConfig YANG hierarchy.

```
/interfaces/interface[name]/subinterfaces/subinterface[index]/ipv{4,6}/vrrp/vrrp-
group[group-id]/
├── config/
│   ├── group-id
│   ├── virtual-address
│   ├── priority
│   ├── preempt
│   ├── preempt-delay
│   └── advertisement-interval
├── state/
│   └── ... (mirror of config plus status)
```

Configure a VRRP Group

The following example configures VRRP on interface `Ethernet0/1`, subinterface `0`, for IPv4.

```
gnmic -a <device ip>:443 -u admin -p <password> --skip-verify set --
update "interfaces/interface[name=ethernet 0/1]/subinterfaces/subinterface[index=0]/ipv4/
addresses/address[ip=1.1.1.1]/vrrp/vrrp-group[virtual-router-id=3]/config/virtual-router-
id:::uint:::10"
```

Enable Preemption

The following example enables preemption for the VRRP group.

```
gnmic -a <device ip>:443 -u admin -p <password> --skip-verify set --update "interfaces/
interface[name=ethernet 0/1]/subinterfaces/subinterface[index=0]/ipv4/addresses/
address[ip=1.1.1.1]/vrrp/vrrp-group[virtual-router-id=3]/config/preempt:::bool:::true"
```

Query VRRP Operational State

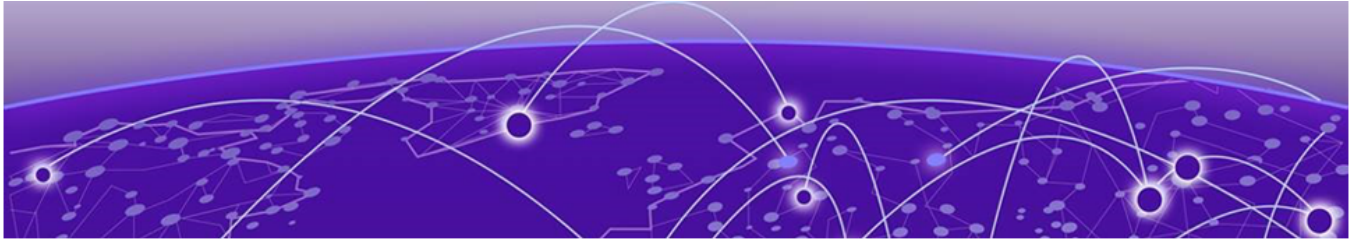
The following example queries the current VRRP state for IPv4 group `10`.

```
gnmi-c get \
  --path /interfaces/interface[name=ethernet 0/1]/subinterfaces/subinterface[index=0]/
ipv4/addresses/address[1.1.1.1]/vrrp/vrrp-group[virtual-router-id=10]/state \
  --target <device>
```

CLI to OpenConfig YANG Mapping

The following table maps the VRRP CLI configuration elements to the corresponding OpenConfig YANG elements documented for VRRP.

CLI command	OpenConfig YANG element
vrrp-group <vr-id>	vrrp-group[virtual-router-id]
virtual-ipv4-address <A.B.C.D>	config/virtual-address
priority <value>	config/priority
preempt-mode	config/preempt
preempt-delay <delay>	config/preempt-delay
advertisement-interval <interval>	config/advertisement-interval



Layer 3 Policy Based Routing

Use this topic to learn about the Layer 3 policy based routing. This topic defines the route filtering object and container for applying routing policies to dynamic routing protocols.

Routing Policy

The Routing Policy feature controls routing information flow. You can configure policies to determine which routes that dynamic routing protocols accept or advertise.

You use the routing policy for the following purposes:

- Filtering routes imported into the routing table
- Controlling route exports
- Manipulating attributes such as the preference value, AS path, and community

How it works

Routing policies offer two control points: Before routing information is added to the BGP routing table and before routing information is advertised to the BGP peers.

Configuration commands are handled by the respective protocol microservice. Routes are evaluated against routing policies using the library's infrastructure.

Configuring Routing Policies

The Routing Policy configuration commands provide detailed information about routing policy configuration. You can find all Routing Policy CLI commands in the *Extreme OS ONE Switching and Routing Command Reference Guide*.

Creating Routing Policies

The following output shows how to configure match conditions, policy results, and policy actions.

```
route-policy
  prefix-set p1
    prefix 192.0.2.0/24 mask-range exact
```

```
    prefix 192.0.2.0/24 mask-range 24..32
  !

  prefix-set p2
    prefix 192.0.2.1/32 mask-range exact
  !

  bgp-defined-sets
    community-set c1
      member 1:1 100:56
      match-set-options any
    !

    community-set c2
      member 10:56 no-advertise no-export NO_EXPORT_SUBCONFED
    !

    ext-community-set ec1
      member 1:1
      match-set-options all
    !

    large-community-set lc1
      member 1:1:1
      match-set-options all
    !

as-path-set as1
      member 10 11 65000 1.1 1[0-9][1-2] 1[1-5][4-7] 65001
    !

    as-path-set test
  !

  policy p1
    statement 1
      conditions
        match-prefix-set p1 any
        bgp-conditions
          local-pref-eq 10
          match-as-path-set as1 any
          match-community-set c1
         match-ext-community-set ec1
         match-large-community-set lc1
        !
      !
      actions
        policy-result permit
        bgp-actions
          set-local-pref 100
          set-med 200
          set-route-origin egp
          set-next-hop 198.51.100.1
          set-community add c1
          set-ext-community add ec1
         set-large-community replace lc1
          set-as-path-prepend 65537 4
        !
      !
    !

    statement 3
      conditions
        match-prefix-set p1 any
```

```
        bgp-conditions
          med-eq 100
        !
      !
      actions
        policy-result permit
        bgp-actions
          set-local-pref 200
          set-med 300
          set-next-hop 198.51.100.2
          set-community add c1
          set-ext-community add ec1
          set-as-path-prepend 21 4
        !
      !
    !
  !

  policy p2
    statement 1
      conditions
        match-prefix-set p2 any
        bgp-conditions
          local-pref-eq 10
        !
      !
      actions
        policy-result permit
        bgp-actions
          set-community add c1
          set-as-path-prepend 20 4
        !
      !
    !
  !
!
```

Attaching Routing Policies

You can attach routing policies at the address-family per peer-group level for BGP clients in either the ingress or egress direction. Route policy is not supported for **l2vpn-evpn** address family.

To do so, you use the following commands.

Import policy at peer group level

```
device# configure terminal
device(config)# vrf default-vrf
device(config-vrf-default-vrf)# router bgp
device(config-vrf-bgp)# peer-group pg1
device(config-vrf-bgp-pg)# address-family ipv4 unicast
device(config-vrf-bgp-pg-ipv4u)# import-policy map1
```

Export policy at peer group level

```
device# configure terminal
device(config)# vrf default-vrf
device(config-vrf-default-vrf)# router bgp
device(config-vrf-bgp)# peer-group pg1
device(config-vrf-bgp-pg)# address-family ipv4 unicast
device(config-vrf-bgp-pg-ipv4u)# export-policy map1
```

The following output shows how to import and export policies at the peer group level:

```
vrf default-vrf
  member ethernet 0/2 vlan 1-8,33-36
  member ethernet 0/3 vlan 100
  member ve 37-44,69-72
  member loopback 1

  router bgp
    local-as 1
    router-id 192.0.2.1

    address-family ipv4 unicast
      activate
    !

    address-family ipv6 unicast
      activate
    !

    peer-group PG_IPV4_1
      remote-as 1
      address-family ipv4 unicast
        export-policy p1
        activate
      !
      neighbor 198.51.100.0
    !

    peer-group PG_IPV4_3
      remote-as 65
      address-family ipv4 unicast
        activate
      !
      neighbor 198.51.100.1
    !
  !
!
```

The following output shows how to display the routing policy configuration that is running currently on the device:

```
route-policy
  prefix-set prefix1
    prefix 192.0.2.0/24 mask-range exact
    prefix 198.51.100.0/24 mask-range exact
    prefix 203.0.113.0/24 mask-range 24..32
  !

  prefix-set prefix2
    prefix 2002:121:1::/64 mask-range exact
    prefix 2002:141:1::/64 mask-range exact
  !

  bgp-defined-sets
    as-path-set as1
      member 10, 10
    !
  !

  policy policy6
    statement 1
      actions
        policy-result permit
```

```
      !
    !
  !

  policy policy1
    statement 1
      conditions
        match-prefix-set prefix1 any
      !
      actions
        policy-result permit
        bgp-actions
          set-med 55
        !
      !
    !
    statement 2
  !

  policy policy2
    statement 1
      conditions
        match-prefix-set prefix1 any
      !
      actions
        policy-result deny
      !
    !
    statement 2
  !

  policy policy6
    statement 1
      conditions
        match-prefix-set prefix2 any
      !
      actions
        policy-result permit
        bgp-actions
          set-med 69
        !
      !
    !
  !
!
```

The following output shows how to display BGP information (including any inbound and outbound policies configured:

```
device# show bgp vrf abc neighbor 2.2.2.1

Peer: 198.51.100.1, Remote Port: 179, Peer-AS: 10
Localhost: , Local Port: 60996, Local-AS: 2, Local-AS-Forced: -
Peer Router ID: 192.0.2.2, Local Router ID: 203.0.113.20, VRF: abc
Route Reflector Client: No, Cluster-ID: -
Fast External Failover: false
State: Idle, Uptime: -, Dynamic Peer: false, Peer Group: peer2

Last Notification Time      : 2023-08-13 19:49:50 +0000 UTC
Last Connection Reset Reason: Update Message Error Malformed AS Path
Send Community: No, Send Extended Community: No

Address-family Inbound Policy Outbound Policy
============== ============== ===============
IPv4 UNICAST                  map10
```

```
Timers          Interval(Sec)      Method             Remaining(Sec)
======          ================== ================== ==============
Connect Retry   31                 Default            -
Hold Timer      0                  Negotiated         -
Keepalive Timer 60                 Default            -
Start Timer     5                  System             -
Update Interval 0                  Default            -
device#
```

Routing Policy Configuration Commands

You use the following commands to configure a routing policy.

- prefix-set: configures the prefix list.

```
device# configure terminal
device(config)# route-policy
device(config-route-policy)# prefix-set p1
device(config-route-policy-prefix-set-p1)# prefix 192.0.2.0/24
device(config-route-policy-prefix-set-p1)# prefix 198.51.100.0/24 mask-range exact
```

- as-path-set: configures the as path set.

```
device# configure terminal
device(config)# route-policy
device(config-route-policy)# bgp-defined-sets
device(config-route-policy-bgp-set)# as-path-set aspath1
device(config-route-policy-bgp-set-as-path-set-aspath1)# member 65535 65400
device(config-route-policy-bgp-set-as-path-set-aspath1)# exit
device(config-route-policy-bgp-set)# as-path-set aspath2
device(config-route-policy-bgp-set-as-path-set-aspath2)# member 45556423 45556420
device(config-route-policy-bgp-set-as-path-set-aspath2)# exit
device(config-route-policy-bgp-set)#
```

- community-set: configures the community set.

```
device# configure terminal
device(config)# route-policy
device(config-route-policy)# bgp-defined-sets
device(config-route-policy-bgp-set)# community-set comm1
device(config-route-policy-bgp-set-community-comm1)# match-set-options all
device(config-route-policy-bgp-set-community-comm1)# member 65535
device(config-route-policy-bgp-set-community-comm1)# exit
device(config-route-policy-bgp-set)#
device(config-route-policy-bgp-set)# community-set comm2
device(config-route-policy-bgp-set-community-comm2)# match-set-options invert
device(config-route-policy-bgp-set-community-comm2)# member no-advertise no-export
device(config-route-policy-bgp-set-community-comm2)# exit
```

- ext-community-set: configures the extended community set.

```
device # configure terminal
device(config)# route-policy
device(config-route-policy)# bgp-defined-sets
device(config-route-policy-bgp-set)# ext-community-set ex-comm1
device(config-route-policy-bgp-set-ext-community-ex-comm1)# match-set-options any
device(config-route-policy-bgp-set-ext-community-ex-comm1)# member route-
target:65535:100 65535:200
device(config-route-policy-bgp-set-ext-community-ex-comm1)# exit
device(config-route-policy-bgp-set)# ex-comm2
device(config-route-policy-bgp-set-ext-community-ex-comm2)# match-set-options all
device(config-route-policy-bgp-set-ext-community-ex-comm2)# member route-
target:65510:100 route-target:65520:200
device(config-route-policy-bgp-set-ext-community-ex-comm2)# exit
```

- large-community-set: configures the large community set.

```
device # configure terminal
device(config)# route-policy
device(config-route-policy)# bgp-defined-sets
device(config-route-policy-bgp-set)# large-community-set large-comm1
device(config-route-policy-bgp-set-large-community-large-comm1)# match-set-options any
device(config-route-policy-bgp-set-large-community-large-comm1)# member 655350:5:100
65000:5:101 64497:.*:.*
device(config-route-policy-bgp-set-large-community-large-comm1)# exit
device(config-route-policy-bgp-set)# large-comm2
device(config-route-policy-bgp-set-large-community-large-comm2)# match-set-options all
device(config-route-policy-bgp-set-large-community-large-comm2)# member 655350:5:102
65000:5:103 64498:.*:.*
device(config-route-policy-bgp-set-large-community-large-comm2)# exit
```

- route-policy: configures route policy definition.

```
device# configure terminal
device(config)# route-policy
device(config-route-policy)# policy map1
device(config-route-policy-policy-map1)# statement 10
device(config-route-policy-policy-map1-stmt-10)# exit
```

- match conditions: indicates the container for all match conditions. All match directives for this statement block will be under conditions.

```
device# configure terminal
device(config)# route-policy
device(config-route-policy)# policy map1
device(config-route-policy-policy-map1)# statement 10
device(config-route-policy-policy-map1-stmt-10)# conditions
device(config-route-policy-policy-map1-stmt-10-conditions)#
```

- prefix-set-name: references a defined prefix set.

```
device# configure terminal
device(config)# route-policy
device(config-route-policy)# policy map1
device(config-route-policy-policy-map1)# statement 10
device(config-route-policy-policy-map1-stmt-10)# conditions
device(config-route-policy-policy-map1-stmt-10-conditions)# match-prefix-set p1 any
```

- BGP specific match statements (conditions): indicates the container for all BGP specific match conditions. All match directives for this statement block will be under bgp-conditions.

```
device# configure terminal
device(config)# route-policy
device(config-route-policy)# policy map1
device(config-route-policy-policy-map1)# statement 10
device(config-route-policy-policy-map1-stmt-10)# conditions
device(config-route-policy-policy-map1-stmt-10-conditions)# bgp-conditions
device(config-route-policy-policy-map1-stmt-10-conditions-bgp-conditions)#
```

- Match based on MED (match med, med-value): multiple exit discriminator (MED) value to be matched against.

```
device# configure terminal
device(config)# route-policy
device(config-route-policy)# policy map1
device(config-route-policy-policy-map1)# statement 10
device(config-route-policy-policy-map1-stmt-10)# conditions
device(config-route-policy-policy-map1-stmt-10-conditions)# bgp-conditions
device(config-route-policy-policy-map1-stmt-10-conditions-bgp-conditions)# med-eq 20
```

- Match based on ORIGIN: indicates match based on route origin and type of origin.

```
device# configure terminal
device(config)# route-policy
```

```
device(config-route-policy)# policy map1
device(config-route-policy-policy-map1)# statement 10
device(config-route-policy-policy-map1-stmt-10)# conditions
device(config-route-policy-policy-map1-stmt-10-conditions)# bgp-conditions
device(config-route-policy-policy-map1-stmt-10-conditions-bgp-conditions)# origin-eq
igp
```

- Match based on NEXTHOP: matches the next hop IP address, and the IP address of the next hop interface.

```
device# configure terminal
device(config)# route-policy
device(config-route-policy)# policy map1
device(config-route-policy-policy-map1)# statement 10
device(config-route-policy-policy-map1-stmt-10)# conditions
device(config-route-policy-policy-map1-stmt-10-conditions)# bgp-conditions
device(config-route-policy-policy-map1-stmt-10-conditions-bgp-conditions)# next-hop-in
192.0.2.1
```

- Match based on AFI or SAFI: delete the configuration, matches AFI or SAFI parameters, represents AFI or SAFI types as defined in oc-bgp-types.

```
device# configure terminal
device(config)# route-policy
device(config-route-policy)# policy map1
device(config-route-policy-policy-map1)# statement 10
device(config-route-policy-policy-map1-stmt-10)# conditions
device(config-route-policy-policy-map1-stmt-10-conditions)# bgp-conditions
device(config-route-policy-policy-map1-stmt-10-conditions-bgp-conditions)# afi-safi-in
IPV4_UNICAST
```

- Match based on local-preference: Used to delete the configuration, matches local preference parameter, local preference value to be matched against.

```
device# configure terminal
device(config)# route-policy
device(config-route-policy)# policy map1
device(config-route-policy-policy-map1)# statement 10
device(config-route-policy-policy-map1-stmt-10)# conditions
device(config-route-policy-policy-map1-stmt-10-conditions)# bgp-conditions
device(config-route-policy-policy-map1-stmt-10-conditions-bgp-conditions)# local-pref-
eq 100
```

- Match based on route-type: Used to delete the configuration, match based on route type, represents type of route internal/external.

```
device# configure terminal
device(config)# route-policy
device(config-route-policy)# policy map1
device(config-route-policy-policy-map1)# statement 10
device(config-route-policy-policy-map1-stmt-10)# conditions
device(config-route-policy-policy-map1-stmt-10-conditions)# bgp-conditions
device(config-route-policy-policy-map1-stmt-10-conditions-bgp-conditions)# route-type
internal
```

- Match based on community-set: defines community-set match condition, name of the community-set that is being referenced.

```
device# configure terminal
device(config)# route-policy
device(config-route-policy)# policy map1
device(config-route-policy-policy-map1)# statement 10
device(config-route-policy-policy-map1-stmt-10)# conditions
device(config-route-policy-policy-map1-stmt-10-conditions)# bgp-conditions
device(config-route-policy-policy-map1-stmt-10-conditions-bgp-conditions)# match-
community-set  comm1
```

- Match based on extended community-set: defines extended community-set match condition, name of the defined community set that is being referenced.

```
device# configure terminal
device(config)# route-policy
device(config-route-policy)# policy map1
device(config-route-policy-policy-map1)# statement 10
device(config-route-policy-policy-map1-stmt-10)# conditions
device(config-route-policy-policy-map1-stmt-10-conditions)# bgp-conditions
device(config-route-policy-policy-map1-stmt-10-conditions-bgp-conditions)# match-ext-
community-set ex-comm1
```

- Match based on large community-set: defines large community-set match condition, name of the defined community set that is being referenced.

```
device# configure terminal
device(config)# route-policy
device(config-route-policy)# policy map1
device(config-route-policy-policy-map1)# statement 10
device(config-route-policy-policy-map1-stmt-10)# conditions
device(config-route-policy-policy-map1-stmt-10-conditions)# bgp-conditions
device(config-route-policy-policy-map1-stmt-10-conditions-bgp-conditions)# match-large-
community-set large-comm1
```

- Match based on as-path-set: match as-path-set condition.

```
device# configure terminal
device(config)# route-policy
device(config-route-policy)# policy map1
device(config-route-policy-policy-map1)# statement 10
device(config-route-policy-policy-map1-stmt-10)# conditions
device(config-route-policy-policy-map1-stmt-10-conditions)# bgp-conditions
device(config-route-policy-policy-map1-stmt-10-conditions-bgp-conditions)# match-as-
path-set aspath1 any
```

- Actions: all actions for this statement block will be under this one.

```
device# configure terminal
device(config)# route-policy
device(config-route-policy)# policy map1
device(config-route-policy-policy-map1)# statement 10
device(config-route-policy-policy-map1-stmt-10)# actions
device(config-route-policy-policy-map1-stmt-10-actions)#
```

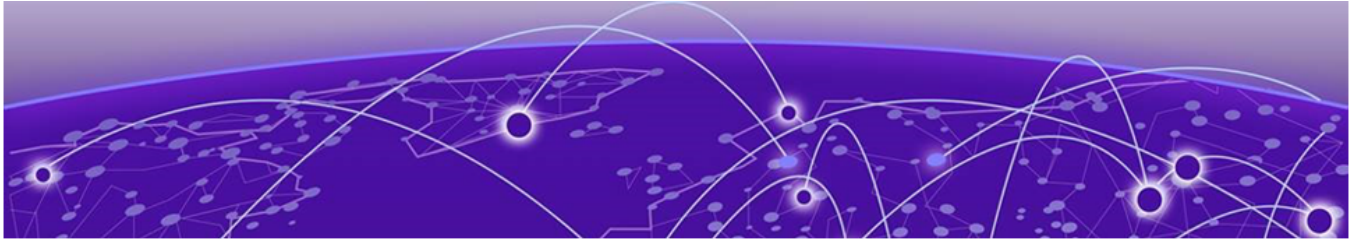
- BGP actions: all BGP actions for this statement block will be under this one.

```
device# configure terminal
device(config)# route-policy
device(config-route-policy)# policy map1
device(config-route-policy-policy-map1)# statement 10
device(config-route-policy-policy-map1-stmt-10)# actions
device(config-route-policy-policy-map1-stmt-10-actions)# bgp-actions
device(config-route-policy-policy-map1-stmt-10-actions-bgp-actions)# set-as-path-
prepend 110 5
device(config-route-policy-policy-map1-stmt-10-actions-bgp-actions)# set-community add
comm1
device(config-route-policy-policy-map1-stmt-10-actions-bgp-actions)# set-ext-community
add extcomm2
device(config-route-policy-policy-map1-stmt-10-actions-bgp-actions)# set-large-
community add large-comm2
device(config-route-policy-policy-map1-stmt-10-actions-bgp-actions)# set-local-pref 50
device(config-route-policy-policy-map1-stmt-10-actions-bgp-actions)# set-med 20
device(config-route-policy-policy-map1-stmt-10-actions-bgp-actions)# set-next-hop
192.0.2.1
device(config-route-policy-policy-map1-stmt-10-actions-bgp-actions)# set-route-origin
```

```
igp
device(config-route-policy-policy-map1-stmt-10-actions-bgp-actions)#
```

- Policy match result: select the final disposition for the route (either permit or deny). If a statement has match or action criteria and policy-result is not configured, then the policy library uses the default value of DENY as a result.

```
device# configure terminal
device(config)# route-policy
device(config-route-policy)# policy map1
device(config-route-policy-policy-map1)# statement 10
device(config-route-policy-policy-map1-stmt-10)# actions
device(config-route-policy-policy-map1-stmt-10-actions)# policy-result permit
device(config-route-policy-policy-map1-stmt-10-actions)#
```



BGP4

The following topics describe how to configure Border Gateway Protocol version 4 (BGP4).

BGP4 Capabilities and Limitations

BGP4 is the standard protocol used on the Internet to route traffic between autonomous systems (ASs) while preventing routing loops. An AS is a group of networks with shared routing and administrative characteristics, such as a company's internal network. These networks can use different internal routing protocols, but to communicate with other ASs, they must use an exterior gateway protocol such as BGP4. This protocol provides the exchange of routing information between devices in different ASs.

Limitations

Extreme OS ONE does not support the following BGP4 features:

- BGP aggregation
- BGP confederation
- BGP route dampening

Supported BGP Features

Extreme OS ONE supports the following BGP features:

- Four Octet AS Number
- Address Family
- Peer Group
- Static Neighbors
- Listen Range
- Authentication
- EBGP Multihop
- Fast External Failover
- Overriding Local-AS
- Timers
- Route Origination Redistribution
- Route Origination Network
- Route Origination Default-Information
- Route Selection and Download
- Route Advertisement
- Policy Enforcement
- Load Balancing or ECMP
- Add Path
- Route Reflection
- BFD
- Multi-Instance Support

- Route Refresh
- Route Aggregation

BGP Communities, Extended Communities, and Route Filtering

BGP Communities

BGP Communities is a variable-length attribute that groups routes with common characteristics. Each community is represented by a 32-bit value, divided into an Autonomous System Number (ASN) and a locally defined value.

Types of Communities

Following are the community types:

- Standard Communities: 4-octet values for grouping routes
- Extended Communities: 8-octet values for larger grouping or categorization
- Large Communities: 12-octet values for larger grouping or categorization

Community Lists and Filtering

Following are the types of community lists and filtering:

- Community Lists: Define a list of communities for filtering or setting path attributes
- Extended Community Lists: Similar to community lists but for extended communities
- Large Community Lists: Similar to community lists but for large communities
- AS Path Lists: Filter routes based on AS numbers or regular expressions
- IP Prefix Lists: Match routes based on prefixes

Route Policy

A route policy defines the following attributes:

- Route Policy: This is a set of match conditions and parameter settings that control routes and change attributes.
- Linking Lists: Route policies reference filter lists, which are then applied to peer groups.
- Direction: You define the direction of route policy application (incoming or outgoing).

Key Points

You should use the following guidelines for community lists:

- The system has no default lists (configuration is required).
- The system links lists through route policies, not directly to peer groups.
- The system carries communities set by policy in BGP path attributes.
- The CLI has no specific command for triggering community sending (varies by vendor).

For details on syntax and parameters for configuring Routing Policy, see the *Extreme OS ONE Switching and Routing Release Notes*.

BGP4 Peering

BGP4 does not have neighbor detection capability. BGP4 neighbors (peers) must be configured manually.

Neighbor address or listen range must be configured.

BGP Static Peers

A BGP peer is a network device that participates in the Border Gateway Protocol (BGP) routing process to exchange routing information with other peers. These peers can be from different Autonomous Systems (ASs) in External BGP (eBGP) or within the same AS in Internal BGP (iBGP).

Key Configuration Points

You should use the following guidelines when configuring BGP static peers:

- Peer Identification: Each BGP peer is identified by its IP address and AS number (for eBGP).
- Peer Grouping: Peers are grouped under a peer group, and group specific configurations are applied. This simplifies configuration and reduces administrative overhead.
- No Default Peers: No default peers exist in the system. Each peer must be manually configured.
- Peer Configuration: Peer specific configurations, such as IP address and AS number, are defined under a peer group. Configurations associated with a peer group can be overridden per peer.
- Address Family: A peer group can only include peers of the same address family (IPv4 or IPv6).
- Link-Local Addresses: Link local addresses can be used as BGP IPv6 peers, but must be associated with a specific interface due to their non-uniqueness.

Best Practices

When configuring BGP static peers, you should create a peer group for multiple peers with shared attributes. You should configure peer group specific attributes rather than individual peer configurations.

BGP Peering with Listen Range

Unlike Interior Gateway Protocol (IGP) neighbors, which are typically discovered automatically and are one hop away, Border Gateway Protocol (BGP) neighbors often require manual configuration and can be multiple hops away. The administrative burden grows with the number of BGP neighbors.

You use the **`listen-range`** command to specify a range of trusted IPv4 or IPv6 addresses to be added dynamically as neighbors to a BGP peer group within a Virtual Routing and Forwarding (VRF) instance. You can also specify the number of BGP neighbors to be accepted for this listen range.

You can use the **listen-limit** *limit* keyword of the **listen-limit** command to specify the number of BGP neighbors (from 1 to 2400) accepted for the specified listen range.

The following example configures two BGP peer groups (group1 and group2) under a VRF instance named violet. In this example, group1 is assigned 200 as its remote autonomous system (AS) number and the IPv4 address range 192.0.2.0/24 as a listen range. Also, group2 is assigned 300 as its remote AS number and the IPv6 address range 2001:db8:1::/64 as a listen range:

```
device# configure terminal
device(config)# vrf violet
device(config-vrf-violet)# router bgp
device(config-vrf-bgp)# peer-group group1
device(config-vrf-bgp-pg)# remote-as 200
device(config-vrf-bgp-pg)# listen-range 192.0.2.0/24
device(config-vrf-bgp-pg)# exit
device(config-vrf-bgp)# peer-group group2
device(config-vrf-bgp-pg)# remote-as 300
device(config-vrf-bgp-pg)# listen-range 2001:db8:1::/64
device(config-vrf-bgp-pg)#
```

Key Benefits

BGP Peering with Listen Range has the following benefits:

- Reduced administrative overhead to eliminate the need for static peer configurations
- Dynamic peer creation so that new peers inherit attributes from the associated peer group
- Support for both IPv4 and IPv6 peers
- Bidirectional Forwarding Detection (BFD) support for dynamic neighbors
- MD5 authentication support for dynamic neighbors

Note

- No default listen range exists. It must be manually configured.
- Each IP address can only belong to one listen range.
- Static neighbors take precedence over dynamic neighbors.
- Link-local IPv6 neighbors are not supported.
- Peer group configurations are inherited by dynamic neighbors.

How Listen Range Works

The BGP Listen Range feature streamlines the process by letting you define a range of IP addresses that can establish dynamic peering relationships. When a device detects an incoming TCP session from an IP address within the specified range, it creates a new BGP peer automatically.

Limitations

Extreme OS ONE does not support using link local addresses as BGP IPv6 peers.

BGP Peering with Next Hop Resolution

You can enable BGP next hop address resolution via the default route in a specific address family forwarding table within a VRF instance. You can also do this through BGP routes. These two features are disabled by default.

Enable BGP Next-Hop Address Resolution via the Default Route

The following example configures a VRF instance named default-vrf. In this example, the VRF provides BGP next hop address resolution via the default route within the IPv4 address family and unicast subaddress family:

```
device# configure terminal
device(config)# vrf default-vrf
device(config-vrf-default-vrf)# router bgp
device(config-vrf-bgp)# address-family ipv4 unicast
device(config-vrf-bgp-ipv4u)# next-hop-enable-default
device(config-vrf-bgp-ipv4u)#
```

Enable BGP Next-Hop Address Resolution through BGP Routes

The following example configures a VRF instance named default-vrf. In this example, the VRF provides BGP next-hop address resolution through BGP routes within the IPv6 address family and unicast subaddress family:

```
device# configure terminal
device(config)# vrf default-vrf
device(config-vrf-default-vrf)# router bgp
device(config-vrf-bgp)# address-family ipv6 unicast
device(config-vrf-bgp-ipv6u)# next-hop-recursion
device(config-vrf-bgp-ipv6u)#
```

BGP4 Message Types

BGP4 messages are of the following types: OPEN, UPDATE, NOTIFICATION, KEEPALIVE, or ROUTE-REFRESH.

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.

OPEN message

After establishing a TCP connection, BGP peers exchange OPEN messages 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 is supported.

Autonomous System

Both 2-byte and 4-byte autonomous system numbers are supported.

BGP Timers and Keepalives

BGP timers play a crucial role in maintaining session stability and ensuring fast convergence. By customizing these timers, you can optimize network performance. However, finding the right balance is key, because overly short timers can cause unnecessary session flaps, while longer timers can delay failure detection.

By understanding and customizing BGP timers, you can optimize network performance and ensure high availability. You can modify the following three primary timers:

1. Keepalive Timer: Sends periodic messages to ensure the connection between BGP peers remains active. Default interval is 60 seconds
2. Hold Timer: Defines the maximum time a BGP router waits without receiving messages from a neighbor before considering the session down. Default hold time is 180 seconds (3 minutes)
3. Connect Retry Timer: Manages the interval between attempts to re-establish a connection to a BGP peer after a session was torn down

You can modify the Keepalive, Hold, and Connect timers at the peer group level. The timers and keepalives use the following default values:

- Keepalive: 60 seconds
- Hold: 180 seconds
- Connect: 30 seconds

BGP Identifier

The BGP Identifier indicates the device ID of the sender. In Extreme OS ONE, you must configure the BGP Identifier or router-id manually. If not configured, the system does not use a default value.

Parameter List

You can configure an optional list of additional parameters used in peer negotiation.

UPDATE message

The UPDATE message advertises new routes, withdraws previously advertised routes, or both. The UPDATE message passes BGP4 attributes to describe the characteristics of a BGP path by the advertising device.

Withdrawn routes length	Withdrawn routes	Total path attributes len	Path attributes	NLRI
2	variable	2	variable	variable

Withdrawn Routes Length

The Withdrawn Routes Length indicates the length of the next (withdrawn routes) field. It can be 0.

Withdrawn Routes

Withdrawn Routes contains a list of routes (or IP-prefix or Length) to indicate routes being withdrawn.

Total Path Attribute Len

The Total Path Attribute Len indicates the length of the next (path attributes) field. It can be 0.

Path Attributes

Path Attributes indicates characteristics of the advertised path. Possible attributes are Origin, AS Path, Next Hop, MED (Multi-Exit Discriminator), Local Preference, Atomic Aggregate, Aggregator, Community, extended-Communities, and large-Communities. All well-known attributes, as described in RFC 4271, are supported.

NLRI

Network Layer Reachability Information (NLRI) is 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

If an error causes the TCP connection to close, the closing peer sends a notification message to indicate the type of error.

Error code	Error subcode	Error data
1	1	variable

Error Code

The Error Code indicates the type of error:

- Message header
- Open message
- Update message
- Hold timer expired

- Finite state-machine
- Cease (voluntarily)

Error Subcode

The Error Subcode provides specific information about the reported error.

Error Data

Error Data provides data based on the error code and the subcode.

KEEPALIVE message

Because BGP does not exchange route updates regularly to maintain a session, it sends KEEPALIVE messages to keep the session alive. The KEEPALIVE time specifies how frequently the device sends KEEPALIVE messages to its BGP4 neighbors.

A KEEPALIVE message contains the BGP header without a data field. The 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 message is useful when the inbound policy has been changed.

BGP Best Path Selection

When multiple paths are available, Border Gateway Protocol (BGP) uses a step-by-step process to select the preferred route to a destination. This process is critical in large scale networks such the Internet, where redundant paths between Autonomous Systems (ASs) or networks often exist.

How BGP Selects the Best Route

BGP evaluates a series of attributes in a specific order to determine the best route. The attributes are checked one by one, and the first matching criterion determines the best route.

Following is the typical sequence for route selection:

1. Highest Local Preference: BGP prefers routes with the highest LOCAL_PREF value, which indicates the preferred exit point in an AS.
2. Shortest AS Path: A shorter AS Path is preferred, because it indicates fewer hops and a more direct route.
3. Lowest Origin Type: BGP prefers routes with the lowest origin type: IGP (Interior Gateway Protocol) over EGP (Exterior Gateway Protocol) over INCOMPLETE.
4. Lowest MED (Multi-Exit Discriminator): A lower MED value is preferred to influence path selection between ASs.
5. EBGP over IBGP: Routes learned from External BGP (EBGP) neighbors are preferred over those from Internal BGP (IBGP) neighbors.

6. Lowest IGP Metric to Next-Hop: BGP prefers routes with the lowest IGP metric to the next-hop router.
7. Shortest Cluster Length: If all else is equal, BGP prefers routes with the shortest cluster length.
8. Older Route: If all attributes are equal, BGP might prefer the older route.
9. Lowest Router ID: Finally, BGP uses the lowest router ID to break ties.

Key Points

These steps apply to IPv4 and IPv6 unicast routes. You cannot modify them. BGP EVPN routes have additional route calculation steps.

BGP Route Origination through Redistribution

BGP establishes sessions with peers to exchange routes, which can be received from other peers or originated locally. One way to originate routes within BGP is through route redistribution, which involves importing routes from one routing protocol into BGP or vice versa. Networks running different routing protocols can exchange routing information to provide greater flexibility in managing multiple routing domains.

How BGP Redistribution Works

When routes are redistributed into BGP, they undergo the regular route selection process. If selected, they go through the regular route advertisement process.

The imported routes inherit default path attribute values, including the following values:

- Local Preference: 100
- Origin: Incomplete
- AS PATH: Empty
- Communities: None
- Ext-Communities: None

Key Points

BGP redistribution operates in the following ways:

- Redistribution is disabled by default and requires the **`table-connections`** command to be configured.
- Route policy association with redistribution is not currently supported.
- Redistributed routes take on default path attribute values.
- BGP considers protocol routes from the best source as the routes to be redistributed.
- BGP cannot redistribute its own routes to prevent routing loops, although it can pick routes from other VRFs (not currently supported).
- Redistributed routes undergo regular route selection and advertisement within BGP.

Configuring BGP Redistribution

To enable BGP redistribution, you use the **table-connections** command to enter VRF table connections configuration (conf-vrf-*name*-table-connections) mode. This is the mode for configuring table connection settings for the source and destination protocols for route redistribution for a specified address family. For details about this command, see the *Extreme OS ONE Switching and Routing Command Reference Guide*.

While in this mode, you use the **src-protocol** command to add table connections configurations for route redistribution to a specific VRF instance. This command configures the routing of redistribution between pairs of protocols for specified address families within a VRF instance. You use this command to add pairs of protocols and the address family (IPv4 or IPv6) that applies to each pair. For details about this command, see the *Extreme OS ONE Switching and Routing Command Reference Guide*.

The following example enables table connections configurations for a VRF instance named red and configures a connections table with four pairs of protocols and the address family (either IPv4 or IPv6) that applies to each pair:

```
device# configure terminal
device(config)# vrf red
device(config-vrf-red)# table-connections
device(config-vrf-red-table-connections)# src-protocol connected dst-protocol bgp address-
family ipv6
device(config-vrf-red-table-connections)# src-protocol connected dst-protocol bgp address-
family ipv4
device(config-vrf-red-table-connections)# src-protocol static dst-protocol bgp address-
family ipv4
device(config-vrf-red-table-connections)# src-protocol static dst-protocol bgp address-
family ipv6
device(config-vrf-red-table-connections)#
```

The following example displays the configuration of a VRF instance named red that is running currently on the device. In this example, the instance is configured with a connections table containing several pairs of protocols and the corresponding address family for each pair:

```
device# show running-config vrf

vrf red
table-connections
src-protocol connected dst-protocol bgp address-family ipv6
src-protocol connected dst-protocol bgp address-family ipv4
src-protocol static dst-protocol bgp address-family ipv4
src-protocol static dst-protocol bgp address-family ipv6
device#
```

If you enable the redistribution of routes from one protocol to another for the specified address family, BGP checks for the default route in the redistribution slot and does not add it to the BGP routing table. But if this default route from redistribution must be used, you enter the **send-default-route** command for the specified address family to ensure its redistribution to the BGP routing table (thereby advertising it to neighbors). For details about this command, see the *Extreme OS ONE Switching and Routing Command Reference Guide*.

The following example configures a routing process under a VRF instance named violet. This example creates an IPv4 unicast address family in the routing process and overrides the exclusion of the default route from the BGP routing table when route redistribution between protocols is enabled:

```
device# configure terminal
device(config)# vrf violet
device(config-vrf-violet)# router bgp
device(config-vrf-default-bgp)# address-family ipv4 unicast
device(config-vrf-bgp-l2vpn-evpn)# send-default-route
device(config-vrf-bgp-l2vpn-evpn)#
```

The following example displays the configuration of a VRF instance named violet that is running currently on the device. This example overrides the exclusion of the default route from the BGP routing table when route redistribution between protocols is enabled for the IPv4 unicast address family and the IPv6 unicast address family:

```
device# show running-config vrf violet

vrf violet
  table-connections
    src-protocol static dst-protocol bgp address-family ipv4
    src-protocol static dst-protocol bgp address-family ipv6
  !
  static-route 192.0.2.1/32 192.0.2.101 enable-bfd profile default
  static-route 2001:db8:20:1:1::/64 2001:db8:30:1:1::65 enable-bfd profile computes
  static-route 192.0.2.8/32 192.0.2.201 enable-bfd profile default
  static-route 2001:db8:20:1:5::/64 2001:db8:30:1:5::65 enable-bfd profile computes
  static-route 198.51.100.0/24 198.51.100.10
  static-route 2001:db8:26:1:1:100::/64 2001:db8:25:1:1::100
  router bgp
    local-as 10001
    router-id 203.0.113.2
    graceful-restart
    use-multiple-paths ebgp maximum-paths 8
    address-family ipv4 unicast
      graceful-restart
      send-default-route
      activate
    !
    address-family ipv6 unicast
      graceful-restart
      send-default-route
      activate
    !
  !
!
device#
```

BGP Route Origination through the Network

When establishing BGP sessions with peers, networks exchange routes learned from other peers or originated locally. One way to originate routes in BGP is through the **network** command, which lets you selectively inject local IGP routes into the BGP routing table.

How it Works

When a route is added via the **network** command, it undergoes the standard route selection process. If selected, it is then advertised to peers through the regular route advertisement process. The route must be available in the specified address family (such as IPv4 or IPv6 unicast).

Key Differences and Benefits

Unlike route redistribution, the **network** command provides granular control over which routes are imported into the BGP routing table. Imported routes inherit default path attribute values such as the following values:

- Local Preference: 100
- Origin: IGP
- AS PATH: Empty
- Communities: None
- Ext-Communities: None

Key Points

BGP route origination through a network has the following limitations:

- No default network routes exist (you must configure these routes explicitly.
- Route policy association with the **network** command is not supported.
- Network routes assume default path attribute values.
- The best source for network routes cannot be BGP itself.
- Imported routes follow standard route selection and advertisement processes within BGP.

Configuring BGP Route Origination through Network

The following example configures a routing process under a VRF instance named violet. This example creates two address families and advertises a network in each one:

```
device# configure terminal
device(config)# vrf violet
device(config-vrf-violet)# router bgp
device(config-vrf-bgp)# address-family ipv4 unicast
device(config-vrf-bgp-ipv4u)# network 192.0.2.1/32
device(config-vrf-bgp-ipv4u)# exit
device(config-vrf-bgp)# address-family ipv6 unicast
device(config-vrf-bgp-ipv6u)# network 2001:db8:1:1::2/32
device(config-vrf-bgp-ipv6u)#
```

The following example displays the configuration of a VRF instance named violet that is running currently on the device. In this example, the instance contains a routing process with two address families and an advertised network in each one:

```
device# show running-config vrf violet

vrf violet
  router bgp
```

```
    address-family ipv4 unicast
      network 192.0.2.1/32
    !
    address-family ipv6 unicast
      network 2001:db8:1:1::2/32
      activate
    !
  !
!
device#
```

BGP Route Origination through Default Route

In BGP, routers advertise routes to neighbors with next-hop information. The receiving device validates and installs these routes in its routing table, pointing to the next hop. When incoming traffic is received, the device performs a routing table lookup, making a routing decision based on the longest match. If no match is found, the packet is typically dropped.

However, in certain scenarios, a device might want to attract traffic when there is no matching route. To achieve this, it can advertise a default route (0.0.0.0/0 for IPv4 or ::/0 for IPv6) to its neighbors. This special route always matches, and the receiving neighbors install it in their hardware to point to the advertised next hop. This feature overrides exclusion of the default route from the BGP routing table when redistribution of routes between protocols is enabled for the specified address family.

A BGP speaker can advertise default routes to its neighbors to attract traffic or act as a default gateway. You can use multiple methods to advertise default routes in BGP, including the following commands:

1. **`network`**
2. **`redistribute`**
3. **`default-route-originate`** under a specific BGP peer group
4. **`send-default-route`** under a specific BGP IPv4 or IPv6 address family

These methods let BGP speakers advertise default routes to enable flexible routing decisions and traffic management.

CLIs for BGP Default Route Origination at the Global Level

Follow this procedure to configure BGP route origination through a default route globally:

1. Access global configuration mode.

   ```
   device# configure terminal
   ```

2. Specify the VRF name and enter VRF configuration mode. You can specify the default VRF (named default-vrf) or a user-created VRF.

   ```
   device(config)# vrf violet
   ```

3. Configure a BGP routing process within the VRF instance and enter BGP configuration mode.

```
device(config-vrf-violet)# router bgp
```

4. Configure an address family routing table for the BGP routing process within the VRF instance and enter BGP address family configuration mode. You can specify **ipv4** or **ipv6** for the address family. You can specify **unicast**, **multicast**, or **evpn** for the address prefixes.

```
device(config-vrf-bgp)# address-family ipv4 unicast
```

5. Enable BGP route origination through the default route for the specified address family.

```
device(config-vrf-bgp-ipv4u)# send-default-route
```

6. (Optional) Enable BGP route origination through the default route for the specified address family by advertising a network to its routing table. The following command identifies which networks to advertise from your local networks. If BGP finds a network in the local routing table that matches the network statement (and the mask), then BGP advertises it.

```
device(config-vrf-bgp-ipv4u)# network 0.0.0.0/0
```

7. Activate the routing table.

```
device(config-vrf-bgp-ipv4u)# activate
```

8. (Optional) Verify the configuration.

```
device(config-vrf-bgp-ipv4u)# do show running-configuration vrf violet

vrf violet

  table-connections
    src-protocol static dst-protocol bgp address-family ipv4
    src-protocol static dst-protocol bgp address-family ipv6
  !
  static-route 192.0.2.1/32 192.0.2.101 enable-bfd profile default
  static-route 2001:db8:20:1:1::/64 2001:db8:30:1:1::65 enable-bfd profile computes
  static-route 192.0.2.8/32 192.0.2.201 enable-bfd profile default
  static-route 2001:db8:20:1:5::/64 2001:db8:30:1:5::65 enable-bfd profile computes
  static-route 198.51.100.0/24 198.51.100.10
  static-route 2001:db8:26:1:1:100::/64 2001:db8:25:1:1::100
  router bgp
    local-as 10001
    router-id 203.0.113.2
    graceful-restart
    use-multiple-paths ebgp maximum-paths 8
    address-family ipv4 unicast
      graceful-restart
      send-default-route
      network 0.0.0.0/0
      activate
    !
    address-family ipv6 unicast
      graceful-restart
      send-default-route
      network 2001:db8:10:1:1::2/32
      activate
    !
  !
!
device#
```

CLIs for BGP Route Default Route Origination at the Peer Group Level

Using network and redistribution, a default route is added into the BGP routing table and is advertised to all peers and peer groups that negotiated a given AFI (Address Family Identifier) or SAFI (Subsequent Address Family Identifier). This is not always desirable where you must advertise to limited or few peer groups. To do so, use the **default-route-originate** command under a peer group to which the default route must be originated.

Follow this procedure to configure BGP route origination through a default route for a specific BGP peer group:

1. Access global configuration mode.

```
device# configure terminal
```

2. Specify the VRF name and enter VRF configuration mode. You can specify the default VRF (named default-vrf) or a user-created VRF.

```
device(config)# vrf violet
```

3. Configure a BGP routing process within the VRF instance and enter BGP configuration mode.

```
device(config-vrf-violet)# router bgp
```

4. Specify a BGP peer group.

```
device# (config-vrf-bgp)# peer-group towards_leaf
```

5. Configure an address family routing table for the BGP routing process within the VRF instance and enter BGP peer group address family configuration mode. You can specify **ipv4** or **ipv6** for the address family. You can specify **unicast**, **multicast**, or **evpn** for the address prefixes.

```
device(config-vrf-bgp-pg)# address-family ipv4 unicast
```

6. Enable BGP route origination through the default route for the specified address family.

```
device(config-vrf-bgp-pg-ipv4u)# default-route-originate
```

7. (Optional) Verify the configuration.

```
device(config-vrf-bgp-pg-ipv4u)# do show running-configuration vrf violet

vrf violet
  table-connections
    src-protocol static dst-protocol bgp address-family ipv4
    src-protocol static dst-protocol bgp address-family ipv6
  !
  static-route 192.0.2.1/32 192.0.2.101 enable-bfd profile default
  static-route 2001:db8:20:1:1::/64 2001:db8:30:1:1::65 enable-bfd profile computes
  static-route 192.0.2.8/32 192.0.2.201 enable-bfd profile default
  static-route 2001:db8:20:1:5::/64 2001:db8:30:1:5::65 enable-bfd profile computes
  static-route 198.51.100.0/24 198.51.100.10
  static-route 2001:db8:26:1:1:100::/64 2001:db8:25:1:1::100
  router bgp
    local-as 10001
    router-id 203.0.113.2
    graceful-restart
    use-multiple-paths ebgp maximum-paths 8
    peer-group towards_leaf
      address-family ipv4 unicast
```

```
        default-route-originate
    !
    address-family ipv4 unicast
      graceful-restart
      send-default-route
      activate
    !
    address-family ipv6 unicast
      graceful-restart
      send-default-route
      activate
    !
  !
!
device#
```

BGP Additional Paths

The BGP Additional Paths feature advertises and receives multiple paths for a given prefix to promote path diversity. This is achieved by introducing a path identifier (ID) that extends the existing NLRI encoding. This feature enhances path diversity and reduces path hiding. Key aspects of this feature include the following functionality:

- Negotiation: Must be negotiated before exchanging additional paths.
- Path ID: A four-octet value that uniquely identifies each path within the NLRI.
- RIB-IN table: Prefix and path ID together serve as the key, enabling multiple route sources from a given peer for the same route.
- Path diversity: Multiple paths can be received and maintained, thereby improving network resilience.
- Faster recovery: The network can recover more quickly from next-hop failures.

By default, BGP routers only advertise their best path to neighbors, replacing the current path when a better one is found. This leads to "path hiding," where many possible paths are unknown to some routers.

The BGP Additional Paths feature addresses this limitation by advertising multiple paths for the same prefix, enabling path diversity. This feature adds a unique path identifier to each path, allowing for more efficient routing.

Key Components

By implementing the following key components, the BGP Additional Paths feature enhances path diversity and improves network resilience in the following ways:

- Configures receive-mode functionality: The system sets up the feature at the global AFI or SAFI level or peer-group AFI SAFI level to allow the router to receive additional paths.
- Negotiates ADD-PATH capability: The system negotiates the ADD-PATH capability with its peers in "receive" mode only.
- Modifies NLRI processing: The system updates the NLRI processing behavior to decode extended NLRI and extract the path ID for each path.

Implementation Considerations and Limitations

Without the Additional Paths feature, a BGP device advertises only its best path to neighboring devices. When a device receives multiple paths for the same prefix from the same peer, it replaces the previous path. With the Additional Paths feature, BGP devices can negotiate the ability to accept multiple paths from the same peer. Each path is assigned a unique path identifier.

The Additional Paths feature has the following implementation considerations and limitations:

- The feature is not enabled by default and requires explicit configuration.
- The feature supports IPv4 and IPv6 unicast address families but not BGP EVPN.
- Local changes to the capability take effect only after a session reset.
- The feature supports receiving additional paths. It does not support ending additional paths or advertising additional paths to peers.
- The system can receive multiple paths for a given prefix from the same peer.
- The device can negotiate the Additional Paths capability with its peers.

CLI commands are available to enable the feature at the global AFI-SAFI and peer-group AFI-SAFI levels.

Various **show** commands are available to display Additional Paths information. The **show neighbor** command displays the Add-Path mode status of the local device and the peer device as well.

Configuring BGP Additional Paths

The following example configures a routing process under a VRF instance named violet. This example creates and activates two address families. In this example, additional paths receive functionality is enabled for each address family:

```
device# configure terminal
device(config)# vrf violet
device(config-vrf-violet)# router bgp
device(config-vrf-bgp)# address-family ipv4 unicast
device(config-vrf-bgp-ipv4u)# add-paths receive
device(config-vrf-bgp-ipv4u)# activate
device(config-vrf-bgp-ipv4u)# exit
device(config-vrf-bgp)# address-family ipv6 unicast
device(config-vrf-bgp-ipv6u)# add-paths receive
device(config-vrf-bgp-ipv6u)# activate
device(config-vrf-bgp-ipv6u)#
```

The following example displays the configuration of a VRF instance named violet that is running currently on the device. In this example, the instance contains a routing process with two activated address families. Each address family uses the additional paths receive functionality:

```
device# show running-config vrf violet

vrf violet
  router bgp
    address-family ipv4 unicast
      add-paths receive
```

```
      activate
    !
    address-family ipv6 unicast
      add-paths receive
      activate
    !
  !
!
device#
```

Support Additional Paths in BGP

By default, when a new path or update is received for the same prefix, Extreme OS ONE BGP handles it according to implicit withdraw behavior. However, with the additional paths feature, Extreme OS ONE BGP can be configured to accept and process multiple paths for the same prefix. How many additional paths can be supported depends on the RIB-IN soft limit of the BGP peer. This limit determines the maximum number of routes that can be stored in the BGP routing table, which in turn affects the number of additional paths that can be accepted and processed.

Note
Enabling the Additional Paths feature will reset the BGP session to renegotiate the capability and process updated NLRI preceded by path-id.

Enabling Additional Paths

You can enable this feature at the global AFI or SAFI level (to apply to all peers) or at the peer-group AFI or SAFI level (to apply to a specific peer group).

Configuring Additional Paths in BGP

To enable additional paths in receive mode, use the following command:

```
dutb(config-vrf-bgp-ipv4u)# add-paths receive
```

This command enables the receive mode for additional IPv4 unicast paths, allowing Extreme OS ONE BGP to accept and process multiple paths for the same IPv4 prefix.

Verification

After enabling the feature, you can verify the configuration using the **`show running-config`** command:

```
dutb(config-vrf-bgp-ipv4u)# do show running-config
```

This command displays the current configuration, including the **`add-paths receive`** command.

Capability Negotiation for Add Path

The Add Path capability is identified by capability code 69, with a variable length. The Value field contains the following information:

1. AFI (Address Family Identifier): Specifies the address family (IPv4 or IPv6)

2. SAFI (Subsequent Address Family Identifier): Specifies the subsequent address family
3. Send/Receive: Indicates the ability of the sender to receive or send multiple paths:
   - Send multiple paths to its peer (value 2)
   - Receive multiple paths from its peer (value 1)
   - Both (value 3)

Because Extreme OS ONE BGP supports receive mode only, the Send/Receive field value is always 1 when sending capability information to neighbors.

NLRI Processing

After negotiating the Additional Paths capability, BGP uses update message NLRI to receive multiple paths. The NLRI format is updated to include a new field, Path Identifier (Path-ID), which is 4 octets in length and precedes the path itself. The Path-ID uniquely identifies each path for a prefix within a peering session.

The following is the updated NLRI format:

```
+--------------------------------+
| Path Identifier (4 octets)     |
+--------------------------------+
| Length (1 octet)               |
+--------------------------------+
| Prefix (variable)              |
+--------------------------------+
```

Path-Id uniquely identifies each path for a prefix in a peering session.

BGP receives multiple paths but installs only the best paths to the Unified Forwarding Table Manager (UFTM) based on ECMP configuration and limits set by the **use-multiple-paths** command. You use this command to configure the maximum number of paths to install for ECMP. If all received paths qualify as best paths, multiple paths are installed to UFTM up to the configured limit.

Process Additional Paths

When a neighboring router sends an update message with additional paths, the paths are decoded and extracted. Each path is identified by a unique Path-ID, which is used to track future updates for that path. The paths undergo regular path selection criteria, and only the best route is chosen and installed to Unified Forwarding Table Manager (UFTM). The key benefits include path diversity (accepting additional paths provides efficient use of multiple paths and hitless planned maintenance) and Improved network utilization (multiple paths can be used to optimize network resource utilization).

The following examples show prefix 192.0.2.13/24 being received from a remote BGP device with three different path IDs (1, 2, and 555):

- Routes shown in the **received-route** option of the peer:

```
device# show bgp vrf default-vrf neighbor 192.0.2.2 received-routes ipv4-unicast
detail

VRF Name: default-vrf
Total number of routes: 3

Prefix: 192.0.2.13/24, Nexthop: 192.0.2.2, Path ID: 1, AS Path:
   Age: 6s, Status: VALID Best-Path: Yes
   Origin: IGP, Local Preference: 100, MED: -, Weight: -, Admin Distance:
   Communities: -
   Extended-Communities:

Prefix: 192.0.2.13/24, Nexthop: 192.0.2.2, Path ID: 2, AS Path:
   Age: 6s, Status: VALID Best-Path: No
   Origin: IGP, Local Preference: 100, MED: -, Weight: -, Admin Distance:
   Communities:
   Extended-Communities:

Prefix: 192.0.2.13/24, Nexthop: 192.0.2.2, Path ID: 555, AS Path:
   Age: 65, Status: VALID Best-Path: No
   Origin: IGP, Local Preference: 100, Weight: -, Admin Distance:
   Communities:
   Extended-Communities:
device#
```

- Routes shown in the **routes-summary** option:

```
device# show bgp vrf default-vrf routes ipv4-unicast 192.0.2.13/24

Prefix: 192.0.2.13/24, Nexthop: 192.0.2.2, Peer: 192.0.2.2, Path ID: 1, Age: 5m20s,
Status: VALID Best-Path: Yes
AS Path:
Origin: IGP, Local Preference: 100, MED: -, Weight: -, Admin Distance:
Communities:
Extended-Communities: -

Prefix: 192.0.2.13/24, Nexthop: 192.0.2.2, Peer: 192.0.2.2, Path ID: 2, AS Path: Age:
5m20s, Status: VALID Best-Path: No
Origin: IGP, Local Preference: 100, MED: -, Weight: -, Admin Distance:
Communities:
Extended-Communities:

Prefix: 192.0.2.13/24, Nexthop: 192.0.2.2, Peer: 192.0.2.2, Path ID: 555, AS Path:
Age: 5m20s, Status: VALID Best-Path: No
Origin: IGP, Local Preference: 100, MED: -, Weight: -, Admin Distance:
Communities:
Extended-Communities:
device#
```

YANG and Configuration Commands

The feature is supported through YANG models. You can enable Add-Path at the global, peer group, and peer levels.

You can display information about Add-Path via **show** commands including **show bgp vrf** *vrf-name* **summary** *afi-safi* and **show bgp vrf** *vrf-name* **neighbor** *nbr-ip*.

Add-Path provides path diversity by accepting multiple paths for the same prefix. The Path-ID uniquely identifies each path by enabling efficient tracking and management of multiple paths. Regular path selection criteria apply to choose only the best route and installed to UFTM.

For details on syntax and parameters, see the *Extreme OS ONE Switching and Routing Command Reference Guide*. For details on YANG modules, see the *Extreme OS ONE Switching and Routing YANG Reference Guide*.

Event Log Messages for BGP Add-Path

These log messages provide valuable information for troubleshooting and monitoring BGP Add-Path functionality. The system generates event log messages for various BGP Add-Path events, such as:

1. **Receiving and decoding ADD-PATH capability:** Logs show when BGP receives and decodes the ADD-PATH capability from a peer:
   - decodeCapability: Received Capability VrfName[default-vrf] Peer[13.1.1.2] CapCode[69] CapLen[4] CapValue[]
   - decodeAddPathCap: Decoded ADD-PATH capability Peer[13.1.1.2] CapLen[4] Afi[1] Safi[1] Mode[3]
2. **Receiving new path with path-id:** Logs indicate when BGP receives a new path with a valid path-id for a given prefix:
   - decodePathIDFromNlri: Path-ID extracted from BGP-Peer Peer[13.1.1.2] NlriLen[4] Path-id[1]
   - decodeIPNlri: Path-ID received for prefix VrfName[default-vrf] Peer[13.1.1.2] Prefix[?0d0d0d] NlriLen[3] Path-id[1]

BGP Allow-Own-AS

The BGP Allow-Own-AS feature helps you manage route advertisements and prevent routing loops in complex BGP setups. BGP normally rejects routes containing its own Autonomous System (AS) number in the AS path to avoid loops. However, in certain scenarios, this check might need to be bypassed.

Why Allow-Own-AS?

In multi-homed networks (connected to multiple upstream ASs), a router might need to advertise routes back to a peer or accept routes with its own AS number in the path. Allowing this can prevent route discard due to standard anti-loop checks. By carefully managing Allow-Own-AS, you can optimize route advertisements and maintain network stability.

Key Points

This feature is disabled by default, which means that routers reject routes with their own AS number. When enabled, BGP accepts routes with its own AS number in the AS path.

This feature is disabled by default, which means that routers reject routes with their own AS number.When enabled, BGP accepts routes with its own AS number in the AS path.

You enable this feature at the peer group level. Enabling it triggers a route refresh message to all peers in the group. Disabling it scans the Routing Information Base In (RIBIN) and removes routes with the device AS number.

Configuring BGP Allow-Own-As

Use the **allow-own-as** command to set how many times the router's own AS number (local AS) can appear in the AS path of a BGP update from peers of this peer group before being rejected or marked as invalid.

The following example configures a BGP peer group named group1 under a Virtual Routing and Forwarding (VRF) instance named violet. A BGP session within the peer group is reset when the local AS number appears in that session for the 11th time:

```
device# configure terminal
device(config)# vrf violet
device(config-vrf-violet)# router bgp
device(config-vrf-bgp)# peer-group group1
device(config-vrf-bgp-pg)# allow-own-as 10
device(config-vrf-bgp-pg)#
```

The following example displays the configuration of a VRF instance named violet that is running currently on the device. In this example, any BGP session in the peer group can encounter the local AS number up to 10 times before being reset

```
device# show running-config vrf violet

vrf violet
 !
   router bgp
     peer-group group1
       allow-own-as 10
!
device#
```

:

BGP Local-AS-Forced

You can override the router local AS number of BGP peers to establish sessions with old BGP peers supporting only two-byte AS numbers. By default, the local AS number override feature is disabled for the specified BGP peer group.

Configuring BGP Local-AS-Forced

Use the **local-as-forced** command to override the router's local autonomous system (AS) number with a forced two-byte AS number to establish sessions with peers of the BGP peer group for which this command is configured.

The following example configures a BGP peer group named group1 under a VRF instance named violet. This example uses the **local-as-forced** command to override the configured local ASs of the router for members of group1 and forces them to use 100 as the AS number instead:

```
device# configure terminal
device(config)# vrf violet
device(config-vrf-violet)# router bgp
device(config-vrf-bgp)# peer-group group1
device(config-vrf-bgp-pg)# local-as-forced 100
device(config-vrf-bgp-pg)#
```

The following example displays the configuration of a VRF instance named violet that is running currently on the device. In this example, members of the group1 peer group are forced to use 100 as the AS number instead of their routers' configured local AS numbers:

```
device# show running-config vrf violet

vrf violet
!
  router bgp
    peer-group group1
      local-as-forced 100
!
device#
```

Configure BGP MD5 Authentication

MD5 authentication provides robust security for BGP sessions to ensure the integrity and authenticity of exchanged messages.

MD5 authentication is a crucial security feature in Border Gateway Protocol (BGP) that ensures the authenticity and integrity of BGP sessions between routers. It prevents unauthorized session initiation, protects against malicious attacks, and verifies the legitimacy of BGP peers.

Use the **auth-password** command to specify a password for MD5 authentication to be used by all members of the specified BGP peer group within a Virtual Routing and Forwarding (VRF) instance.

```
device# configure terminal
device(config)# vrf violet
device(config-vrf-violet)# router bgp
device(config-vrf-bgp)# peer-group group1
device(config-vrf-bgp-pg)# auth-password MyPassword1
device(config-vrf-bgp-pg)#
```

When two routers establish a BGP session, they exchange messages with a cryptographic hash generated using the MD5 algorithm and a shared secret key. The

receiving router recalculates the hash and compares it to the received hash. If they match, the session is authenticated. Otherwise, it is rejected.

No default password is set. Manual configuration of a password is required. Session expiration or immediate termination depends on password configuration.

This feature is limited to 75 peer groups with ListenRange for dynamic groups because of kernel memory limitations.

This feature has the following key benefits and features:

- Protects against unauthorized access and tampering
- Supports both IPv4 and IPv6 peers
- Supports dynamic neighbors
- Ensures integrity and authentication, but does not encrypt BGP messages
- Performs MD5 hashing and processing in the kernel for minimal computational overhead

BGP Multiprotocol

Multiprotocol BGP feature enhances the flexibility and scalability of BGP routing.

Multiprotocol BGP (MP-BGP) extends standard BGP4 to support the following address families:

- IPv4 and IPv6 addresses
- Unicast variants as follows: address-family IPv4 unicast, address-family IPv6 unicast, and address-family L2VPN EVPN

You use MP-BGP to configure BGP routers to communicate the types of routes to exchange with peers by specifying an Address Family Identifier (AFI) and a Subsequent Address Family Identifier (SAFI).

You can activate or deactivate each AFI or SAFI individually for flexible route exchange. MP-BGP is enabled by default and cannot be disabled. BGP routers negotiate with peers only for activated AFI or SAFI pairs.

BGP Route Refresh

You use BGP Route Refresh to configure BGP routers to request and exchange updated routing information without resetting BGP sessions. This is useful when routing policies change or peers are added or removed so that routers can obtain fresh routing information.

When a route policy is applied to a Routing Information Base (RIB), BGP accepts only routes that pass the filter and discards others. If the policy changes, BGP might not know which routes were filtered out. To resolve this, BGP sends a route refresh message to request a fresh copy of the peer's RIB. This feature streamlines routing information updates to reduce the need for manual intervention and improve network efficiency.

Key Points

This feature has the following key benefits and features:

- Route Refresh is enabled by default with no CLI option to disable.
- The capability is negotiated per Address Family Identifier (AFI) and Subsequent Address Family Identifier (SAFI).
- Route refresh can be manually triggered using the **clear bgp vrf** *vrf-name afi-safi* **neighbor all** command.
- This implementation supports only the basic route refresh capability, not the advanced route refresh capability that RFC 7313 defines.

Configure EBGP Multihop: Extending BGP Reachability

EBGP Multihop offers flexibility and scalability, which makes it essential for large, complex networks that span multiple locations or organizations.

External Border Gateway Protocol (EBGP) sessions are established between directly connected routers. With EBGP Multihop, BGP sessions can traverse multiple routers to provide greater flexibility and scalability for large networks.

This feature does not apply to IPv6 link local peers (which are single hop by nature).

Use the **ebgp-multihop** command to enable external BGP multihop and optionally set the time-to-live (TTL) value of the IP header (hop count) for a specific external BGP neighbor or all external neighbors in the specified BGP peer group within a VRF instance.

```
device# configure terminal
device(config)# vrf violet
device(config-vrf-violet)# router bgp
device(config-vrf-bgp)# peer-group group1
device(config-vrf-bgp-pg)# ebgp-multihop 10
device(config-vrf-bgp-pg)# exit
device(config-vrf-bgp)# peer-group group2
device(config-vrf-bgp-pg)# ebgp-multihop
device(config-vrf-bgp-pg)#
```

Key Benefits and Considerations

This feature has the following key benefits:

- Supports BGP sessions over intermediate routers
- Improves network management and policy control
- Supports scalable and fault-tolerant network designs
- Provides alternate data paths in case of link failures

This feature has the following key considerations:

- The default TTL is 1 for eBGP peers and 64 for iBGP peers.
- EBGP Multihop overrides TTL hop count for eBGP peers.
- This feature does not apply to IPv6 link-local peers (which are single-hop by nature).

Comparison to Standard BGP

Standard BGP requires direct router connections. EBGP Multihop provides peering over multiple hops, making it ideal for expansive networks.

BGP Fast External Failover

BGP Fast External Failover accelerates BGP convergence when an external link or peer fails. This feature is beneficial in environments requiring rapid routing path recovery, such as ISP networks or critical infrastructure that relies on BGP.

In traditional BGP, failure detection, route withdrawal, and routing table re-computation can be time-consuming, leading to significant disruptions in data traffic. But, the BGP Fast External Failover feature provides continuous monitoring of BGP peers over a given link. Upon detecting a link failure, BGP takes immediate corrective action without waiting for traditional timeout mechanisms. This is achieved through direct detection of link failures and rapid notification and response. By using BGP Fast External Failover, networks can minimize downtime and ensure faster recovery from external link or peer failures. Multihop scenarios rely on BFD for failure detection.

This feature is only applicable to eBGP peers and is suitable for directly connected eBGP peers.

This feature is not enabled by default. Explicit configuration required.

Configuring BGP Fast External Failover

Use the **`fast-external-failover`** command to quickly terminate the EBGP session when a directly-connected link to the EBGP peer goes down, without waiting for the hold-down timer to expire.

The following example configures a BGP peer group named group1 under a Virtual Routing and Forwarding (VRF) instance named violet. In this example, the BGP session is reset if the link to an EBGP peer goes down:

```
device# configure terminal
device(config)# vrf violet
device(config-vrf-violet)# router bgp
device(config-vrf-bgp)# peer-group group1
device(config-vrf-bgp-pg)# fast-external-failover
device(config-vrf-bgp-pg)#
```

The following example displays the configuration of a VRF instance named violet that is running currently on the device. In this example, the BGP session is reset if the link to an EBGP peer in a peer group named group1 is lost:

```
device# show running-config vrf violet
vrf violet
!
  router bgp
    peer-group group1
      allow-own-as 10
      fast-external-failover
!
device#
```

BGP IPv6

The Border Gateway Protocol (BGP) IPv6 feature adds IPv6 support to BGP routing, enabling the exchange of routing information for IPv6 networks alongside traditional IPv4 networks. This feature includes:

- IPv6 Unicast Support: BGP can exchange IPv6 unicast routes, forwarding IPv6 packets across networks.
- IPv6 Address Family Configuration: Commands specific to IPv6 are configured independently of IPv4, providing flexibility in managing routing for different traffic types.
- Route Propagation and Filtering: IPv6 routes can be propagated across Autonomous Systems (ASs) with configurable policies for filtering or modifying route advertisements.
- Address Family Independent BGP Peering: IPv6 peering can be configured separately from IPv4, allowing organizations to maintain distinct peering for each address family.

IPv4 peers cannot carry IPv6 routes, and IPv6 peers cannot carry IPv4 routes. IPv6 Unicast operates independently of IPv4 Unicast.

An IPv6 address family must be explicitly activated.

For details on command syntax and parameters for configuring an address family, see the *Extreme OS ONE Switching and Routing Command Reference Guide*.

BGP IPv6 Prefix Advertisement over IPv4 BGP Peers

This document describes how to configure and manage IPv6 prefix advertisement over existing IPv4 BGP sessions using Extreme OS ONE. This feature exchanges IPv6 routing information through your IPv4 BGP infrastructure without establishing separate IPv6 BGP sessions.

This feature provides a powerful tool for IPv6 deployment while maintaining your IPv4 BGP infrastructure. Use the following configuration procedures and best practices to ensure successful implementation.

Overview of BGP IPv6 Prefix Advertisement over IPv4 BGP Peers

You can advertise IPv6 prefixes over existing IPv4 BGP peering sessions. This capability uses Multiprotocol Extensions for BGP (MP-BGP) to carry IPv6 routing information within IPv4 TCP connections.

The system encapsulates IPv6 routing information within MP-BGP update messages and transmits them over established IPv4 BGP sessions. Both routers must support and negotiate the IPv6 capability during the BGP session establishment process.

Key Components

The BGP IPv6 Prefix Advertisement over IPv4 BGP Peers uses existing IPv4 BGP sessions as transport and maintains complete IPv6 routing attributes (next-hop, AS_PATH, communities, and Multi-Exit Discriminator (MED)). The feature implements RFC 2858 (Multiprotocol Extensions for BGP4) and supports RFC 2545 for IPv6 routing information.

Benefits

The BGP IPv6 Prefix Advertisement over IPv4 BGP Peers feature has the following primary benefits.

Simplified Network Management

- Reduces the number of BGP sessions to maintain
- Eliminates duplicate session management overhead
- Uses a single IPv4 session for both IPv4 and IPv6 routing

Streamlined IPv6 Migration

The BGP IPv6 Prefix Advertisement over IPv4 BGP Peers feature streamlines IPv6 migration in the following ways:

- Transitions to IPv6 gradually while maintaining existing IPv4 infrastructure
- Avoids complex dual stack BGP configurations
- Reduces resource consumption on network devices

Operational Efficiency

The BGP IPv6 Prefix Advertisement over IPv4 BGP Peers feature increases operational efficiency in the following ways:

- Lowers configuration complexity
- Has fewer sessions to monitor and troubleshoot
- Provides consolidated routing policy management

Limitations

The BGP IPv6 Prefix Advertisement over IPv4 BGP Peers feature does not support IPv4-mapped IPv6 addresses on Layer 3 interfaces. Also, you must use route policies to ensure next hop reachability for native IPv6 addresses.

Session Behavior

Extreme OS ONEestablishes IPv4 BGP sessions when you activate both IPv4 and IPv6 address families at the router BGP level and at the peer group level. You can advertise IPv4 prefixes as well as IPv6 prefixes over one IPv4 BGP session.

Extreme OS ONE establishes IPv4 BGP sessions even if you have activated only IPv6 address family. You can send IPv6 prefixes over IPv4 sessions, but the system does not advertise IPv4 prefixes.

Resource Impact

Memory usage increases proportionally with the IPv6 routing table size. Standard BGP scaling limits apply to combined IPv4 and IPv6 operations. No additional performance or scalability challenges are expected.

Next-Hop Handling

Following is the order of priority in which the IPv6 next hop is picked up for IPv6 routes advertised to IPv4 BGP peers:

1. Route policy with **`set-next-hop`** applied for peer-group: This requires a route policy to be configured with **`set-next-hop`** set to any IPv6 address that is configured on the BGP peering interface and applying this route policy as an outbound/export policy for an IPv4 peer.

   This configuration overrides the default behavior, which picks the lowest IPv6 address of the IPv4 BGP session's source interface as the next hop of the IPv6 routes advertised to IPv4 BGP peers.
2. Lowest IPv6 address of the IPv4 BGP session's source interface is picked up as the next hop of the IPv6 routes advertised to IPv4 BGP peers if **`route-policy`** (with **`set-next-hop`**) is not applied for the IPv4 peer.
3. IPv4-mapped IPv6 address: If no IPv6 address is configured on the IPv4 BGP session's source interface, the IPv6 routes are advertised to IPv4 peers with the next hop in IPv4-mapped-IPv6 format ::*ffff*:*a.b.c.d*, where *a.b.c.d* is the IPv4 address of the BGP source interface.

Next-Hop Reachability Considerations

You must consider the following issues for Next-Hop Reachability of IPv4-mapped IPv6 addresses:

- Next-hop reachability depends on your IPv4 routing table.
- Standard IPv4 forwarding handles the packet delivery.

You must consider the following issues for Next-Hop Reachability of native IPv6 addresses:

- Configure IPv6 interfaces with matching addresses.
- Ensure that the IPv6 routing table contains reachable paths.
- Use route policies to set appropriate next-hop values.

Prerequisites

The BGP IPv6 Prefix Advertisement over IPv4 BGP Peers feature has the following prerequisites.

Network Prerequisites

Make sure that you prepare your network in the following ways:

- Establish existing IPv4 BGP peering relationships
- Ensure that both BGP peers support multiprotocol extensions for BGP
- Ensure that IPv6 connectivity is available for next hop reachability (when using native IPv6 next hops)

BGP Peer Prerequisites

Remote BGP speakers must support the MP-BGP IPv6 capability. Capability negotiation must succeed during BGP session establishment.

Enable the BGP IPv6 Prefix Advertisement over IPv4 BGP Peers Feature

You enable the BGP IPv6 Prefix Advertisement over IPv4 BGP Peers feature at the per peer-group level. Configuring this feature via the CLI consists of the following basic procedures:

- Configuring the IPv6 address family
- Configuring the peer group
- Applying the configuration

The following example shows how to enable BGP IPv6 Prefix Advertisement over IPv4 BGP Peers:

```
device(config-vrf-bgp-pg-ipv6u)# do show running-config vrf vr1

vrf vr1
  router-id 192.0.2.1
  member ve 2-9
  router bgp
    local-as 100
    router-id 192.0.2.1

    address-family ipv4 unicast
      activate
    !
    address-family ipv6 unicast
      activate
```

```
    !

    peer-group pgV6OverV4
      remote-as 200
      address-family ipv4 unicast
        activate
      !
      address-family ipv6 unicast
        export-policy mpbgpSetNH
        activate
      !
      neighbor 192.0.2.2
    !
  !
!
device(config-vrf-bgp-pg-ipv6u)#
```

Verification and Monitoring

For the BGP IPv6 Prefix Advertisement over IPv4 BGP Peers feature, the following sections describe the following tasks:

- Check that both peers have negotiated IPv6 capability
- Display IPv6 routes learned through the IPv4 BGP session
- Confirm that the BGP session carries both address families

Verify BGP Capability Negotiation

To check that both peers have negotiated IPv6 capability, enter the following command:

show bgp vrf *vrf-name* **neighbor all**

For example:

```
device# show bgp vrf vr1 neighbor all

Peer: 192.0.2.3, Remote Port: 179, Peer-AS: 200
 Localhost: 192.0.2.4, Local Port: 49882, Local-AS: 100, Local-AS-Forced: -
 Peer Router ID: 192.0.2.2, Local Router ID: 192.0.2.1, VRF: vr1
 Route Reflector Client: No, Cluster-ID: -
 Fast External Failover: false
 State: Established, Uptime: 5m17s, Dynamic Peer: false, Peer Group: pgV6OverV4

 Last Notification Time  :
 Last Connection Reset Reason:
 Send Community: No, Send Extended Community: No

  Capabilities       Negotiated   Advertised   Received
  ============       ==========   ==========   ========
  Route-Refresh      true         true         true
  AS-4Byte           true         true         true
  Cisco-RR           true         true         true
  MPBGP              true         true         true

  MPBGP:
  ======
   Advertised: IPv4Ucast,IPv6Ucast
```

```
   Received : IPv4Ucast,IPv6Ucast
   Negotiated: IPv4Ucast,IPv6Ucast

  Timers               Interval(Sec)   Method        Remaining(Sec)
  ======               =============   ======        ==============
  Connect Retry        31              Default       -
  Hold Time            180             Negotiated    163
  Keepalive Timer      60              Default       43
  Start Timer          5               System        -
  Update Interval      0               Default       -
device#
```

In the command output, verify that the MP-BGP section shows the following data:

- Advertised: IPv4Ucast,IPv6Ucast
- Received: IPv4Ucast,IPv6Ucast
- Negotiated: IPv4Ucast,IPv6Ucast

Monitor IPv6 Routes

To display the IPv6 routes learned through your IPv4 BGP session for the BGP IPv6 Prefix Advertisement over IPv4 BGP Peers feature, enter the following command:

show bgp vrf *vrf-name* **routes ipv6-unicast**

For example:

```
device# show bgp vrf default-vrf routes ipv6-unicast

BGP routing table information for VRF default-vrf
Status: *-valid, >-best, S-stale
Path type: i-internal, e-external, l-local, r-redist, m-multipath, a-additional,
           p-primary, s-secondary
Origin codes: I - IGP, E - EGP, ? - incomplete

IPv6 Routes
-----------
Total number of routes: 1

Flags        Prefix                 Nexthop                Peer
============ ====================== ====================== =========
             2001:db8:0:2::34/128   ::ffff:192.0.2.2       192.0.2.2
             2001:db8:0:2::34/128   ::ffff:192.0.2.10      192.0.2.10
             2001:db8:0:2::34/128   ::ffff:192.0.2.4       192.0.2.4
             2001:db8:0:2::34/128   ::ffff:192.0.2.6       192.0.2.6
             2001:db8:0:2::34/128   ::ffff:192.0.2.8       192.0.2.8
             2001:db8:0:2::34/128   ::ffff:192.0.2.0       192.0.2.0
device#
```

This output shows the following details:

- IPv6 prefixes (such as 2001:db8:0:2::34/128)
- Next-hop addresses (such as ::ffff:192.0.2.2)
- Peer information (such as 192.0.2.2)

Verify Session Status

To confirm that your BGP session carries both address families for the BGP IPv6 Prefix Advertisement over IPv4 BGP Peers feature, enter the following command:

show bgp vrf *vrf-name* **summary ipv6-unicast**

For example:

```
device# show bgp vrf vr1 summary ipv6-unicast

VRF Name : vr1
Local AS: 6500, Router ID: 2001:db8:25:25:25:25
Confederation Identifier: not configured
Confederation Peers: not configured
Graceful Restart: Disabled

IPv6 Unicast Summary Information
--------------------------------
Graceful Restart: Disabled
Prefix Independent Convergence: Disabled
Nexthop address resolution - Recursion: Disabled, Default-Route: Disabled
Number of Routes :0 (Paths: 0)
Number of Peer(s) Configured : 125, Up : 125

D: Dynamically created based on a listen range command, Dynamically created neighbors: 0
C: Route Reflector Client

Peer IP   Peer AS  Local IP             State       Health  UpTime     RIB-IN  RIB-OUT
Flags Peer Group
=======   ======== ==================== =========== ======= ========== ======= ========
===== ==========
192.0.2.2 6501     2001:db8:1:1:84::1   Established -       1h19m12s   0       0
-     pg_ipv6_1
192.0.2.3 6501     2001:db8:1:1:99::1   Established -       1h19m12s   0       0
-     pg_ipv6_1
192.0.2.4 6501     2001:db8:1:1:ad::1   Established -       1h19m11s   0       0
-     pg_ipv6_1
192.0.2.5 6501     2001:db8:1:1:d9::1   Established -       1h19m10s   0       0
-     pg_ipv6_1
192.0.2.6 6501     2001:db8:1:1:df::1   Established -       1h19m10s   0       0
-     pg_ipv6_1
192.0.2.7 6501     2001:db8:1:1:b7::1   Established -       1h19m11s   0       0
-     pg_ipv6_1
192.0.2.8 6501     2001:db8:1:1:ed::1   Established -       1h19m10s   0       0
-     pg_ipv6_1
192.0.2.9 6501     2001:db8:1:1:97::1   Established -       1h19m12s   0       0
-     pg_ipv6_1
192.0.2.10 6501    2001:db8:1:1:dc::1   Established -       1h19m10s   0       0
-     pg_ipv6_1
192.0.2.11 6501    2001:db8:1:1:eb::1   Established -       1h19m10s   0       0
-     pg_ipv6_1
192.0.2.12 6501    2001:db8:1:1:7f::1   Established -       1h19m13s   0       0
-     pg_ipv6_1
192.0.2.13 6501    2001:db8:1:1:a9::1   Established -       1h19m11s   0       0
-     pg_ipv6_1
192.0.2.14 6501    2001:db8:1:1:ac::1   Established -       1h19m11s   0       0
-     pg_ipv6_1
192.0.2.15 6501    2001:db8:1:1:d7::1   Established -       1h19m10s   0       0
-     pg_ipv6_1
192.0.2.16 6501    2001:db8:1:1:c8::1   Established -       1h19m11s   0       0
-     pg_ipv6_1
192.0.2.17 6501    2001:db8:1:1:d6::1   Established -       1h19m10s   0       0
```

```
-     pg_ipv6_1
192.0.2.18 6501    2001:db8:1:1:a5::1   Established -       1h19m12s   0       0
-     pg_ipv6_1
device#
```

This command shows the following details:

- Session state information
- Number of IPv6 prefixes received (RIB-IN) and advertised (RIB-OUT)
- Session uptime and statistics

BGP Route Aggregation

This document describes how to aggregate routes from a range of networks into a single network prefix for a specific BGP address family in Extreme OS ONE. BGP Route Aggregation (RFC 4271) is a technique that combines multiple specific IP addresses into a single, broader route advertisement. Instead of announcing numerous individual routes across the internet, a device sends out one "supernet" prefix to reduce the sizes of routing information bases (RIBs).

Overview of BGP Route Aggregation

By default, a device advertises individual routes for all networks. To minimize the size of the BGP routing information base (RIB), you can aggregate the routes in a range of networks into one IPv4 or IPv6 prefix.

This feature defines an aggregate prefix under the current BGP address family. This feature creates the aggregate only when at least one contributing more-specific route exists in the BGP RIB (also called a routing table).

The feature originates the aggregate locally with the next hop set to **`self`**. The aggregate participates in BGP best path selection and, if selected, advertises it to eligible peers.

The aggregate route does not inherit the attributes of any particular contributing route. When the aggregate route is formed, its attributes are determined from the contributing routes according to the configured aggregation options, including whether **`as-set`** is configured. The **`attribute-policy`** command explicitly sets the ORIGIN, MED, COMMUNITY, EXTENDED COMMUNITY, LARGE COMMUNITY, and LOCAL_PREFERENCE attributes for the aggregate and applies them before advertisement. For details, see attribute-policy.

The system withdraws the aggregate when the last contributing specific route disappears. The system makes any previously suppressed specific routes eligible for advertisement again if no other suppression rule applies.

If the **`attribute-policy`** command attempts unsupported SET actions (such as next-hop or AS path attribute modifications), the system logs a configuration error.

BGP aggregate attributes: When the `as-set` option is not configured, the aggregated route includes the `ATOMIC_AGGREGATE` attribute, which indicates that AS_PATH information was lost during aggregation. The `AGGREGATOR` attribute identifies the router that created the aggregated route.

Benefits

The BGP Route Aggregation feature has the following primary benefits.

Reduced Routing Information Base (RIB) Size

The BGP Route Aggregation feature decreases the device RIB size in the following ways:

- Decreased memory usage: By compressing multiple specific IP subnets into one aggregate block, the device must store fewer prefixes in the RIB and in the Forwarding Information Base (FIB).
- Scalability: The feature prevents the global BGP routing table from excessive growth, which is an important requirement for the sustainability of the internet.

Lower Device Workload

The BGP Route Aggregation feature decreases the workload of a device in the following ways:

- Faster route lookups: Smaller RIBs means that a device can process packets and look up paths much faster.
- Reduced CPU and bandwidth: This feature reduces the number of BGP update messages that the device generates and transmits to peers. This conserves both bandwidth and router CPU usage during network convergence.

Enhanced Network Stability (Route Flap Damping)

The BGP Route Aggregation feature increases network stability in the following ways:

- Isolation of changes: If a specific, more granular prefix flaps (frequently goes up and down) within the aggregate, this feature hides this instability from the wider network. Upstream and global routers do not need to recalculate paths or process constant updates for minor changes.
- Resilience: If one specific route within the aggregated block drops, the broader summarized path remains valid, which avoids localized or total outages for the overall block.

Simplified Administration

The BGP Route Aggregation feature simplifies network administration in the following ways:

- Clean network management: You can manage one large, summarized route much more easily than managing hundreds of scattered, individual subnet announcements.

- Security: You can filter out specific, more granular prefixes from leaving their Autonomous System (AS), which reduces the risk of unauthorized external access to internal network details.

Configure BGP Route Aggregation

You can aggregate BGP4 routes into one network prefix, for example, 2001:db8::/24. The device advertises this single prefix to BGP4 neighbors instead of the individual BGP routes in the routing BGP routing information base (RIB) that fall within the specified range. Configuring this feature via the CLI consists of two basic procedures:

- Configuring BGP aggregate route policies
- Configuring a BGP aggregate route

Configure BGP Route Aggregation Policies

By default, when `summary-only` and route policies are not configured, the device advertises the aggregate route and the contributing more-specific routes. You can create three types of policies to perform the following actions, which you attach to aggregate routes that you create:

- Advertisement policy: Contributor-eligibility gate that determines which more-specific routes are eligible to contribute to the aggregate. Only routes matching the policy (Permit) become contributors; the aggregate originates only if ≥1 matching route exists. Routes that fail the policy are excluded from AS_PATH and attribute calculations.
- Suppression policy: Suppress from advertisement a set of more-specific routes that fall under the aggregate route for a specific BGP address family
- Attribute policy: Set or override the attributes of an aggregate route for a specific BGP address family

The following example shows how to configure a route policy that suppresses the advertisement of a set of more-specific routes that fall under the aggregate route for a specific BGP address family:

```
device# show running-config route-policy policy my_aggreg_suppress_policy_1

route-policy
  policy suppress-24-subnet-prefixes
    statement 1
      conditions
        match-prefix-set subnet-24-prefixes any
      !
      actions
        policy-result permit
      !
    !
  !
!
device# show running-config route-policy prefix-set subnet-24-prefixes
route-policy
  prefix-set subnet-24-prefixes
    prefix 198.51.100.0/24 mask-range exact
```

```
  !
!
device#
```

For details about configuring route policies, see Layer 3 Policy Based Routing on page 93.

Configure the BGP Aggregate Route

Extreme OS ONE does not use BGP Route Aggregation by default. You must configure it explicitly at the address family level. Follow this procedure to configure BGP Route Aggregation.

For details about the commands in this section, see the *Extreme OS ONE Switching and Routing Command Reference Guide*.

1. Access global configuration mode.

   ```
   device# configure terminal
   ```

2. Specify the VRF name and enter VRF configuration mode. You can specify the default VRF (named default-vrf) or a user-created VRF.

   ```
   device(config)# vrf violet
   ```

3. Configure an address family routing table for the BGP routing process within the VRF instance and enter BGP address family configuration mode. You must specify **`ipv4 unicast`** or **`ipv6 unicast`** for the address family.

   ```
   device(config-vrf-bgp)# address-family ipv4 unicast
   ```

4. Aggregate the routes from a range of networks into a single network prefix within the current unicast address family and enter VRF BGP address family route aggregation configuration mode.

   Note
   You cannot configure overlapping aggregate prefixes within the same BGP address family.

   ```
   device(config-vrf-bgp-ipv4u)# aggregate-address 198.51.100.0/24
   ```

   - For an IPv4 address, use a unicast address in CIDR notation (*A.B.C.D*/*M*), where *M* is the prefix length in the range of 0–32 for IPv4 addresses.
   - For an IPv6 address, use a unicast address in CIDR notation (*X:X::X:X*/*M*), where *M* is the prefix length in the range of 0–128 for IPv6 addresses.

5. (Optional) Include the Autonomous System (AS) set of contributing routes in the AS path of the aggregate route. This generates and advertises the aggregate route while retaining the AS path information from all of the specific, more granular routes that it covers. This prevents the loss of AS path loop detection information that occurs when the system summarizes specific routes into an aggregate route.

   ```
   device(config-vrf-bgp-ipv4u-aggr)# as-set
   ```

6. (Optional) Enable the device to suppress (not advertise) all more-specific routes that the current aggregate route covers.

```
device(config-vrf-bgp-ipv4u-aggr)# summary-only
```

Note

The **summary-only** command is less selective than the **suppress-policy** command. The **summary-only** command suppresses all of the specific component routes making up the aggregate route, while the **suppress-policy** command uses a route map to conditionally suppress only specific prefixes.

7. Apply a route policy to determine which more-specific routes are eligible to contribute to the aggregate route.

   Only routes within the aggregate range that are permitted by the policy are admitted as contributors. The aggregate route is originated only when at least one permitted contributing route exists.

```
device(config-vrf-bgp-ipv4u-aggr)# advertise-policy my_aggreg_advertise_policy_1
```

8. Apply a route policy that suppresses from advertisement a set of more-specific routes that fall under the aggregate route. Suppression affects only outbound advertisements, not local BGP routing information base (RIB) installation. The system suppresses routes that match this policy even when you did not configure the **summary-only** command.

```
device(config-vrf-bgp-ipv4u-aggr)# suppress-policy my_aggreg_suppress_policy_2
```

9. (Optional) Apply a route policy to set or override the attributes of the aggregate route before route advertisement. The route policy supports a SET of the ORIGIN, multi-exit discriminator (MED), COMMUNITY, EXTENDED_COMMUNITY, LARGE COMMUNITY, and LOCAL_PREFERENCE attributes. The attribute policy cannot modify the NEXT_HOP or AS_PATH attributes of an aggregate route. If the policy contains set actions for these attributes, the system ignores the set actions. Attributes that the policy does not set will retain their default values: ORIGIN INCOMPLETE, empty AS_PATH (unless you have configured the **as-set** command), and next hop set to **self**.

```
device(config-vrf-bgp-ipv4u-aggr)# attribute-policy my_aggreg_attrib_policy_1
```

10. Exit VRF BGP address family route aggregation configuration mode.

```
device(config-vrf-bgp-ipv4u-aggr)# exit
```

11. Activate the routing table for the current BGP address family.

```
device(config-vrf-bgp-ipv4)# activate
```

12. (Optional) Verify the configuration.

```
device(config-vrf-bgp-ipv4)# do show running-config vrf violet

vrf violet
    router bgp
        address-family ipv4 unicast
        aggregate-address 198.51.100.0/24
            as-set
            summary-only
            advertise-policy my_aggreg_advertise_policy_1
            suppress-policy my_aggreg_suppress_policy_1
```

```
            attribute-policy my_aggreg_attrib_policy_1
            !
            activate
        !
    !
!
device(config-vrf-bgp-ipv4)#
```

The following example configures a routing process under a VRF instance named violet. This example aggregates routes from a range of networks into the single network prefix 198.51.100.0/24 within the IPv4 unicast address family:

```
device# configure terminal
device(config)# vrf violet
device(config-vrf-violet)# router bgp
device(config-vrf-bgp)# address-family ipv4 unicast
device(config-vrf-bgp-ipv4u)# aggregate-address 198.51.100.0/24
device(config-vrf-bgp-ipv4u-aggr)# advertise-policy my_aggreg_advertise_policy_1
device(config-vrf-bgp-ipv4u-aggr)# as-set
device(config-vrf-bgp-ipv4u-aggr)# attribute-policy my_aggreg_attrib_policy_1
device(config-vrf-bgp-ipv4u-aggr)# summary-only
device(config-vrf-bgp-ipv4u-aggr)# exit
device(config-vrf-bgp-ipv4u)# activate
device(config-vrf-bgp-ipv4u)#
```

The following example displays the configuration of a VRF instance named violet that is running currently on the device. In this example, the instance configures a routing process with two IPv4 aggregate routes and an IPv6 aggregate route:

```
device# show running-config vrf violet

vrf violet
  router bgp
    address-family ipv4 unicast
      aggregate-address 198.51.100.0/24
        as-set
        summary-only
        advertise-policy my_aggreg_advertise_policy_1
        attribute-policy my_aggreg_attrib_policy_1
      !
      aggregate-address 203.0.113.0/24
        as-set
        summary-only
        advertise-policy my_aggreg_advertise_policy_1
        attribute-policy my_aggreg_attrib_policy_1
      !
      activate
    !
    address-family ipv6 unicast
      aggregate-address 2001:db8:100::/64
        as-set
        suppress-policy my_aggreg_suppress_policy_2
        attribute-policy my_aggreg_attrib_policy_2
      !
      activate
    !
  !
!
device#
```

The following example configures both an advertise policy and a suppress policy for the same aggregate route. The advertise policy is evaluated first and determines which more-specific routes are admitted as contributors to the aggregate. The suppress

policy is then evaluated on the admitted contributors and determines which of those routes are suppressed from advertisement. Contributing routes that do not match the suppress policy remain eligible for advertisement, subject to other configured policies.

```
device# configure terminal
device(config)# vrf violet
device(config-vrf-violet)# router bgp
device(config-vrf-bgp)# address-family ipv4 unicast
device(config-vrf-bgp-ipv4)# aggregate-address 198.51.100.0/24
device(config-vrf-bgp-ipv4u-aggr)# advertise-policy my_aggreg_advertise_policy_1
device(config-vrf-bgp-ipv4u-aggr)# suppress-policy my_aggreg_suppress_policy_1
device(config-vrf-bgp-ipv4u-aggr)# activate
```

The following example displays the running configuration for an aggregate route with both an advertise policy and a suppress policy configured. The advertise policy is evaluated first to determine the contributing routes, and the suppress policy is then evaluated on those contributors to determine which routes are suppressed from advertisement.

```
device(config-vrf-bgp-ipv4)# do show running-config vrf violet

vrf violet
    router bgp
        address-family ipv4 unicast
        aggregate-address 198.51.100.0/24
            advertise-policy my_aggreg_advertise_policy_1
            suppress-policy my_aggreg_suppress_policy_1
            activate
        !
    !
!
device(config-vrf-bgp-ipv4)#
```

Verification and Monitoring

The following sections describe how to display BGP Route Aggregation configuration and operational information on the device:

- Display the configuration of all BGP aggregate prefixes or a specified aggregate prefix for a specific VRF and address family.
- Display the aggregate routes to which a prefix contributes within a specific VRF and address family.
- Display the runtime state of all BGP aggregate prefixes or a specified BGP aggregate prefix for a specific VRF and address family.

Verify BGP Route Aggregation Configuration

To display the configurations of all BGP aggregate prefixes or a specified aggregate prefix for a specific VRF and address family, enter the following command:

show bgp vrf { **default-vrf** | *vrf-name* } **aggregate-config** { **ipv4-unicast** [*ipv4-addr*] | **ipv6-unicast** [*ipv6-addr*] }

For more information about the **show bgp vrf aggregate-config** command, see the *Extreme OS ONE Switching and Routing Command Reference Guide*.

The following example displays all aggregate route configurations for the IPv4 address family and unicast subaddress family within a VRF named violet. In this example, the system uses aggregate routes for two IPv4 prefixes:

```
device# show bgp vrf violet aggregate-config ipv4-unicast

BGP Aggregate-Address Configuration
VRF: violet | AFI-SAFI: IPV4_UNICAST
=============================================================================================================
Aggregate-prefix  AS-SET  Summary-Only  Suppress-Policy  Advertise-Policy     Attribute-Policy
================  ======  ============  ===============  ================     ================
198.51.100.0/24   Yes     No            -                my_aggr_adve_poli_1  my_aggr_attr_poli_1
203.0.113.0/24    Yes     No            -                my_aggr_adve_poli_2  my_aggr_attr_poli_2

Total: 2 aggregate(s)
device#
```

The following example displays the aggregate route configuration for the IPv4 address family, unicast subaddress family, and the 198.51.100.0/24 prefix within a VRF named violet:

```
device# show bgp vrf violet aggregate-config ipv4-unicast 198.51.100.0/24

BGP Aggregate-Address Configuration
VRF: violet | AFI-SAFI: IPV4_UNICAST
=============================================================================================================
Aggregate-prefix  AS-SET  Summary-Only  Suppress-Policy      Advertise-Policy  Attribute-Policy
================  ======  ============  ===============      ================  ================
198.51.100.0/24   Yes     No            my_aggr_supp_poli_1  -                 my_aggr_attr_poli_1

Total: 1 aggregate(s)
device#
```

Verify BGP Route Aggregation Information

To display the aggregate routes to which a prefix contributes within a specific VRF and address family, enter the following command:

show bgp vrf { **default-vrf** | *vrf-name* } **aggregate-info** { **ipv4-unicast** [*ipv4-addr*] | **ipv6-unicast** [*ipv6-addr*] }

For more information about the **show bgp vrf aggregate-info** command, see the *Extreme OS ONE Switching and Routing Command Reference Guide*.

The following example displays all aggregate routes to which the 198.51.100.128/26 prefix contributes in the IPv4 address family and unicast subaddress family within a VRF named violet:

```
device# show bgp vrf violet aggregate-info ipv4-unicast 198.51.100.128/26

(VRF: violet  AFI: IPV4_UNICAST)

Route 198.51.100.128/26 contributes aggregates:
```

```
  198.51.100.0/24
device#
```

Verify BGP Route Aggregation State

To display the runtime state of all BGP aggregate prefixes or a specified BGP aggregate prefix for a specific VRF and address family, enter the following command:

show bgp vrf { **default-vrf** | *vrf-name* } **aggregate-state** { **ipv4-unicast** [*ipv4-addr*] | **ipv6-unicast** [*ipv6-addr*] } [**detail**]

The **detail** keyword displays contributor prefixes and provides deeper attribute context. You can use this information to troubleshoot which routes form an aggregate route and why the system advertised or suppressed it.

For more information about the **show bgp vrf aggregate-state** command, see the *Extreme OS ONE Switching and Routing Command Reference Guide*.

The following example displays the operational state of all BGP aggregate routes for the IPv4 address family and unicast subaddress family within a VRF named violet:

```
device# show bgp vrf violet aggregate-state ipv4-unicast

BGP aggregate route state for VRF: violet  AFI/SAFI: IPV4_UNICAST

Total number of routes: 1
Prefix             Status             Age       Contributor Count
================== ================== ========= =================
198.51.100.0/24    Active|Advertised  5h45m47s  3
device#
```

The following example displays details about the operational state of all BGP aggregate routes for the IPv6 address family and unicast subaddress family within a VRF named violet:

```
device# show bgp vrf violet aggregate-state ipv6-unicast detail

BGP aggregate route state for VRF: violet  AFI/SAFI: IPV6_UNICAST

Total number of routes: 2
Aggregate Prefix: 2001:db8:1000::/48  Age: -  Status: Inactive
  Origin: -  Local-Pref: -  MED: -
  As-Path: -
  Communities: -
  Ext-Communities: -
  Route-State: 0x0  Age: -
  Contributor Count: 0

Aggregate Prefix: 2001:db8:2000::/48  Age: 5h45m10s  Status: Active|Advertised
  Origin: IGP  Local-Pref: -  MED: -
  As-Path: {65003, 1234}
  Communities: 100:10, 200:20
  Ext-Communities: AS4:RT:1001:1001, AS4:RT:2002:2002
  Route-State: 0x500004a  Age: 5h45m10s
  Contributor Count: 2
  Contributing routes:
    2001:db8:2000:10::/64 (suppressed)
    2001:db8:2000:20::/64 (suppressed)
device#
```

BGP Monitoring with Bidirectional Forwarding Detection (BFD)

BFD support for BGP provides fast detection of link failures between BGP peers. By default, BFD for BGP is disabled. When enabled, BFD notifies BGP of any faults, allowing for quicker convergence. You use the **enable-bfd** command to enable BFD for a specified BGP neighbor or all neighbors in the specified BGP peer group within a VRF instance.

This feature has the following key functionalities:

- Supports single hop and multihop iBGP and eBGP sessions with IPv4 or IPv6 neighbors
- Works across default and nondefault VRF instances
- Uses behavior consistent for iBGP and eBGP, regardless of session type or neighbor IP version

Configuration Considerations

- Registration is global across all VRFs (BFD sessions require configuration on both neighbors).
- Each neighbor can have custom settings for transmit interval, receive interval, and detection multiplier.
- Peer group or global settings are inherited if not configured.
- Enabling BFD is controlled at the peer group level. All session parameters are inherited from the peer group level.
- BFD is not enabled by default and requires explicit configuration.
- The default timers are 300ms (Rx and Tx) with a multiplier of 3.
- BGP monitoring with BFD is supported for IPv4 and IPv6 peers, including dynamic peers.

Configuring BGP Monitoring with BFD

The following example configures a BGP peer group named group1 under a VRF instance named violet. The **enable-bfd** command is used in BGP peer group configuration mode and therefore enables BFD for all neighbors in the peer group:

```
device# configure terminal
device(config)# vrf violet
device(config-vrf-violet)# router bgp
device(config-vrf-bgp)# peer-group group1
device(config-vrf-bgp-pg)# enable-bfd profile profile1
device(config-vrf-bgp-pg)#
```

The following example displays the configuration of a VRF instance named violet that is running currently on the device. This example configures a BGP peer group named group1 and enables BFD for all neighbors in the peer group:

```
device # show running-config vrf violet

vrf violet
!
```

```
  router bgp
    peer-group group1
      enable-bfd profile profile1
!
device#
```

The following example configures a BGP peer group named group1 under a VRF instance named violet. The **enable-bfd** command enables BFD only for the 192.0.2.0 neighbor within the peer group:

```
device# configure terminal
device(config)# vrf violet
device(config-vrf-violet)# router bgp
device(config-vrf-bgp)# peer-group group1
device(config-vrf-bgp-pg)# neighbor 192.0.2.0
device(config-vrf-bgp-pg-neighbor)# enable-bfd profile profile1
device(config-vrf-bgp-pg-neighbor)#
```

BGP Protocol Event Monitoring and Notification

The BGP Protocol Event Monitoring and Notification feature automates alerts about issues such as policy misconfigurations and route flaps rather than requiring manual CLI checks, which improves efficiency and network stability. Key events monitored include BGP IPv4 and IPv6 connection state transitions (one of the pre-Established to Established state) and Established to one of the pre-Established state).

Supported Functionalities

You can configure BGP Protocol Event Monitoring and Notification to send the following types of data:

- gNMI notifications as part of BGP Finite State Machine (FSM) events and BGP session UP/DOWN events
- RAS trace logs as part of BGP UP/DOWN events
- SNMP traps as part of BGP FSM events for Enterprise MIB and Standard MIB

BGP Enterprise and Standard MIB Notifications

BGP reports significant events to the message bus when a BGP session changes state to Established or experiences backward transitions. These BGP traps contain session information within their payload/Varbind. The BGP Enterprise and Standard MIB define specific trap OID and Varbind lists for this purpose.

BGP standard MIB notifications are sent for the peers of IPv4 types. Table 7 lists the BGP standard MIB notifications and varbinds.

Table 7: BGP standard MIB notifications

Trap name and OID	Varbinds	Description
bgpEstablishedNotification 1.3.6.1.2.1.15.0.1	bgpPeerRemoteAddr bgpPeerLastError bgpPeerState	The bgpEstablishedNotification event is generated when the BGP FSM enters the established state.
bgpBackwardTransNotification 1.3.6.1.2.1.15.0.2	bgpPeerRemoteAddr, bgpPeerLastError, bgpPeerState	The bgpBackwardTransNotification event is generated when the BGP FSM moves from a higher numbered state to a lower numbered state.

The BGP enterprise MIB notifications are sent for the peers of IPv6 types. Table 8 lists the BGP enterprise MIB notifications and varbinds.

Table 8: BGP enterprise MIB notifications

Trap name and OID	Varbinds	Description
extremeBGP4V2EstablishedNotification 1.3.6.1.4.1.1916.1.51.0.1	extremeBgp4V2PeerState extremeBgp4V2PeerLocalPort extremeBgp4V2PeerRemotePort extremeBgp4V2PeerRemoteAddr	The extremeBGP4V2EstablishedNotification event is generated when the BGP FSM enters the established state.
extremeBGP4V2BackwardTransitionNotification 1.3.6.1.4.1.1916.1.51.0.2	extremeBgp4V2PeerState extremeBgp4V2PeerLocalPort extremeBgp4V2PeerRemotePort extremeBgp4V2PeerLastErrorCodeReceived, extremeBgp4V2PeerLastErrorSubCodeReceived, extremeBgp4V2PeerLastErrorReceivedText extremeBgp4V2PeerRemoteAddr	The extremeBGP4V2BackwardTransitionNotification event is generated when the BGP FSM moves from a higher numbered state to a lower numbered state.

Following is an example of a BGP SNMP trap message:

```
2025-02-17 09:49:36 <UNKNOWN> [UDP: [10.32.100.182]:53908->[10.37.32.26]:200]:
iso.3.6.1.2.1.1.3.0 = Timeticks: (25057400) 2 days, 21:36:14.00 iso.3.6.1.6.3.1.1.4.1.0 =
OID: iso.3.6.1.4.1.1916.1.51.0.2      iso.3.6.1.4.1.1916.1.51.1.2.1.13 = STRING: "Idle"
iso.3.6.1.4.1.1916.1.51.1.2.1.5 = STRING: "1000::1"     iso.3.6.1.4.1.1916.1.51.1.3.1.1
= Counter32: 6  iso.3.6.1.4.1.1916.1.51.1.3.1.2 = Counter32: 6
iso.3.6.1.4.1.1916.1.51.1.2.1.6 = Counter32: 179        iso.3.6.1.4.1.1916.1.51.1.2.1.9
```

```
= Counter32: 38292      iso.3.6.1.4.1.1916.1.51.1.3.1.4 = STRING: "CEASE
OTHER_CONFIG_CHANGE"
```

gNMI Notifications

BGP neighbor configuration or state changes are published to the gNMI path `network-instances/network-instance[name=*]/protocols/protocol[identifier=BGP][name=bgp]/bgp/neighbors/neighbor[neighbor-address=*]/`

```
+--rw network-instances
     +--rw network-instance* [name]

      . . .

         +--rw protocols
           |  +--rw protocol* [identifier name]

           |+--rw bgp

      . . .

           |        +--rw neighbors
           |        |  +--rw neighbor* [neighbor-address]
           |        |     +{}rw neighbor-address{-}    > ../config/neighbor-address
      . . .
           |        |     +--ro state                                  oc-inet:as-number
           |        |     |  +--ro session-state?
enumeration
           |        |     |  +--ro last-established?                              oc-
types:timeticks64
           |        |     |  +--ro established-transitions?                       oc-
yang:counter64
           |        |     |  +--ro supported-capabilities*
identityref
           |        |     |  +--ro messages
           |        |     |  |  +--ro sent
           |        |     |  |  |  +--ro UPDATE?                             uint64
           |        |     |  |  |  +--ro NOTIFICATION?                       uint64
           |        |     |  |  |  +--ro last-notification-time?             oc-
types:timeticks64
           |        |     |  |  |  +--ro last-notification-error-code?      identityref
           |        |     |  |  |  +--ro last-notification-error-subcode?   identityref
           |        |     |  |  +--ro received
           |        |     |  |     +--ro UPDATE?                             uint64
           |        |     |  |     +--ro NOTIFICATION?                       uint64
           |        |     |  |     +--ro last-notification-time?             oc-
types:timeticks64
           |        |     |  |     +--ro last-notification-error-code?      identityref
           |        |     |  |     +--ro last-notification-error-subcode?   identityref
```

RASlogs

The following are Session UP/Down RASlogs:

```
2025-01-16 11:04:36.7728 bgp[15]: {"Level":"info","LogID":9008,"Topic":2,"VRF":"default-
vrf","Neighbor":"192.x.x.x","Reason":"ADMIN-DOWN","Msg":"Session DOWN"}
2025-01-16 11:04:36.7729 bgp[15]: {"Level":"info","LogID":9008,"Topic":2,"VRF":"default-
vrf","Neighbor":"10.x.x.x","Reason":"ADMIN-DOWN","Msg":"Session DOWN"}
2025-01-16 11:06:29.1857 bgp[15]: {"Level":"info","LogID":9008,"Topic":2,"VRF":"default-
vrf","Neighbor":"10.x.x.x","Msg":"Session UP"}
2025-01-16 11:06:31.8469 bgp[15]: {"Level":"info","LogID":9008,"Topic":2,"VRF":"default-
vrf","Neighbor":"10.x.x.x","Msg":"Session UP"}
```

Configure BGP Address Families

BGP supports multiple address families for diverse network types and applications. Each address family specifies the type of IP address or data structure BGP will handle. The currently supported address families are:

1. IPv4 Unicast: BGP routes IPv4 addresses (the traditional and most commonly used address family)
2. IPv6 Unicast: BGP exchanges routes for IPv6 addresses
3. EVPN (Ethernet VPN): Extends BGP to carry Ethernet frames and IP routes as a modern service for multi-tenant Layer 2 and Layer 3 Ethernet services

Each BGP instance can support one or more address families (except EVPN, which is available only in the default instance). Address families operate independently within an instance with no correlation between instances.

Each address family must be activated individually before it becomes operational. Features can be enabled under specific address families.

Memory allocation occurs only when an address family is activated, optimizing memory utilization.

You can access the address-family IPv4 or IPv6 unicast configuration level from global configuration mode.

```
device# configure terminal
device(config)# vrf violet
device(config-vrf-violet)# router bgp
device(config-vrf-bgp)# address-family ipv4 unicast
device(config-vrf-bgp-ipv4u)# activate
device(config-vrf-bgp-ipv4u)# exit
device(config-vrf-bgp)# address-family ipv6 unicast
device(config-vrf-bgp-ipv6u)# activate
device(config-vrf-bgp-ipv6u)#
```

BGP4 Peer Groups

Peer groups simplify BGP configuration by grouping multiple peers with shared settings. Instead of configuring each peer individually, you define common settings once and apply them to multiple peers.

You can use the **`peer-group`** command to configure a BGP peer group within a Virtual Routing and Forwarding (VRF) instance and enter BGP peer group configuration mode. No default peer group exists. Each peer group must be manually configured.

You can configure BGP peers only as peer group members (not as standalone or independent peers). Each peer group requires the same capability negotiation and address family for all peers.

You can associate BGP neighbors with a BGP4 peer group. For details on syntax and parameters, see the *Extreme OS ONE Switching and Routing Command Reference Guide*.

Creating multiple peer groups increases the number of RIBOUTs, which impacts memory consumption.

How Peer Groups Work

You configure a peer group by following two main procedures:

1. Define a peer group with common parameters. For example, remote-as and an authentication password
2. Assign individual BGP peers to the group to eliminate the need for separate configurations

Key Benefits

Using BGP4 peer groups has the following main benefits:

- Simplified Maintenance: When you edit the peer group, changes are propagated across all associated peers.
- Optimization of Updates: Peers in a group share the same update to reduce the load by sending a single update to all members.
- Shared Configuration: Parameters such as timers, policies, and authentication can be shared across multiple neighbors.

Configure a BGP4 Peer Group

A BGP4 peer group is a collection of BGP neighbors with the same attributes, parameters, and address families.

The following sample output shows the configuration of a VRF named red that contains two peer groups:

```
device(config-vrf-bgp-pg-ipv4u)# do show running-config vrf red

vrf red
  member   ve 2,8192
  table-connections
    src-protocol connected dst-protocol bgp address-family ipv4
    src-protocol connected dst-protocol bgp address-family ipv6
  !
  router bgp
    local-as 4200000001
    router-id 192.0.2.171
    route-distinguisher 192.0.2.171:1
    as-notation as-plain

    address-family ipv4 unicast
      use-multiple-paths ibgp maximum-paths 8
      use-multiple-paths ebgp maximum-paths 8
    !

    address-family ipv6 unicast
      use-multiple-paths ibgp maximum-paths 8
      use-multiple-paths ebgp maximum-paths 8
    !

    instance-type evpn
```

```
      nve fs
        l3vni 8192
        address-family ipv4 unicast
          route-target export 101:101
          route-target import 101:101
        !
        address-family ipv6 unicast
          route-target export 101:101
          route-target import 101:101
        !
      !
    !

    peer-group pg1
      remote-as 651000
      neighbor 192.0.2.1
      enable-bfd profile profile_100_100_3
      update-source 192.0.2.10
      address-family ipv4 unicast
        nexthop-self
        activate
      !
    !

    peer-group pg2
      remote-as 651000
      neighbor 192.0.2.2
      enable-bfd profile profile_100_100_3
      update-source 2001:db8::ff
      address-family ipv4 unicast
        activate
      !
    !
  !
!
device#
```

BGP Four-Byte AS Number

BGP supports RFC 4893, which extends AS numbers to four bytes to provide a much larger range of 0 to 4,294,967,295. This feature is enabled by default and cannot be disabled.

Local-AS configurations support four-byte AS numbers, including two-byte AS numbers. Remote AS numbers always support four-byte ranges, although two-byte AS numbers can be configured.

AS Number Format

Four-byte AS numbers are represented in a two-part format:

- High-order 2 bytes (0 to 65,535)
- Low-order 2 bytes (0 to 65,535)

You can control the AS number format using the **as-notation** option at the instance level, with the following three available formats:

- as-dot (default): Displays 4-byte AS numbers in dot notation and 2-byte AS numbers in decimal
- as-dot-plus: Displays all AS numbers in dot notation
- as-plain: Displays all AS numbers in decimal

BGP supports interaction with both four-byte AS capable and two-byte AS only capable routers. If a route policy sets an AS number that exceeds the capability of the router, it is ignored.

BGP Multi-VRF

You use BGP Multi-VRF to set up the Border Gateway Protocol (BGP) to operate with multiple Virtual Routing and Forwarding (VRF) instances on a single router. This means that BGP can maintain separate routing tables for different network environments to enable each VRF to exchange routing information independently. This adds flexible and scalable network design, but also requires careful management to avoid potential issues.

The BGP multi-VRF feature has the following key functionalities:

- Multi-Instance Capability: BGP can advertise routes into multiple VRFs, ensuring isolated routing information exchange.
- Implementation: A single BGP process internally manages multiple VRF instances, with bifurcation within the process.
- Enabled by Default: This feature is always enabled and cannot be disabled.
- Single Process: One process serves all VRF instances.
- Per-VRF Instance: Each VRF has its own BGP instance, configurable using the **router bgp** command.
- Instance-Specific Configuration: Each VRF instance has its own routing configuration, including router ID, AS number, peers, and routing tables.
- Shared Routing Policies: Routing policies are not VRF-aware and are shared across all VRFs.
- Resource Sharing: A single memory space and CPU processing are shared among all instances, which can lead to the following issues:
  - Resource contention: One instance can consume significant resources, impacting others.
  - Single point of failure: A single instance failure can affect the entire BGP setup.

BGP Router-ID

The BGP router-ID is a unique 4-octet unsigned nonzero integer that identifies the sender of BGP messages within an Autonomous System (AS). It's typically set to an IP address assigned to the BGP speaker and remains the same across all local interfaces and BGP peers.

Use the **router-id <ipv4-address>** command to set the router ID for BGP. This AS number is called the BGP router ID, and it is one per router BGP instance. For details on syntax and parameters, see the *Extreme OS ONE Switching and Routing Command Reference Guide*.

- Dual Router-IDs: Two router-IDs exist in the configuration: One at the VRF level and another at the BGP-global level.
- Default Behavior: If the BGP-global level router-ID is not configured, it inherits the VRF-level router-ID.
- Configuration Changes: When the BGP-global level router-ID is deleted, it falls back to the VRF-level router-ID (if configured). If no VRF-level router-ID exists, the router-ID path is removed from the SDB.

A BGP router-id is set only if explicitly configured and if not configured, the BGP router-id remains unset. Configuration of the VRF-level router-ID is not supported.

Configure BGP4 Route Reflection

BGP Route Reflection (RR) provides efficient routing information exchange within an Autonomous System (AS) without requiring direct peer-to-peer relationships between all routers. This enhances scalability in large networks.

In traditional IBGP networks, each router must establish a direct session with every other router, which causes complexity and management issues in large environments. A Route Reflector acts as a central point for routing information, which simplifies IBGP design. Routers establish BGP sessions with one or more route reflectors, which propagate routes to the rest of the network.

When a route reflector receives a route from a client, it reflects the route to other clients, eliminating the need for a full mesh. The route reflector won't send the route back to the client that originated it.

The following are the key components of BGP route reflection:

- Route Reflector (RR): Redistributes BGP routes to other IBGP peers
- Client Routers: Form IBGP sessions with a route reflector
- Non-Client Routers: Participate in BGP without being part of a route reflector client group

The default Cluster ID is the Router ID, which can be overridden. The originator ID is always the router ID of the peer and cannot be overridden.

Feature and peer configuration require manual setup.

BGP4 Route Reflection does not support Optimal Route Reflection (RFC 9107).

Configure a Cluster ID for a BGP4 Route Reflector

When you have multiple route reflectors, you change the cluster ID so that all route reflectors belong to the same cluster.

1. Access global configuration mode.

```
device# configure terminal
```

2. Access VRF BGP peer group configuration mode (config-vrf-bgp-pg) and configure BGP peer group (group1) under a VRF instance named violet. The group1 group is assigned a cluster ID (10) in integer format.

```
device# configure terminal
device(config)# vrf violet
device(config-vrf-violet)# router bgp
device(config-vrf-bgp)# peer-group group1
device(config-vrf-bgp-pg)# cluster-id 10
device(config-vrf-bgp-pg)# exit
```

Configure a BGP4 Route Reflector Client

You configure a BGP peer as a route-reflector client from the device that is the route reflector.

1. Access global configuration mode.

```
device# configure terminal
```

2. Access BGP configuration mode.

```
device(config)# router bgp
```

3. Access the VRF BGP peer group configuration (config-vrf-bgp-pg) mode. Specify the peer group to be the route reflector client.

```
device# configure terminal
device(config)# vrf violet
device(config-vrf-violet)# router bgp
device(config-vrf-bgp)# peer-group group1
device(config-vrf-bgp-pg)# route-reflector-client
device(config-vrf-bgp-pg)#
```

Configure BGP Prefix Independent Convergence

In large-scale networks with thousands of BGP peers exchanging millions of routes, many routes are reachable via multiple next hops. When network topology changes, BGP recalculates the best routes for affected prefixes, which can take significant time and cause data traffic to be black-holed. To achieve faster data plane convergence and reduce the impact of network topology changes on traffic forwarding, BGP Prefix Independent Convergence (PIC) is used.

BGP PIC is supported on the Extreme 8730 platform. BGP PIC is disabled by default.

Functional overview

BGP PIC reduces data plane convergence time in large-scale networks. When enabled, BGP installs both the best path and a secondary path to a destination for fast failover after a failure. You can enable or disable PIC at the VRF level for IPv4 or IPv6 unicast.

BGP PIC supports IPv4 and IPv6 address families. You enable BGP PIC at the global AFI-SAFI level.

BGP PIC is supported for all VRFs. It is not supported for Layer 3 VPN, EVPN, or IP over MPLS (IPoMPLS).

Benefits

BGP PIC has the following key benefits:

- Faster convergence time: BGP PIC provides faster data plane convergence in case of link or node failures in the core network or edge network, reducing the time it takes to restore traffic after a failure.
- Prefix-independent convergence: Convergence time is no longer proportional to the number of prefixes, ensuring faster recovery.
- Secondary path installation: With BGP PIC enabled, BGP also installs a secondary path to the destination to improve network resilience.
- Fast failover: When a failure is detected, BGP switches to the secondary path to reduce traffic loss.
- Best path selection: BGP selects the best path to a destination and downloads it to Unified Forwarding Table Manager (UFTM).
- Less traffic loss: By using a backup path, traffic loss is minimized until BGP updates the next hop.
- OptiScale: BGP PIC is compatible with OptiScale profiles.

BGP PIC Functional Scenarios

BGP PIC functionality varies based on the location and type of network failure.

BGP PIC provides fast convergence if the network fails. It uses two main scenarios.

BGP PIC Core Network Failure

Figure 9 illustrates a BGP PIC core network failure.

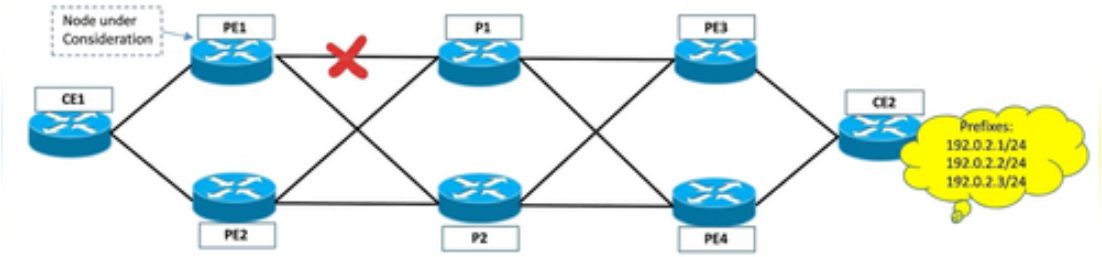


Figure 9: BGP PIC core node failure

In the scenario in Figure 9, PE1 has two paths to CE2 prefixes: one via PE3 and another via PE4. With BGP PIC enabled, PE1 selects a primary path (via PE3) and a secondary path (via PE4). The primary next hop PE3 is reachable via two paths: PE1-P1-PE3 and PE1-P2-PE3. When the link between PE1 and P1 goes down, you must use static routes

or recursive or alternative BGP routes. Because the BGP next hop did not change, recovery time is based on IGP convergence.

BGP PIC Edge Network Failure

Figure 10 illustrates a BGP PIC edge network failure.

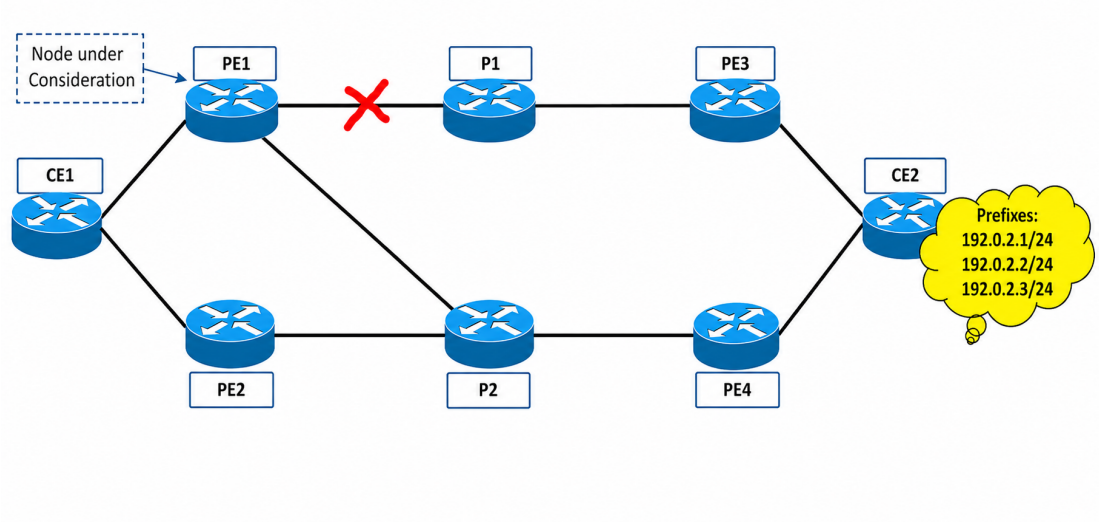


Figure 10: BGP PIC edge network failure

In the scenario in Figure 10, PE1 has two paths to CE2 prefixes: one via PE3 and another via PE4. With BGP PIC enabled, PE1 selects a primary path (via PE3) and a secondary path (via PE4). When the link between PE1 and P1 goes down, UFTM detects the IGP path failure and fails over to the secondary next hop PE4 reachable via IGP path P2. The prefix routes from CE2 will be updated to point to P2, and recovery time will be based on IGP convergence.

BGP PIC Edge Node Failure

Figure 11 illustrates a BGP PIC edge node failure and the behavior of the route tables.

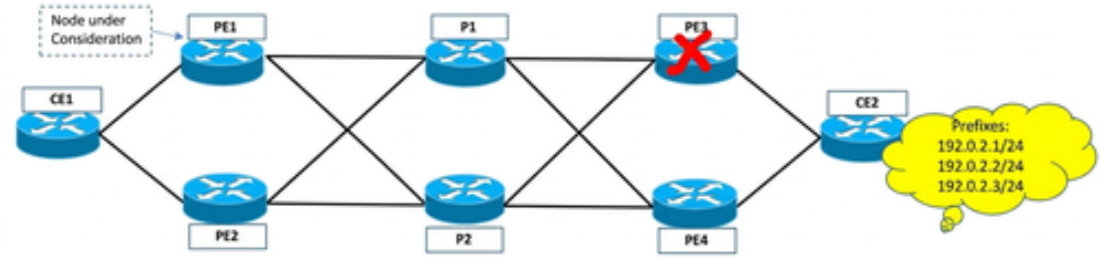


Figure 11: BGP PIC edge node failure

In the scenario in Figure 11, when PE3 fails or the BGP session between PE1 and PE3 fails, BGP notifies UFTM with a NextHopDown event. UFTM resolves the secondary BGP next hop PE4 to IGP next hop and updates the next hop ID to point to PE4. Traffic will failover to the PE4 next hop, ensuring fast convergence.

Key Benefits

BGP PIC provides fast convergence in case of network failures. Recovery time is based on IGP convergence, reducing the time it takes to restore traffic.

BGP PIC Configuration Considerations

BGP graceful restart must be configured for PIC high availability (HA) to work. As a best practice, use the **`profile route maximum-paths`** configuration command to reduce the default ECMP value from 64 to either 8 or 16.

BGP and BFD Session Down Event

BGP/BFD session down events will signal link failures, which cause BGP to withdraw routes rapidly for faster convergence.

BGP Session Down Event

When a BGP session goes down, the peer IP might not be unreachable. Therefore, BGP sends only a NextHopDown event to the Unified Forwarding Table Manager (UFTM) for a nex thop advertised by the peer if it is not advertised by another peer. Upon receiving the NextHopDown event, UFTM fails over to the secondary BGP next hop. Following are the BGP session down processing events:

1. BGP session down detection: BGP detects the session down event and checks if the peer IP is used as a next hop by any routes.
2. NextHopDown event: If the peer IP is used as a next hop and not advertised by any other peer, BGP sends a NextHopDown event to UFTM.
3. Failover to secondary next hop: UFTM fails over to the secondary BGP next hop.
4. Route selection algorithm: BGP reruns the route selection algorithm and downloads the new primary and secondary paths to UFTM.

BFD Session Down Event

When a BFD session goes down, it is confirmed that the peer is unreachable. BGP checks if the peer IP is used as a next hop by any routes and sends a NextHopDown event to UFTM if necessary. The following are the BFD session down processing event:

1. BFD session down detection: BFD detects the session down event and notifies BGP.
2. NextHopDown event: BGP checks if the peer IP is used as a next hop and sends a NextHopDown event to UFTM if necessary.
3. Failover to secondary next hop: UFTM fails over to the secondary BGP next hop.
4. Route selection algorithm: BGP reruns the route selection algorithm and downloads the new primary and secondary paths to UFTM.

Configure BGP Prefix-Independent Convergence

You can enable BGP Prefix-Independent Convergence to accelerate data path convergence under failover conditions.

- Enabling BGP PIC

  BGP PIC is disabled by default. To enable it, you can configure it for an address family under a VRF. When enabled, BGP selects primary and secondary paths and downloads them to Unified Forwarding Table Manager (UFTM).

- Disabling BGP PIC

  When BGP PIC is disabled, BGP runs the route selection algorithm and select the best path. No secondary path will be selected, and only the best path will be downloaded to the hardware.

For details on syntax and parameters, see the *Extreme OS ONE Switching and Routing Command Reference Guide*.

.

1. Enter global configuration mode.

```
device# configure terminal
```

2. Add VRF instance.

```
device# configure terminal
device(config)# vrf violet
```

3. Run the **router bgp** command.

```
device# configure terminal
device(config)# vrf violet
device(config-vrf-violet)# router bgp
device(config-vrf-bgp)#
```

4. Run the **address-family** command with IP4 or IPv6 unicast.

```
device# configure terminal
device(config)# vrf violet
device(config-vrf-violet)# router bgp
device(config-vrf-bgp)# address-family ipv4 unicast
device(config-vrf-bgp-ipv4u)#
```

5. Run the **prefix-independent-convergence** command and activate it.

```
device# configure terminal
device(config)# vrf violet
device(config-vrf-violet)# router bgp
device(config-vrf-bgp)# address-family ipv4 unicast
device(config-vrf-bgp-ipv4u)# prefix-independent-convergence
device(config-vrf-bgp-ipv4u)# activate
device(config-vrf-bgp-ipv4u)# exit
device(config-vrf-bgp)# address-family ipv6 unicast
device(config-vrf-bgp-ipv6u)# prefix-independent-convergence
device(config-vrf-bgp-ipv6u)# activate device(config-vrf-bgp-ipv6u)# exit
device(config-vrf-bgp)# address-family l2vpn evpn
device(config-vrf-bgp-l2vpn-evpn)# prefix-independent-convergence
device(config-vrf-bgp-l2vpn-evpn)# activate
device(config-vrf-bgp-l2vpn-evpn)#
```

BGP4 Graceful Restart

BGP Graceful Restart (GR) smoothly restarts BGP sessions to minimize disruptions to network routing and forwarding. During a restart, both BGP peers collaborate to maintain network stability to prevent route or topology changes. It minimizes network disruptions during BGP restarts or device reboots and ensures network stability and continuity. It comes with the following key functionalities:

- Negotiation: The GR capability is negotiated through BGP OPEN messages, and both peers must support GR for the feature to be activated.
- Restart Process: When a BGP session is lost, the GR helper router marks routes as "stale" but continues to forward packets using these routes for a predefined duration.
- Non-Stop Forwarding (NSF): Forwarding continues while the control plane converges, minimizing network disruptions.

When configuring restart timers with regular BGP timers, you should consider scale. Proper configuration of restart timers is crucial to ensure GR functionality.

Helper-Only mode is supported, where the router acts as a GR Helper by default if GR is not enabled under any AFI or SAFI.

Configure BGP Graceful Restart

Graceful Restart is not configured by default and must be explicitly configured at the instance level. You can enable or disable it at the AFI or SAFI level. Follow this procedure to configure BGP graceful restart.

Note
High availability (HA) requires Graceful Restart to be enabled.

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

```
device# configure terminal
```

2. Enter the config-vrf-bgp mode.

```
device# configure terminal
device(config)# vrf violet
device(config-vrf-violet)# router bgp
device(config-vrf-bgp)#
```

3. Configure graceful restart at the router BGP level.

```
device(config-vrf-bgp)# graceful-restart
device(config-vrf-bgp-gr)# restart-time 100
device(config-vrf-bgp-gr)# stale-route-time 500
device(config-vrf-bgp-gr)# helper-only
device(config-vrf-bgp-gr)#
```

4. Exit the config-vrf-bgp-gr mode.

```
device(config-vrf-bgp-gr)# exit
device(config-vrf-bgp)#
```

5. (Optional) Enter BGP peer group configuration mode.

```
device(config-vrf-bgp)# peer-group group1
device(config-vrf-bgp-pg)#
```

6. (Optional) Configure graceful restart at the router BGP peer group level.

```
device(config-vrf-bgp-pg)# graceful-restart
device(config-vrf-bgp-pg-gr)# restart-time 100
device(config-vrf-bgp-pg-gr)# stale-route-time 500
device(config-vrf-bgp-pg-gr)#
```

7. (Optional) Exit BGP peer group configuration mode.

```
device(config-vrf-bgp-pg-gr)# exit
device(config-vrf-bgp)#
```

8. Enter the **address-family ipv4 unicast** command to enter IPv4 address-family configuration mode.

```
device(config-vrf-bgp)# address-family ipv4 unicast
device(config-vrf-bgp-ipv4u)#
```

9. Enter the **graceful-restart** command to enable the graceful restart feature.

```
 device# configure terminal
device(config)# vrf violet
device(config-vrf-violet)# router bgp
device(config-vrf-bgp)# address-family ipv4 unicast
device(config-vrf-bgp-ipv4u)# graceful-restart
device(config-vrf-bgp-ipv4u)#
```

The following example enables the graceful restart mode for the IPv4 unicast address family:

```
device# configure terminal
device(config)# vrf violet
device(config-vrf-violet)# router bgp
device(config-vrf-bgp)# address-family ipv4 unicast
device(config-vrf-bgp-ipv4u)# graceful-restart
device(config-vrf-bgp-ipv4u)#
```

The following example displays the configuration of a VRF instance named violet that is running currently on the device. In this example, the routing process is configured to enable graceful restart (with a restart timer and a stale route time of 100 seconds and 500 seconds respectively) on its neighbors:

```
device# show running-config vrf violet

vrf violet
  router bgp
    address-family ipv4 unicast
      graceful-restart    !
    !
 !
device#
```

The following example shows CLI output where BGP routes are marked as stale:

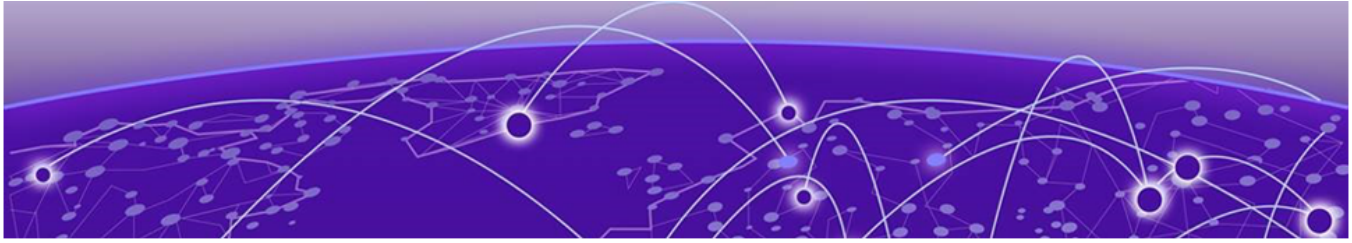
```
device# show bgp vrf default-vrf routes ipv4-unicast

BGP routing table information for VRF default-vrf
Status: *-valid, >-best, S-stale
Path type: i-internal, e-external, l-local, r-redist, m-multipath, a-additional,
```

```
           p-primary, s-secondary
Origin codes: I - IGP, E - EGP, ? - incomplete
IPv4 Routes
-----------
Total number of routes: 5
Flags        Prefix             Nexthop            Peer
============ ================== ================== ==================
*>Ie         203.0.113.3/32     198.51.100.1       198.51.100.1
*>?Si        198.51.100.0/24    198.51.100.1       198.51.100.1
*>?Si        192.0.2.0/24       198.51.100.1       198.51.100.1
*>ISi        192.0.2.1/32       198.51.100.1       198.51.100.1
*>Ie         203.0.113.0/24     198.51.100.1       198.51.100.1

device# show bgp vrf default-vrf routes ipv4-unicast 198.51.100.0/24

Prefix: 198.51.100.0/24, Nexthop: 198.51.100.1, Peer: 198.51.100.1, AS Path:
        Age: 42s, Status:VALID Path-Type: Best, Stale, Internal,
        Origin: INCOMPLETE
device#
```



BGP Ethernet VPN for IP Fabrics

The following topics describe how to configure BGP Ethernet VPN for IP fabrics.

BGP EVPN for IP Fabrics Features and Benefits

Ethernet VPN (EVPN) is a standards based solution for data center overlay and data center interconnect (DCI).

BGP EVPN is a control plane solution for Ethernet Layer 2 and Layer 3 VPN services and is ideal for data centers and large scale networks. It uses BGP to provide scalable and flexible network virtualization to extend Ethernet services across geographically dispersed sites.

RFC 7432 (BGP MPLS-Based Ethernet VPN) specifies multiprotocol extensions to BGP to exchange Layer 2 routes in an Ethernet VPN network. These extensions are useful in DCI scenarios.

A BGP EVPN-based IP fabric consists of BGP EVPN for VxLAN overlay networks and a broad set of Layer 2, Layer 3, and infrastructure features to provide seamless deployment, on-demand usage of forwarding entries in hardware, and network flooding minimization.

Key Features

BGP EVPN for IP Fabrics has the following key functionalities:

- BGP EVPN route propagation occurs only at the spine level for the control plane in Leaf and Spine architecture.
- Dynamic VxLAN Tunnel Endpoint (VTEP) discovery and VxLAN tunnel formation simplifies network setup and management.

- Route exchange supports Layer 2 MAC, MAC-IP (ARP or ND), and Layer 3 prefix routes, including Multi-VRF and symmetric or asymmetric routing.
- BGP EVPN for IP Fabrics supports logical VTEP support, static anycast gateway, ARP suppression, but no multi-homing support.

Key Benefits

The BGP EVPN for IP Fabrics feature has the following key benefits:

- Multi-tenancy support: Each tenant's traffic is isolated, crucial for cloud data centers and service provider networks.
- Integrated control plane: BGP EVPN combines Layer 2 and Layer 3 forwarding for efficient forwarding and unified management.
- Efficient Broadcast, Unknown-unicast, and Multicast (BUM) traffic handling: The control plane disseminates information to reduce flooding typically required in traditional Ethernet networks.

Build Modern Data Centers with VxLAN and BGP

The rapid growth of applications and storage in data centers brings new challenges in providing scalable and efficient connectivity between virtual machines (VMs). Traditional solutions such as VPLS (Virtual Private LAN Service) have limited redundancy, multicast optimization, and provisioning simplicity.

Limitations of Traditional Solutions

Current VPLS solutions lack support for flexible multihoming with all-active redundancy mode and have complex provisioning requirements. Also, traditional data center designs, such as port channel based designs, have limited bandwidth utilization, scalability, and network resilience.

Evolution of Data Center Networks

Data center networks have evolved significantly over the past decade with new technologies and designs emerging to address the limitations of traditional solutions. The use of VxLAN and BGP in data center networks offers a promising solution for building massive, scalable, and efficient data centers.

VxLAN and BGP: A New Approach

You can use VxLAN (Virtual Extensible LAN) and BGP (Border Gateway Protocol) to build modern data centers that are scalable, efficient, and resilient. By using these technologies, you can create a network with flexible multihoming, efficient traffic management, and simplified provisioning.

Use the following sections to learn how VxLAN and BGP can be used to build a large data center.

Data Center IP Fabric Architecture

A typical data center deployment consists of three types of devices:

1. Leaf devices: Connected to physical servers or hypervisors that host virtual machines (VMs)
2. Spine devices: Interconnected with leaf devices in a Clos fabric topology, providing multiple paths for efficient traffic load-balancing and redundancy
3. Border nodes: Devices that connect the data center to external networks to handle north-south traffic

Key Characteristics

1. Clos fabric topology: A matrix of connections between network devices to enable horizontal scalability and efficient traffic management.
2. IP-based connectivity: The fabric is built on IP connectivity to enable flexible and scalable network design.
3. Point of Delivery (PoD): A group of leaf and spine devices that can be interconnected using super spines to form a larger data center.

Border Node Functionality

The border node, whether a border leaf or border spine, is at the edge of the data center and provides connectivity to external networks. This provides access to services, applications, or servers from the internet.

Benefits

1. Scalability: The IP fabric architecture provides horizontal scalability, making it easy to add new devices and increase capacity.
2. Redundancy and resiliency: Multiple paths between leaf devices ensure efficient traffic load-balancing and redundancy.
3. Efficient traffic management: The Clos fabric topology provides efficient traffic management to reduce the risk of network congestion and downtime.

Underlay Network

The underlay network provides IP connectivity between devices in a data center fabric. A routing protocol is needed to provide IP reachability and Layer 3 capabilities such as ECMP and load balancing.

BGP as Underlay Protocol

BGP (Border Gateway Protocol) is used commonly as the underlay protocol in data center fabrics. You have two options for building the underlay using BGP:

- iBGP (Internal BGP): Leaf devices establish iBGP sessions with spine devices, and spine devices act as route reflectors. Each leaf and border node has a dedicated loopback address that serves as a VTEP endpoint.
- eBGP (External BGP): Each leaf device is configured in a different AS, while all spine devices are in a single AS, and all border nodes are in a single AS. eBGP multipath is enabled for load balancing IP traffic.

Key Features

Underlay networks provide the following key features:

- IP reachability: BGP provides IP reachability between devices in the fabric.
- ECMP and load balancing: BGP provides ECMP and load balancing features, ensuring efficient traffic management.
- L2VPN EVPN: BGP negotiates L2VPN EVPN address family to synchronize overlay network addresses (MAC, ARP, or prefix routes).

Benefits

Underlay networks provide the following overall benefits:

- Scalability: BGP-based underlay architecture provides scalable and efficient network design.
- Flexibility: Both iBGP and eBGP options provide flexibility in designing the underlay network.
- Efficient traffic management: BGP provides efficient traffic management to reduce the risk of network congestion and downtime.
- Recommendation: Physical interface or Port-channel used as underlay (L3 interface) for VxLAN tunnel should not be part of any bridge-domain or sub-interface configuration.

Overlay Networks

With an underlay network in place, each device in the fabric can reach interface and loopback addresses. This is a prerequisite for setting up overlay networks.

Overlay networks are logical or virtual tunnels that provide communication between Virtual Machines (VMs) or resources placed on physical servers. These networks can support Layer 2 or Layer 3 transport between VMs, applications, or services while hiding the underlying network infrastructure.

By using overlay networks such as VXLAN, you can create a scalable and flexible network architecture that supports multiple applications and services.

Key Components

Overlay networks have the following key components:

- Tunnel Encapsulation: A VXLAN tunnel between VTEPs can carry traffic for multiple VXLAN networks. The VXLAN Network Identifier (VNI) in the VXLAN header identifies the VXLAN network to which the encapsulated traffic belongs.
- Control Plane: The control plane exchanges VM or tenant application topology information (MAC addresses and IP routes) between tunnel endpoints.
- Data Plane Encapsulation: The data plane encapsulates and forwards traffic between overlay tunnel endpoints across the virtual tunnel.

VXLAN and VXLAN Tunnel Endpoints (VTEPs)

VXLAN is a tunneling encapsulation that can transport data packets over an underlying IP network. It uses a MAC-in-UDP encapsulation to extend Layer 2 segments to overlay Layer 2 over a Layer 3 network.

Each Leaf and Border Node has a loopback interface configured as a VTEP. VXLAN tunnels can be statically provisioned or dynamically created using protocols like BGP.

Benefits

Overlay networks, such as VXLAN, have the following benefits:

- Layer 2 and Layer 3 connectivity between VMs, applications, or services
- Network virtualization, which hides the underlying network infrastructure
- Scalability to support multiple tenants and services

Layer 2 or Layer 3 Multitenancy with VxLAN

In a multitenant environment, Virtual Machines (VMs) deployed on servers behind Leaf nodes may require Layer 2 or Layer 3 connectivity between them. To achieve this, data packets are encapsulated in VxLAN packets at the ingress Leaf and routed across the fabric to the egress Leaf where they are tunnel terminated and forwarded to the destination VM.

By using VxLAN VNIs, you can provide scalable and flexible Layer 2 and Layer 3 multitenancy in your network.

VLAN and VNI Mapping

Each tenant's traffic is segregated using VLANs, which are mapped to a tenant ID or VNI (VxLAN Network Identifier). A VNI represents a broadcast domain, similar to a VLAN, but in the Layer 3 world. VNIs are of two types:

- L2VNI represents a Layer 2 broadcast domain, which is used for bridging traffic.
- L3VNI represents a Layer 3 domain or VRF, which is used for routing traffic.

Key Concepts

A VNI is carried in the VxLAN header to map traffic to the correct broadcast domain or VRF.

A bridge domain (BD) represents a VLAN or broadcast domain in Extreme OS ONE. Each bridge domain uses a unique L2VNI, while each routed domain uses a unique L3VNI to identify the VRF. Incoming VLANs are mapped to L2VNIs for bridging or L3VNIs for routing.

While VLANs and VRFs have local significance, VNIs are globally significant. VxLAN VNIs provide Layer 2 or Layer 3 isolation across the fabric, enabling multitenancy.

Basic Terminologies

The following key terms are used in these topics:

1. Bridge Domain: A single broadcast domain, which can be mapped to a single VLAN domain or group of VLAN domains. Each bridge domain has a MAC table for Layer 2 bridging.
2. Ethernet VPN (EVPN) Instance: A logical binding of bridge domains that participate in a single broadcast domain across multiple devices. EVPN instances can represent a single broadcast domain or a group of broadcast domains.
3. MAC-VRF: A representation of Layer 2 forwarding instance that is mapped to a single EVPN instance.
4. L3 Interface: A logical Layer 3 interface (VE) associated with a single VRF instance that is linked to a bridge domain and EVPN instance.
5. VRF Instance: A virtual routing and forwarding Layer 3 instance that represents a specific customer Layer 3 constructs.
6. NVE (Network Virtualization Edge): A device that can handle VxLAN tunneled packets by originating and terminating VxLAN tunnels based on VxLAN Tunnel Endpoint (VTEP) IPs.
7. NVO (Network Virtualization Overlay): A group of devices with the same mapping of VxLAN Network Identifier (VNI) mapping to extend to Layer 2 or Layer 3 domains.

Key Concepts

Devices in the same NVO must have the same VNI-domain mapping to extend Layer 2 or Layer 3 domains.

A VLAN-based service interface represents a single broadcast domain, while a VLAN-bundle service interface represents a group of broadcast domains.

IP Topology

Use this topic to view the topology diagram of 3-Stage Clos and 5-Stage Clos.

3-Stage Clos with Border Leaves

Figure 12 shows an example of 3-Stage Clos with border leaves.

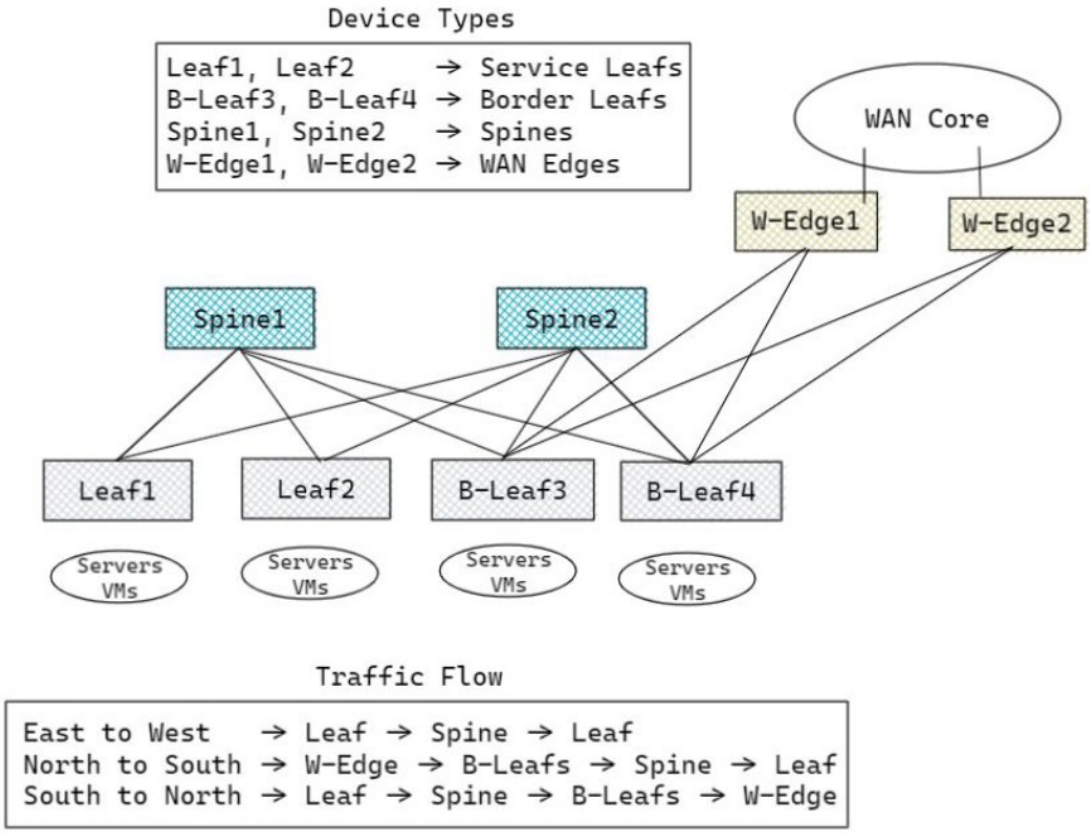


Figure 12: 3-Stage Clos with border leaves

3-Stage Clos with Collapsed Spines

Figure 13 shows an example of 3-Stage Clos with collapsed spines.

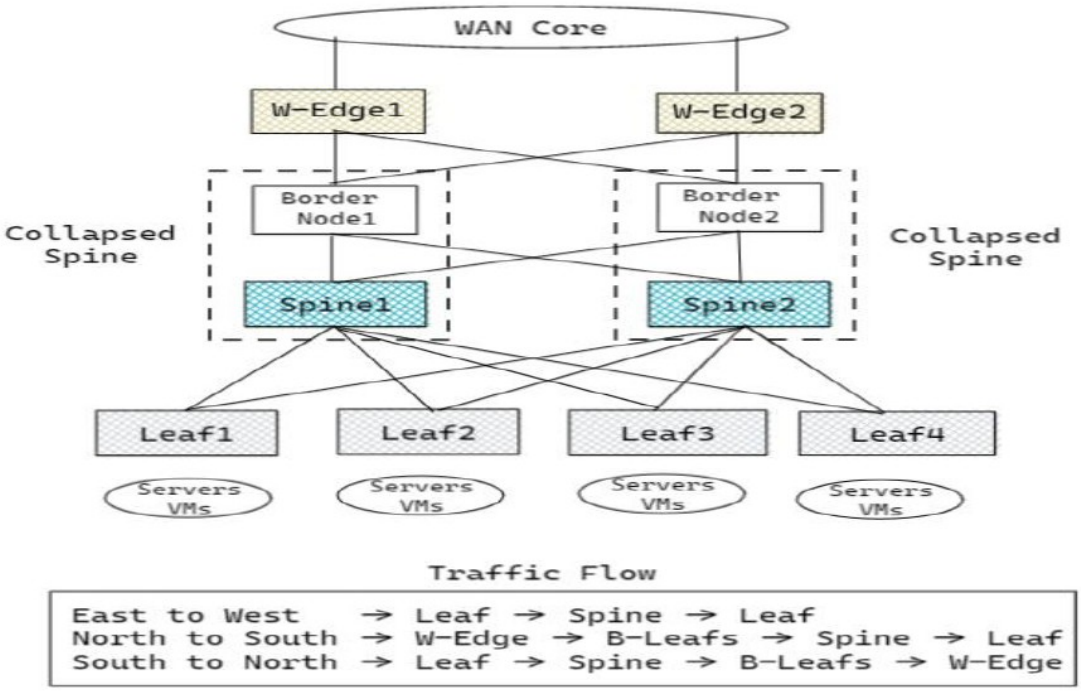


Figure 13: 3-Stage Clos with collapsed spines

5-Stage Clos

Figure 14 shows an example of 5-Stage Clos.

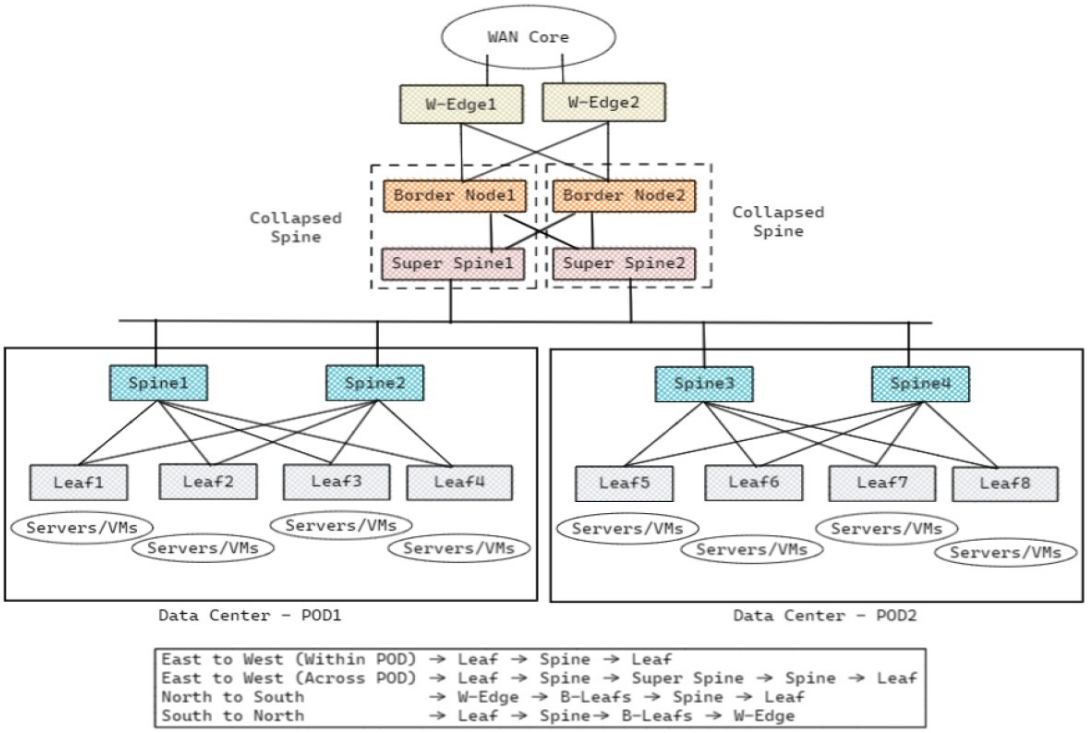


Figure 14: 5-Stage Clos

Sample Topologies

Figure 15 shows a sample topology with EVPN multihoming.

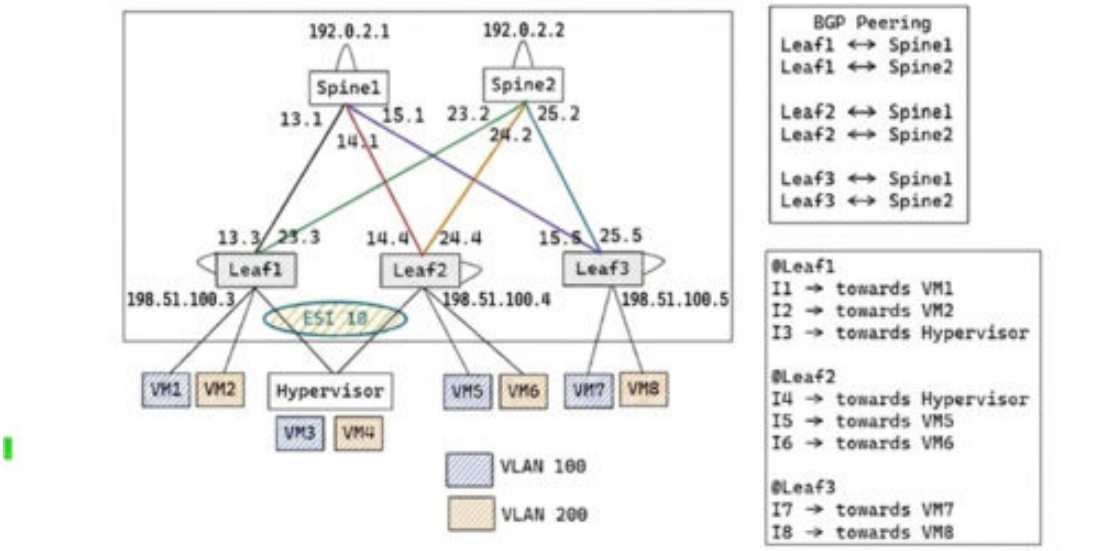


Figure 17: Sample Topology with Physical connections

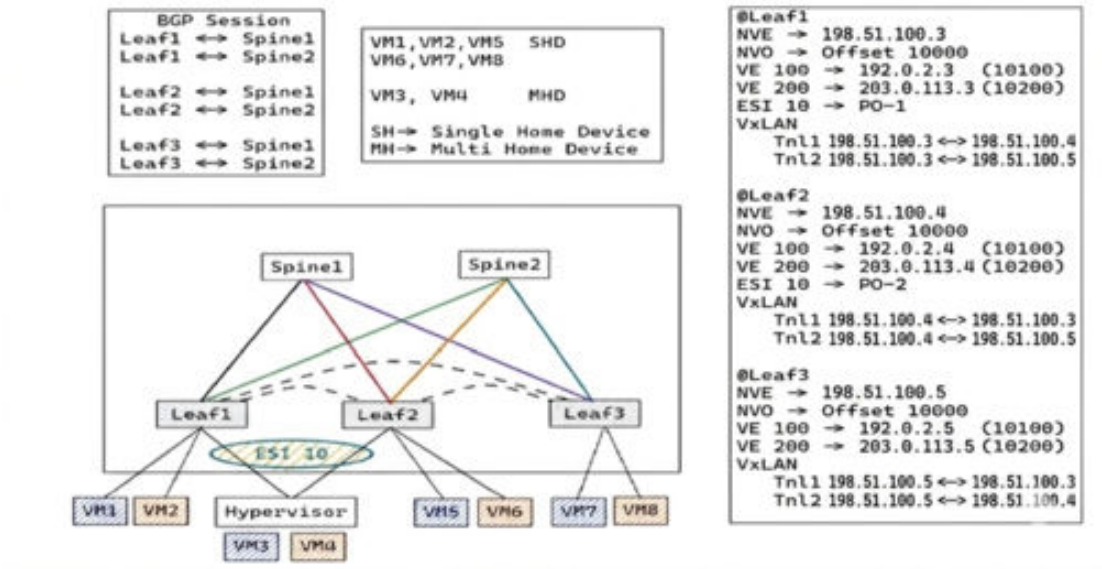


Figure 15: Sample topology with EVPN multihoming

Figure 16 shows a sample topology with MLAG and EVPN.

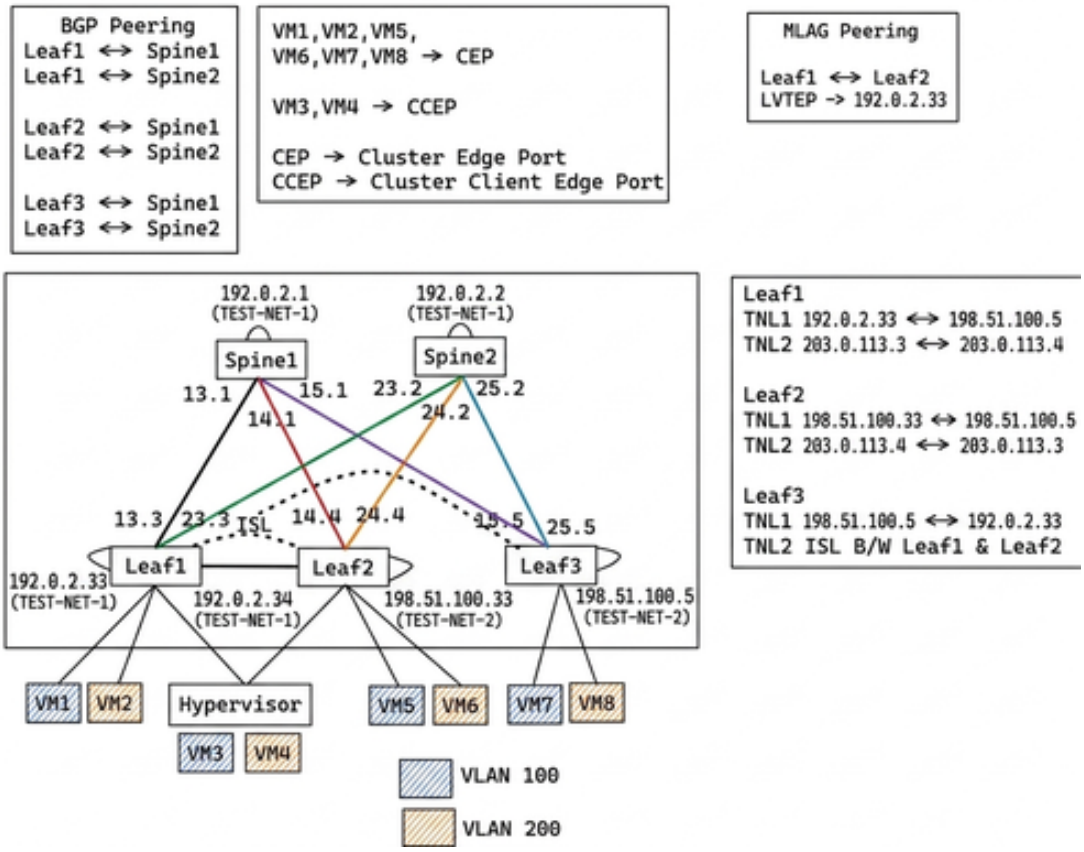


Figure 16: Sample topology with MLAG and EVPN

MAC Synchronization (Type 2)

In an EVPN network, MAC synchronization is crucial for efficient traffic forwarding. When a device learns a new MAC address, it propagates it to other devices in the network.

Local MAC Learning

The following is step-by-step overview of how MAC learning works:

1. A device receives a packet with an unknown destination MAC address and floods it across all member ports in the VLAN.
2. The packet is also sent over a tunnel to other devices in the network, which terminate the tunnel and perform local flooding.
3. The device that learns the new MAC address updates its hardware cache and propagates the MAC learned event to BGP.
4. BGP originates a Type 2 MAC route, which carries the MAC address, BD ID, and source interface information.
5. The MAC route is advertised to other devices in the network, which import it if it matches their configured RT.

MAC Route Origination

A MAC route that a device originates includes the following information:

- MAC address
- Bridge domain ID
- Source interface (IFINDEX)
- Sequence number (to ensure correct MAC/IP advertisement route retention)
- Sticky bit (for static MAC routes)

Received MAC Routes

When a device receives a MAC Type 2 route, it goes through regular EVPN route processing. If the route is imported, the device updates its MAC table with the received information.

End-to-End Data Path

The following steps provide an overview of how L2 traffic is forwarded across the fabric:

1. A packet is received at a device, which performs a MAC lookup to determine the exit path.
2. The packet is encapsulated in a VxLAN header and routed towards the destination VTEP.
3. The packet is forwarded across the fabric, potentially being load-balanced across multiple spines.
4. The destination device terminates the tunnel, performs MAC lookup, and forwards the original L2 frame to the destination port.

MLAG Considerations

When MLAG is involved, MAC learning and route origination work slightly differently:

1. A device learns a new MAC address and synchronizes it with its MLAG peer.
2. The device originates a MAC route, which is advertised to BGP peers.
3. The MLAG peer also installs the remote MAC pointing to the ISL.

MAC Route Selection

The device selects the best MAC route source based on distance. The selected source is then propagated to other microservices, such as Fwd-HAL, BGP, and MLAG.

MAC Move

When a MAC address moves within a device or between devices, the device updates its MAC table and propagates the new information to other devices in the network.

Layer 2 Extension

IMR routes enable Layer 2 extension over a fabric by indicating device interest in extending specific Layer 2 domains. BGP triggers tunnel creation when a peer receives an IMR route. MLAG setups use a shared VTEP IP, which gives multiple devices access to the same tunnel.

Layer 3 Extension

ARP or ND Synchronization and Routing

This section explains how ARP or ND synchronization works in a network. This includes local ARP or ND learning, propagation of ARP or ND learn events, and origination of ARP or ND routes.

Local ARP or ND Learning

When a device receives an ARP packet, it updates its ARP cache and routing table with the source MAC-IP binding. The device then originates a Type 2 MACIP route, which carries the MAC address, IP address, BD, and VRF information.

Propagation of ARP or ND Learn Events

The ARP or ND learn event is propagated to other devices in the network through BGP. Each device imports the MACIP route into its EVPN instance and VRF instance, updating its ARP cache and routing table accordingly.

Origination of ARP or ND Routes

The origination of ARP or ND routes involves several key components:

- Sequence Number: A sequence number is used to ensure that devices retain the correct MAC or IP advertisement route when multiple updates occur for the same MAC address.
- Static ARP: You can configure static ARP entries, which are then propagated to other devices in the network.
- Route Selection: The system performs route selection based on the source of the route, with different distances associated with each source (such as local, BGP, MLAG, or static).

Asymmetric and Symmetric Routing

The system supports both asymmetric and symmetric routing:

1. Asymmetric Routing: In asymmetric routing, the packet is forwarded based on the ARP cache, and the DMAC is set to the destination VMs MAC address.

2. Symmetric Routing: In symmetric routing, the packet is forwarded based on the IP VRF routing table, and the DMAC is set to the egress leaf's MAC address.

MLAG and Route Selection

The system also supports Multichassis Link Aggregation Group (MLAG) and performs route selection based on the source of the route.

Centralized vs. Distributed Routing

The system can operate in either centralized or distributed routing modes, each with its own advantages and disadvantages.

EVPN Model Overview

The EVPN standard (RFC 7432) defines several EVPN service interface types. The following service interface types are used in the VxLAN-based implementations described in this guide:

- VLAN-Based Service Interface: A single bridge domain is mapped to a single EVPN instance, providing a 1:1 mapping between EVPN instance, MAC-VRF, and bridge domain. This approach provides granular route import or export at the bridge domain level but requires one EVPN instance per bridge domain.
- VLAN-Aware Bundle Service Interface: Multiple bridge domains can be mapped to a single EVPN instance, sharing the same RD and RT set. This approach reduces EVPN instance configuration requirements but lacks granularity in importing or exporting routes on a per bridge domain basis.

Key Differences

The differences between a VLAN-Based Service Interface and a VLAN-Aware Bundle Service Interface are as follows:

1. VLAN-Based: 1:1 mapping between EVPN instance and bridge domain, granular route control, but more configuration is required.
2. VLAN-Aware Bundle: Multiple bridge domains per EVPN instance, reduced configuration, but less granular route control.

Implications

The choice between VLAN-Based and VLAN-Aware Bundle service interfaces depends on the specific network requirements and the trade-off between configuration complexity and route control granularity.

Module Interactions for BGP EVPN

The following sections outline the interactions between BGP and other subsystems to support EVPN.

GP and Management Service Interaction

The management service needs to be enhanced to support EVPN, including the following configurations:

1. EVPN instance: Installing EVPN instance configuration commands
2. VRF instance: Installing VRF instance configuration commands
3. ESI: Installing ESI configuration commands
4. BGP-specific: Installing BGP-specific configuration commands for EVPN

BGP and Interface Manager Interaction

The interface manager supports EVPN functionality, including the following features:

- Handling of tunnel creation and deletion requests from BGP
- L2VNI and L3VNI membership (adding or deleting BGP-triggered tunnels between L2VNI and L3VNI)
- Bridge domain commands to map EVPN instances
- Enhanced overlay configuration to have a single VNI-domain per NVO

BGP and UFTM Interaction

The Unified Forwarding Table Manager (UFTM) supports EVPN functionality that includes the following features:

1. Remote MAC route installation: Installing, deleting, or updating remote MAC routes pointing to VxLAN tunnels
2. Remote ARP route installation: Installing, deleting, or updating remote ARP routes pointing to VxLAN tunnels
3. Next hop resolution: Resolving next hops for VTEPs (BGP next hops)

BGP and Forward HAL Interaction

Fwd-HAL supports EVPN functionality that includes blocking rules to prevent traffic for non-designated forwarders and port channels.

BGP and Kernel Interaction

BGP packets are sent and received through the kernel with no requirements for BGP EVPN.

BGP and ARP/ND Interaction

The ARP or ND module supports EVPN functionality, including remote ARP route installation. This includes installing, deleting, or updating remote ARP routes pointing to VxLAN tunnels and handling MAC sync requests for L2VNIs.

BGP EVPN Configuration Examples

BGP EVPN Neighbor Configuration Examples

This section contains configuration examples of basic BGP EVPN neighbor configurations.

Configuring a BGP EVPN Peer Group

The following example shows how to use the **peer-group** command to configure a BGP peer group within a VRF instance:

```
device# configure terminal
device(config)# vrf default-vrf
device(config-vrf-default-vrf)# router bgp
device(config-vrf-bgp)# local-as 3
device(config-vrf-bgp)# peer-group BGPEVPN2
device(config-vrf-bgp-pg)#  remote-as 1
device(config-vrf-bgp-pg)# nvo nvo1
device(config-vrf-bgp-pg)# timers
device(config-vrf-bgp-pg-timers)# hold-time 30
device(config-vrf-bgp-pg-timers)# exit
device(config-vrf-bgp-pg)# address-family ipv4 unicast
device(config-vrf-bgp-pg-ipv4u)# activate
device(config-vrf-bgp-pg-ipv4u)# exit
device(config-vrf-bgp-pg)# exit
device(config-vrf-bgp-pg)# address-family l2vpn evpn
device(config-vrf-bgp-pg-l2vpn-evpn)# activate
device(config-vrf-bgp-pg-l2vpn-evpn)# exit
device(config-vrf-bgp-pg)# neighbor 192.0.2.0
device(config-vrf-bgp-pg-neighbor)# exit
device(config-vrf-bgp-pg)# neighbor 192.0.2.1
device(config-vrf-bgp-pg-neighbor)#
```

As with any other VPN technology, EVPN neighbors must always be in the default VRF. The preceding example shows an EVPN address family being activated for IPv4 BGP neighbors.

Retaining EVPN Routes without Importing Them

For spine or specific border leaf cases, routes are not imported into MAC-VRF or IP-VRF tables, but instead are reflected or re-advertised to iBGP or eBGP peers. The following configuration is required to retain all routes in the VPN table:

```
device# configure terminal
device(config)# vrf default-vrf
device(config-vrf-default-vrf)# router bgp
device(config-vrf-default-bgp)# address-family l2vpn evpn
device(config-vrf-bgp-l2vpn-evpn)# retain-route-target-all
device(config-vrf-bgp-l2vpn-evpn)#
```

Keeping Next-Hop Unchanged for eBGP Neighbors

When BGP advertises received routes to an eBGP peer, the next hop value in the route is modified to carry the local peering IP address. In the case of a BGP EVPN VxLAN IP Fabric, the next hop value in the route is the VTEP IP address of the advertising router. A BGP EVPN spine distributing the routes to other leaf nodes should not modify the

next hop value. Each eBGP peer activated under an EVPN address family should be configured so that it does not modify the next hop value, as in the following example:

```
device# configure terminal
device(config)# vrf default-vrf
device(config-vrf-default-vrf)# router bgp
device(config-vrf-bgp)# peer-group BGPEVPN2
device(config-vrf-bgp-pg)# address-family l2vpn evpn
device(config-vrf-bgp-pg-l2vpn-evpn)# nexthop-unchanged
device(config-vrf-bgp-pg-l2vpn-evpn)#
```

Configuring iBGP EVPN Neighbors

In a full iBGP mesh of EVPN nodes, a leaf node needs to peer only with all other leaf nodes. It does not need a session with any spine node. The configuration of a route-reflector client on the spine nodes is not required.

In a non-full iBGP mesh configuration, to reflect routes from one iBGP neighbor to another iBGP neighbor, the iBGP neighbors should be configured as route-reflector clients, as in the following example:

```
device# configure terminal
device(config)# vrf default-vrf
device(config-vrf-default-vrf)# router bgp
device(config-vrf-bgp)# address-family l2vpn evpn
device(config-vrf-bgp-l2vpn-evpn)# peer-group BGPEVPN2
device(config-vrf-bgp-pg)# route-reflector-client
```

The '"client-to-client-reflection" feature is by default enabled under the EVPN address family.

The route-reflector configuration does not force routes that are not imported to be retained in BGP. The **`retain route-target-all`** command as described previously is required in all cases.

Negotiating only EVPN Address Family

For better resiliency, separate BGP neighbors for overlay and underlay may be desired. This separation lets you manipulate, debug, and maintain EVPN neighbors without affecting underlay connectivity. You can achieve this separation by deactivating the underlay address family for EVPN neighbors.

IPv4 neighbors are automatically activated under the IPv4 unicast address family, and they must be deactivated as in the following example:

```
device# configure terminal
device(config)# vrf default-vrf
device(config-vrf-default-vrf)# router bgp
device(config-vrf-bgp)# peer-group BGPEVPN2
8730-32d(config-vrf-bgp-pg-ipv4u)# address-family ipv4 unicast
8730-32d(config-vrf-bgp-pg-ipv4u)# no activate
8730-32d(config-vrf-bgp-pg-ipv4u)# exit
8730-32d(config-vrf-bgp-pg)# address-family l2vpn evpn
8730-32d(config-vrf-bgp-pg-l2vpn-evpn)# exit
8730-32d(config-vrf-bgp-pg)# exit
8730-32d(config-vrf-bgp)# peer-group BGPEVPN2
8730-32d(config-vrf-bgp-pg)# address-family l2vpn evpn
```

```
8730-32d(config-vrf-bgp-pg-l2vpn-evpn)# activate
8730-32d(config-vrf-bgp-pg-l2vpn-evpn)#
```

Displaying the BGP EVPN Neighbor Configuration

The following example displays the configuration of the default VRF instance that is running currently on the device.

```
device# show running-config vrf default-vrf

vrf default-vrf
  evpn-instance evpn1
    nve nve2
    ignore-as
    bridge-domain 21,44,100,200,222,300
  !
  router bgp
    local-as 3
    address-family ipv4 unicast
    address-family l2vpn evpn
      retain-route-target-all
    !
    peer-group BGPEVPN2
      remote-as 1
      nvo nvo1
      timers
        hold-time 30
      !
      route-reflector-client
      address-family ipv4 unicast
      address-family l2vpn evpn
        activate
      !
      neighbor 192.0.2.0
      neighbor 192.0.2.1
    !
  !
!
```

BGP EVPN Instance Configuration Examples

With a VLAN based service model, each VLAN or BD corresponds to a unique EVPN instance or MAC-VRF. Each route in EVPN belonging to a MAC-VRF has a unique route distinguisher (RD) that differentiates routes from other MAC-VRFs.

In addition, sets of route targets (export RTs) are associated with each MAC-VRF. These RTs are applied to the routes while advertising. RTs in the route are matched against the configured import RTs. If any of the RTs match, the route is imported. Otherwise, it is discarded.

Note
Auto RD is mandatory for EVI extension with manual RD and RT/auto RT.

The following example associates an EVPN instance named evpn1 with the default VRF and configures the instance:

```
device# configure terminal
device(config)# vrf default-vrf
```

```
device(config-vrf-default-vrf)# evpn-instance evpn1
device(config-evpn-evpn1)# nve nve2
device(config-evpn-evpn1)# ignore-as
device(config-evpn-evpn1)# bridge-domain 21,44,100,200,222,300
device(config-evpn-evpn1)#
```

Note
Configuration of an EVPN instance is supported only for the default VRF (not for any other user-configured VRF instance).

In this example, the instance uses the **ignore-as** command to ignore the local AS in the AS path of a received EVPN route. Such routes are accepted as valid routes and might qualify for best path selection.

The following example displays the configuration of the default VRF that is running currently on the device. In this example, an EVPN instance named evpn1 is configured:

```
device# show running-config vrf default-vrf

vrf default-vrf
  evpn-instance evpn1
    nve nve2
    ignore-as
    bridge-domain 21,44,100,200,222,300
  !
  router bgp
    local-as 3
    address-family ipv4 unicast
    address-family l2vpn evpn
      retain-route-target-all
    !
    peer-group BGPEVPN2
      remote-as 1
      nvo nvo1
      timers
        hold-time 30
      !
      route-reflector-client
      address-family ipv4 unicast
      address-family l2vpn evpn
        activate
      !
      neighbor 192.0.2.0
      neighbor 192.0.2.1
    !
  !
!
```

Static VxLAN

This topic covers the configuration and behavior of static VxLAN tunnels for Layer 2 or Layer 3 traffic.

Static VxLAN Overview

The configuration setup involves VxLAN L2 gateways (Leaf 1, 2, and 3) that extend bridge domains through VxLAN tunnels.

Traffic Behavior

The following list describes Static VxLAN traffic behavior:

- BUM Traffic: When traffic arrives on AC ports without a MAC table entry, it is flooded to VxLAN tunnels that are members of the bridge domain using head-end replication.
- MAC Learning: MAC addresses are learned over VxLAN tunnels, and the source MAC address of the inner payload is used for learning.

  The **`show mac-address-table`** command displays MAC address entries for a bridge domain, including static and dynamic entries learned over VxLAN tunnels.

```
device# show mac-address-table bridge-domain 500

Total number of Mac Entries: 2
Hardware Status Codes - #:Failed
Mac-Address              Type             Interface
------------------------ ---------------- ------------------------
00:10:20:30:40:50        Static           ethernet 0/33.90133
00:10:20:30:AA:AA        Dynamic          tunnel 2001:db8::1
device#
```

- VxLAN Tagging: Extreme OS ONE strips VLAN tags before sending traffic over VxLAN tunnels in VLAN mode and default mode bridge domains.

Split Horizon

Each NVO has a split horizon group, and BUM traffic that is received on a VxLAN tunnel is not flooded to another tunnel in the same group. BUM traffic received on a VxLAN tunnel in one split horizon group can be flooded to VxLAN tunnels in other groups.

Head-End Replication

When traffic arrives on the access ports but does not find a MAC table entry, it is flooded to the VxLAN tunnels that are members of the bridge domain. In this case, head end replication sends a copy of the same packet over multiple VxLAN tunnels.

Sample Topology

A sample configuration involves VxLAN Layer 2 gateways (Leafs 1, 2, and 3) that extend bridge domains through VxLAN tunnels. Key components of this topology are as follows:

- VxLAN Tunnels: Leaf 1 and Leaf 3 are connected through two VxLAN tunnels (Tunnel 1 and Tunnel 2) that extend bridge domains 500 and 600, respectively.
- Bridge Domains: Bridge domain 500 is extended to Leaf 3 through VxLAN Tunnel 1, and bridge domain 600 is extended to Leaf 3 through VxLAN Tunnel 2.
- Host Mapping: Host H1 on bridge domain 500 is mapped to VNI 500, and traffic from H1 is flooded to VNI 500 on remote leaf nodes through VxLAN Tunnel 1.

Example Output

The **`show mac-address-table`** command displays MAC address entries for a bridge domain, including static and dynamic entries learned over VxLAN tunnels.

Static VxLAN Limitations

The Extreme 8730 platform has the following limitations:

- VxLAN Traffic Termination: VxLAN traffic will be terminated with the correct VTEP IP and VNI even if the tunnel is not extended in the bridge domain.
- Tunnel Underlay: Tunnel underlay is supported only in the default VRF.
- Single VNI Domain: Only a single VNI domain is supported.
- Single VxLAN Next hop: A single VxLAN next hop (underlay) is supported per physical port or port channel. This results in the following restrictions:
  - Multiple IPv4 tunnels can share the same underlay, but more than one IPv4 VxLAN next hop is not supported on the same port.
  - More than one IPv6 VxLAN next hop is not supported on the same port.
  - IPv4 and IPv6 tunnels cannot share a next hop.
  - IPv6 tunnels with different source VTEP IPs cannot share a next hop.
- MAC Learning: MAC learning continues to happen on the static VxLAN tunnel/BD even when the tunnel is not added as a member of the bridge domain. This can

cause issues when the VxLAN tunnel is added to the BD on one side but not on the other side.

- VE as Static VxLAN Underlay: VE (Virtual Ethernet) is not supported as a static VxLAN underlay.
- Recommendation: Physical interface or Port-channel used as underlay for VxLAN tunnel should not be part of any Bridge-domain or sub-interface configuration.

Event Log Messages

The system generates log messages for tunnel operational status changes. The log messages indicate when a tunnel becomes operationally down or up, which provides important information to monitor and troubleshoot the tunnel status:

- Tunnel Down: `interface-mgr[7]: Level:info LogID:25014 Msg:Interface tunnel ISL_10.7.8.7 Operationally DOWN`
- Tunnel Up: `interface-mgr[7]: Level:info LogID:25013 Msg:Interface tunnel ISL_10.7.8.7 Operationally UP`

Configure Static VxLAN

You should configure NVE with a valid IPv4 or IPv6 address and associate it with a VRF.

You should configure an NVO with an NVE, a VNI domain, and a split horizon group. Tunnels will not come up if any of these configurations are missing.

Configuring Network Virtualization Overlay (NVO)

NVO defines a logical group of VxLAN tunnels that belong to the same IP fabric domain. To configure NVO, follow these steps:

1. Create NVO
```
DUT1(config-overlay)# nvo base
```
2. Associate NVE
```
DUT1(config-nvo-base)# member nve base
```
3. Associate VNI Domain
```
DUT1(config-nvo-base)# member vni-domain base
```
4. Configure Split Horizon Group
```
DUT1(config-nvo-base)# split-horizon group1
```

Configuring Network Virtualization Endpoint (NVE)

NVE defines the VxLAN tunnel endpoints. To configure NVE, follow these steps:

1. Create an NVE.
```
DUT1(config-overlay)# nve base
```
2. Specify the loopback Interface.
```
DUT1(config-nve-base)# source-interface loopback 1
```

Configuring VNI Domain

VNI domain defines the mapping of bridge domains to VNIs. To configure VNI domain, follow these steps:

1. Create a VNI Domain.
```
DUT1(config-overlay)# vni-domain base
```
2. Configure Auto-Offset.
```
DUT1(config-nve-base)# auto-offset 60000
```

Manual VNI Configuration for Bridge Domain

To override the auto-offset for a specific bridge domain, use the following configuration:

1. Configure Bridge Domain.
```
DUT1(config)# bridge-domain 200
```
2. Configure VNI Mapping.
```
DUT1(config-bd-200)# vni-domain base vni 2000
```

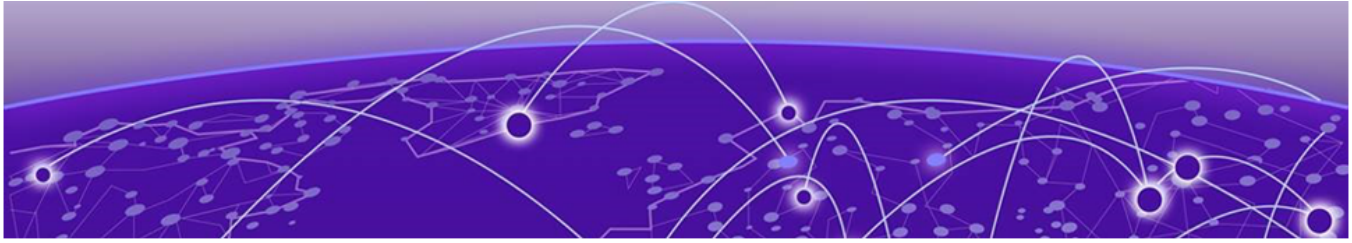
Default VNI Offset Configuration

To configure the default VNI auto-offset, use the following command:
```
DUT1(config-overlay)# default-vni-offset 100000
```

VxLAN Tunnel Configuration

To configure IPv4 or IPv6 VxLAN tunnels, follow these steps:

1. Create Tunnel Interface.
```
DUT1(config)# interface tunnel ipv4_tunnel
Or
DUT1(config)# interface tunnel ipv4_tunnel
```
2. Specify VxLAN Type.
```
DUT1(config-tun-ip_v4_tunnel)# type vxlan
```
3. Associate NVE.
```
DUT1(config-tun-ip_v6_tunnel)# source nve base
```
4. Specify IPv4 destination IP.
```
DUT1(config-tun-ip_v4_tunnel)# destination 200.200.200.1
```



Resilient Hashing

Resilient hashing enhances your network resilience by intelligently managing traffic flows during topology changes. This reduces service disruption while maintaining optimal path utilization.

Static Hashing Versus Resilient Hashing in ECMP Routing

Resilient hashing minimizes traffic disruption in equal cost multipath (ECMP) routing when ECMP links fail. Traditional static hashing redistributes all traffic flows when ECMP group membership changes, which could interrupt service. Whereas resilient hashing preserves existing flow mappings to active links when member links fail by permitting traffic redistribution only on failed links.

Static Hashing in ECMP Routing

In traditional ECMP routing, static hashing is used to distribute traffic across multiple paths. This method uses a hash based on packet headers and a subsequent modulo operation based on the number of paths N (hash % N, where N = number of paths) in the ECMP group.

If a member is added or removed in the ECMP group, because of a modulo number change, the static hashing algorithm might choose a new path for existing flows. When paths change, the modulo value changes to potentially remap all existing flows to different paths. The change in the mapping of the flow might cause traffic disruption.

Resilient Hashing in ECMP Routing

Resilient hashing uses a concept called a *flowset* table of much larger size F. The ECMP path is picked by (hash % F), where all existing paths are distributed multiple times to fill the table. When a path fails, only those flows specifically mapped to that path require redistribution, while flows on remaining active paths stay intact.

This approach ensures that existing flows are not remapped when links change. The resilient hashing flowset table maintains flow consistency, even when links are added or removed.

Key Benefits

Resilient hashing provides several key benefits:

- Minimizes traffic disruption: Preserves existing flow mappings when ECMP members change
- Reduces convergence time: Limits redistribution to only affected flows
- Maintains service quality: Prevents unnecessary flow remapping that could impact latency-sensitive applications
- Improves network stability: Reduces the impact of link failures by using a flowset table to maintain flow consistency and ensures that existing flows are preserved even when links are added or removed

Limitations

The resilient hashing feature has the following limitations:

- Protocol next hop changes: Resilient hashing cannot prevent explicit ECMP path changes directed by routing protocols. When protocols modify the next-hop set directly, complete path reprogramming occurs.
- Resource sharing: The Extreme 8520, Extreme 8720, and Extreme 8730 platforms share ECMP physical tables between normal and resilient hashing modes, which potentially limits scale.

Best Practices

The following list describes the configuration and best practices for resilient hashing:

- Configure appropriately: Enable resilient hashing only on VRFs that require flow stability during convergence events.
- Monitor resources: On the Extreme 8520, Extreme 8720, and Extreme 8730 platforms, monitor ECMP table usage to ensure sufficient resources for both normal and resilient hashing mode operations.
- Plan capacity: On the Extreme 8520, Extreme 8720, and Extreme 8730 platforms, consider the 50 percent resource limitation when designing network topology.

Impact of Configuration Changes

Changing your resilient hashing configuration affects the system as follows:

- Enabling or disabling resilient hashing: Triggers complete next hop group reprogramming for all affected routes
- Changes to the **`ecmp-max-path`** setting: Causes route and next hop group re-installation
- Path sorting: The system sorts protocol next hops in increasing IP address order when trimming paths beyond the **`ecmp-max-path`** setting

Scalability

The scalability of resilient hashing differs among the platforms supported by Extreme OS ONE Switching and Routing.

Extreme 8520, Extreme 8720, and Extreme 8730 Platforms

These platforms share an ECMP resource physical table between normal mode ECMP groups and resilient hashing ECMP groups. Therefore, Extreme OS ONE Switching and Routing restricts physical table capacity to 50 percent, beyond which resilient hashing is treated as normal mode ECMP creation. Normal mode ECMP operations have no cap.

Unlike the Extreme 8520 and Extreme 8720 platforms, the Extreme 8730 platform does not have native hardware support for resilient hashing. Instead, resilient hashing is implemented in software on this platform.

Table 9 shows the ECMP member and group scalability for the Extreme 8520, Extreme 8720, and Extreme 8730 platforms.

Table 9: ECMP member and group scalability for the Extreme 8520, Extreme 8720, and Extreme 8730 platforms

Platform	Max ECMP groups (normal mode)	Max ECMP groups (RH mode)	Max ECMP members (normal mode)	Max ECMP members (RH mode)
Extreme 8520	2,000	256	16,000	16,000
Extreme 8720	2,000	256	16,000	16,000
Extreme 8730	4,000	256	32,000	16,000

Extreme 8820 Platform

This platform uses a dedicated hash mapping table for resilient hashing. In other words, it does not share an ECMP resource physical table between normal mode ECMP groups and resilient-hashing ECMP groups. Therefore, Extreme OS ONE Switching and Routing does not restrict physical table capacity on this platform.

On this platform, resilient hashing is implemented in hardware.

Table 10 shows the ECMP member and group scalability for the Extreme 8820 platform.

Table 10: ECMP member and group scalability for the Extreme 8820 platform

Max ECMP group size	Max ECMP group number	Max ECMP table size
16 (small) 64 (medium) 256 (large)	256 (small) 64 (medium) 32 (large)	256 (small) 64 (medium) 32 (large)

Configure Resilient Hashing

The device provides several CLI commands to configure and display the ECMP and resilient hashing settings. You configure the maximum number of ECMP paths globally (applies to all VRFs unless overridden) or per VRF.

Configure ECMP Maximum Paths

You configure the maximum number of ECMP paths globally or per VRF. For details on command syntax and parameters, see the *Extreme OS ONE Switching and Routing Command Reference Guide*.

The system applies the resilient hashing maximum paths setting in the following priority order:

1. VRF-specific **ecmp-max-path** setting (highest priority)
2. Global **ecmp-max-path** setting
3. System default setting (eight paths)

Configuring ECMP Maximum Paths Globally

To configure system level ECMP maximum paths to a nondefault value:

1. Access global configuration mode.
```
device# configure terminal
```
2. Enter system configuration mode.
```
device(config)# system
```
3. Enter global system configuration mode.
```
device(config-system)# global
```
4. Set the maximum ECMP paths. Use an integer from 2 to 128 (power of 2).
```
device(config-system-global)# ecmp-max-path 64
```
5. (Optional) Verify the configuration.
```
device(config-system-global)# do show running-config system global

system
  global
    ecmp-max-path 64
  !
```

```
!
device(config-system-global)#
```

Configuring ECMP Maximum Paths on a VRF

The default for a VRF is eight paths. Changing the default configuration takes effect immediately for all routes in the VRF.

To configure VRF-level ECMP maximum paths to another value:

1. Access global configuration mode.
   ```
   device# configure terminal
   ```
2. Specify the VRF name and enter VRF configuration mode.
   ```
   device(config)# vrf vrf1
   ```
3. Set the maximum ECMP paths on the VRF. Use an integer from 2 to 128 (power of 2). .
   ```
   device(config-vrf-vrf1)# ecmp-max-path 32
   ```
4. (Optional) Verify the configuration.
   ```
   device(config-vrf-vrf1)# do show running-configuration vrf vrf1

   vrf vrf1
      ecmp-max-path 32
   !
   device#
   ```

Enable Resilient Hashing on a VRF

You enable or disable resilient hashing on a per VRF basis. Resilient hashing is available for the default VRF as well as VRFs that you create.

Use the following procedure to configure resilient hashing on a VRF:

1. Access global configuration mode.
   ```
   device# configure terminal
   ```
2. Specify the VRF name and enter VRF configuration mode.
   ```
   device(config)# vrf vrf1
   ```
   You can specify the default VRF (named default-vrf) or a user-created VRF.
3. Enable resilient hashing on the VRF.
   ```
   device(config-vrf-vrf1)# resilient-hash
   ```
4. (Optional) Verify the configuration.
   ```
   device(config-vrf-vrf1)# do show running-configuration vrf vrf1

   vrf vrf1
      ecmp-max-path 32
      resilient-hash
   !
   device#
   ```

Display ECMP Maximum Paths Route Information for VRFs

The **show ipv4 route vrf all** and **show ipv6 route vrf all** commands display route information for all VRFs, including the resilient hashing status and maximum ECMP paths settings. For details on command syntax and parameters, see the *Extreme OS ONE Switching and Routing Command Reference Guide*.

The following example displays IPv4 routing table information for a VRF instance named vrf1:

```
device# show ipv4 route vrf vrf1

VRF:vrf1
---------------
Resilient Hash: Enabled
Ecmp Max Path: 32
device#
```

The following example displays IPv4 routing table information for all VRF instances:

```
device# show ipv4 route vrf all

VRF:mgmt-vrf
---------------
Total number of IPv4 routes: 3, Max Routes: Not Set
'[x/y]' denotes [preference/metric]
0.0.0.0/0, static, [1/1], tag 0, 4h1m47s
192.0.2.0/24, attached, [0/0], tag 0, 4h1m47s
192.0.2.20/32, local, [0/0], tag 0, 4h1m47s

VRF:vrf1
---------------
Resilient Hash: Enabled
Ecmp Max Path: 32

VRF:default-vrf
---------------
Resilient Hash: Disabled
Ecmp Max Path: 8
device#
```

The following example displays IPv6 routing table information for a VRF instance named vrf1:

```
device# show ipv6 route vrf vrf1

VRF:vrf1
---------------
Resilient Hash: Enabled
Ecmp Max Path: 32
device#
```

The following example displays IPv6 routing table information for all VRF instances:

```
device# show ipv6 route vrf all

VRF:mgmt-vrf
---------------
Total number of IPv6 routes: 1, Max Routes: Not Set
'[x/y]' denotes [preference/metric]
::/0, dhcp, [0/1024], tag 0, 3h58m23s
```

```
VRF:vrf1
---------------
Resilient Hash: Enabled
Ecmp Max Path: 32

VRF:default-vrf
---------------
Resilient Hash: Disabled
Ecmp Max Path: 8
device#
```

The following example displays IPv4 routing table information for the management VRF instance:

```
device# show ipv4 route vrf mgmt-vrf

VRF:mgmt-vrf
---------------
Total number of IPv4 routes: 3, Max Routes: Not Set
'[x/y]' denotes [preference/metric]
0.0.0.0/0, static, [1/1], tag 0, 4h1m47s
192.0.2.0/24, attached, [0/0], tag 0, 4h1m47s
192.0.2.20/32, local, [0/0], tag 0, 4h1m47s
device#
```

The following example displays IPv4 routing table information for the default VRF instance:

```
device# show ipv4 route vrf default-vrf

VRF:default-vrf
---------------
Resilient Hash: Disabled
Ecmp Max Path: 8
device#
```

Logging

The system generates logs for various events related to resilient hashing and ECMP configuration. You can use these logs to troubleshoot issues and monitor system behavior.

Following are several examples of logging:

- ECMP max path configuration: Logs show when the maximum ECMP path configuration is processed, including additions and deletions.
- Resilient hash configuration: Logs indicate when resilient hashing is enabled or disabled on a VRF.
- Dropping ECMP paths: Logs show when ECMP paths are dropped when the maximum path limit is exceeded.
- Encoding maximum ECMP path and resilient hashing: Logs indicate when the maximum ECMP path and resilient hashing information is encoded for HAL.