IP Routing Configuration Guide, Cisco Catalyst IE3x00 Rugged, IE3400 Heavy Duty, and ESS3300 Series Switches

Cisco Catalyst IE3200 Rugged Series
Updated June 25, 2026

Full Cisco Trademarks with Software License

Full Cisco Trademarks with Software License

THE SPECIFICATIONS AND INFORMATION REGARDING THE PRODUCTS IN THIS MANUAL ARE SUBJECT TO CHANGE WITHOUT NOTICE. ALL STATEMENTS, INFORMATION, AND RECOMMENDATIONS IN THIS MANUAL ARE BELIEVED TO BE ACCURATE BUT ARE PRESENTED WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED. USERS MUST TAKE FULL RESPONSIBILITY FOR THEIR APPLICATION OF ANY PRODUCTS.

THE SOFTWARE LICENSE AND LIMITED WARRANTY FOR THE ACCOMPANYING PRODUCT ARE SET FORTH IN THE INFORMATION PACKET THAT SHIPPED WITH THE PRODUCT AND ARE INCORPORATED HEREIN BY THIS REFERENCE. IF YOU ARE UNABLE TO LOCATE THE SOFTWARE LICENSE OR LIMITED WARRANTY, CONTACT YOUR CISCO REPRESENTATIVE FOR A COPY.

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

NOTWITHSTANDING ANY OTHER WARRANTY HEREIN, ALL DOCUMENT FILES AND SOFTWARE OF THESE SUPPLIERS ARE PROVIDED “AS IS" WITH ALL FAULTS. CISCO AND THE ABOVE-NAMED SUPPLIERS DISCLAIM ALL WARRANTIES, EXPRESSED OR IMPLIED, INCLUDING, WITHOUT LIMITATION, THOSE OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OR ARISING FROM A COURSE OF DEALING, USAGE, OR TRADE PRACTICE.

IN NO EVENT SHALL CISCO OR ITS SUPPLIERS BE LIABLE FOR ANY INDIRECT, SPECIAL, CONSEQUENTIAL, OR INCIDENTAL DAMAGES, INCLUDING, WITHOUT LIMITATION, LOST PROFITS OR LOSS OR DAMAGE TO DATA ARISING OUT OF THE USE OR INABILITY TO USE THIS MANUAL, EVEN IF CISCO OR ITS SUPPLIERS HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

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

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

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

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

1 Configure Bidirectional Forwarding Detection

Topics:

Bidirectional Forwarding Detection

Describes the Bidirectional Forwarding Detection (BFD) protocol for fast forwarding path failure detection.

Bidirectional Forwarding Detection (BFD) is a detection protocol that

- provides fast forwarding path failure detection times for all media types, encapsulations, topologies, and routing protocols
- offers a consistent failure detection method for network administrators, and
- enables uniform forwarding path failure detection at a consistent rate rather than variable rates for different routing protocol hello mechanisms.

BFD benefits

BFD provides fast forwarding path failure detection with consistent benefits for network operations:

- Network profiling and planning becomes easier because administrators can detect forwarding path failures at a uniform rate
- Reconvergence time remains consistent and predictable across different network scenarios

Prerequisites for BFD

Outlines the prerequisites for bidirectional forwarding detection configuration.

Before configuring bidirectional forwarding detection (BFD), ensure that all participating switches and routing protocols meet the following requirements:

- All participating switches must enable Cisco Express Forwarding and IP routing.
- Before BFD is deployed on a switch, it is necessary to configure one of the IP routing protocols that are supported by BFD. You should implement fast convergence for the routing protocol that you are using.

  See IP routing documentation for your version of Cisco IOS software for information on configuring fast convergence. See the "Restrictions for Bidirectional Forwarding Detection" section for more information on BFD routing protocol support in Cisco IOS software.

Restrictions for BFD

Outlines the limitations and constraints when implementing Bidirectional Forwarding Detection in network configurations.

Consider these restrictions when implementing BFD:

- BFD support is not available for all platforms and interfaces. To confirm if a specific platform or interface has BFD support and to obtain the most accurate platform and hardware restrictions, see the Cisco IOS software release notes for your software version.
- BFD HA is not supported.
- BFD Echo Sessions scale: Up to 28 BFD Sessions of 100ms interval, Echo-mode BFD sessions allowed per device.
- As of IOS XE 17.5.1, only OSPF and OSPFv3 support for BFD are supported.
- Supported Time Intervals by port type:

Port Type	Minimum Interval
Routed-Ports	100ms
SVIs	100ms

Port Type	Minimum Interval
L3 Port	250ms

BFD operation

Describes how BFD provides failure detection in the forwarding path between adjacent devices.

BFD operation is a detection protocol process that you enable at the interface and routing protocol levels. It provides a low-overhead, short-duration method of detecting failures in the forwarding path between two adjacent devices and includes the interfaces, data links, and forwarding planes.

BFD asynchronous mode operation

Cisco supports BFD asynchronous mode. BFD asynchronous mode depends on the sending of BFD control packets between two systems to activate and maintain BFD neighbor sessions between devices. Therefore, in order to create a BFD session, you must configure BFD on both systems (or BFD peers). A BFD session is created once BFD is enabled on the interfaces and at the device level for the appropriate routing protocols. BFD timers are negotiated, and the BFD peers begin to send BFD control packets to each other at the negotiated interval.

Neighbor relationships

Explains how BFD establishes relationships with new neighbors and removes relationships when network failures occur.

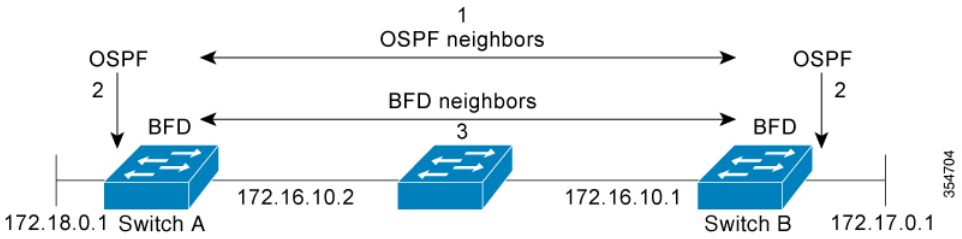
A neighbor relationship is a network connection that

- provides fast BFD peer failure detection times independently of media types, encapsulations, topologies, and routing protocols such as BGP, EIGRP, IS-IS, and OSPF,
- sends rapid failure detection notices to the routing protocols in the local device to initiate the routing table recalculation process, and
- contributes to greatly reduced overall network convergence time.

BFD neighbor session establishment and teardown process

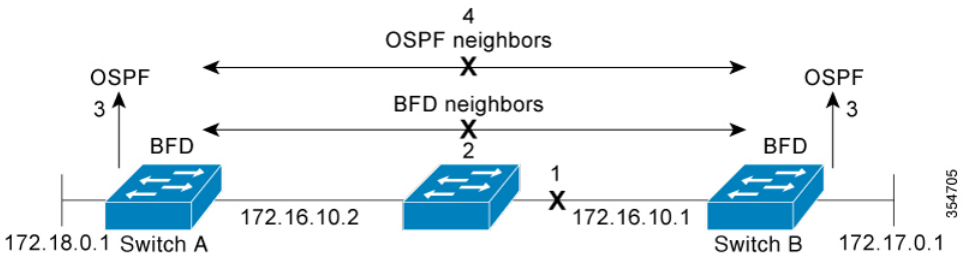
The figure here shows a simple network with two devices running OSPF and BFD. When OSPF discovers a neighbor (1), it sends a request to the local BFD process. It initiates a BFD neighbor session with the OSPF neighbor device (2). The BFD neighbor session with the OSPF neighbor device is established (3).

Figure 1: BFD process on a network configured with OSPF



When a failure occurs in the network (1), the BFD neighbor session with the OSPF neighbor device is torn down (2). BFD notifies the local OSPF process that the BFD neighbor is no longer reachable (3). The local OSPF process tears down the OSPF neighbor relationship (4). If an alternative path is available, the devices immediately start converging on it.

Figure 2: BFD process during a network failure



A routing protocol must register with BFD for every neighbor it acquires. Once a neighbor is registered, BFD initiates a session with the neighbor if a session does not already exist.

OSPF registers with BFD when:

- A neighbor finite state machine (FSM) transitions to full state.
- Both OSPF BFD and BFD are enabled.

On broadcast interfaces, OSPF establishes a BFD session only with the designated router (DR) and backup designated router (BDR). The session is not established between any two devices in a DROTHER state.

BFD detection of failures

Describes how BFD detects failures through control packets after session establishment and timer negotiations are complete.

BFD detection of failures is a forwarding path failure detection mechanism. It uses BFD control packets, which act like IGP hello protocols, to detect liveliness at an accelerated rate. BFD detects failures but requires the routing protocol to act to bypass a failed peer. It also shares session information with multiple client protocols through a single BFD session per peer. .

BFD failure detection characteristics

Once a BFD session is established and timer negotiations are complete, BFD peers send BFD control packets. The packets act like an IGP hello protocol to detect liveliness but at a higher rate. Consider this information about BFD failure detection:

- BFD is a forwarding path failure detection protocol. BFD detects a failure, but the routing protocol must act to bypass a failed peer.
- Starting with Cisco IOS XE Denali 16.3.1, Cisco devices support BFD version 0. Devices use one BFD session for multiple client protocols in the implementation. For example, if a network is running OSPF and EIGRP across the same link to the same peer, only one BFD session is established. BFD shares session information with both routing protocols.

BFD version interoperability

Describes how BFD version detection ensures that the highest common BFD version is run when there are two neighbors running on different BFD versions.

BFD version interoperability is a BFD capability that automatically detects BFD versions between neighbors. It establishes sessions using the highest common BFD version. This feature also ensures compatibility between BFD Version 0 and Version 1.

BFD version detection behavior

All BFD sessions come up as Version 1 by default and are interoperable with Version 0. The system automatically performs BFD version detection. BFD sessions between neighbors run in the highest common BFD version. For example, if one BFD neighbor is running BFD Version 0 and the other BFD neighbor is running Version 1, the session runs BFD Version 0. The output from the **show BFD neighbors** [**details**] command verifies which BFD version a BFD neighbor is running.

BFD support for nonbroadcast media interfaces
Describes BFD support for nonbroadcast media interfaces including routed, SVI, and L3 port channels, requiring the BFD interval command configuration on the interface to initiate BFD monitoring.

The BFD feature is supported on routed, SVI and L3 portchannels. The **bfd interval** command must be configured on the interface to initiate BFD monitoring.

Bidirectional Forwarding Detection configuration

Explains configurational information about bidirectional forwarding detection.

Bidirectional forwarding detection configuration is a network protocol setup that enables rapid failure detection between adjacent forwarding engines.

Configurational information

The sections provide configurational information about bidirectional forwarding detection.

Configure BFD session parameters on the interface

Configure BFD session parameters on the interface.

Configure baseline BFD session parameters to enable BFD on an interface for monitoring network connectivity to BFD neighbors.

To configure BFD on an interface, you must set the baseline BFD session parameters. Repeat the steps in this procedure for each interface over which you want to run BFD sessions to BFD neighbors.

The following procedure shows BFD configuration steps for a physical interface. Please use the corresponding BFD timer values for SVIs and ether-channels respectively.

1. Use the **configure terminal** command to enter global configuration mode.

```
Device# configure terminal
```

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

```
Device(config)# interface gigabitethernet 1/1
```

3. Use the **no switchport** command to disable the switchport.

```
Device(config-if)# no switchport
```

4. Perform one of the following steps:
   - Use the **ip address** *ipv4-address* | *mask* command to configure an IPv4 address for the interface.

```
Device(config-if)# ip address 10.201.201.1 255.255.255.0
```

   - Use the **ipv6 address** *ipv6-address*/*mask* command to configure an IPv6 address for the interface.

```
Device(config-if)# ipv6 address 2001:db8:1:1::1/32
```

5. Use the **bfd interval** *milliseconds* **min_rx** *milliseconds* **multiplier** *interval-multiplier* command to enable BFD on the interface.

```
Device(config-if)# bfd interval 100 min_rx 100 multiplier 3
```

The BFD interval configuration is removed when the subinterface on which it is configured is removed.

The BFD interval configuration is not removed when:

- An interface removes an IPv4 address.
- An interface removes an IPv6 address is removed from an interface.
- An interface disables IPv6.
- An interface is shutdown
- An interface globally or locally disables IPv4 CEF.
- An interface globally or locally disables IPv6 CEF.

6. Use the **end** command to return to privileged EXEC mode.

```
Device(config-if)# end
```

BFD support for OSPF configuration

Describes the procedures for configuring BFD support for OSPF so that OSPF is a registered protocol with BFD and will receive forwarding path detection failure messages from BFD.

BFD support for OSPF configuration is a networking procedure that

- enables OSPF as a registered protocol with BFD to receive forwarding path detection failure messages
- can be configured globally on all interfaces or selectively on one or more interfaces, and
- provides two methods for enabling BFD support for OSPF routing interfaces.

Configuration methods

There are two methods for enabling BFD support for OSPF:

- You can enable BFD for all of the interfaces for which OSPF is routing by using the **bfd all-interfaces** command in router OSPF configuration mode. You can disable BFD support on individual interfaces using the **ip ospf bfd** [**disable**] command in interface configuration mode.
- You can enable BFD for a subset of the interfaces for which OSPF is routing by using the **ip ospf bfd** command in interface configuration mode.

Configure BFD support for OSPF for all interfaces

Configure BFD support globally on all interfaces associated with an OSPF routing process to enable faster failure detection.

This task configures BFD (Bidirectional Forwarding Detection) support globally on all interfaces associated with an OSPF routing process to enable faster failure detection and convergence.

To configure BFD for all OSPF interfaces, perform the steps in this section.

If you do not want to configure BFD on all OSPF interfaces and would rather configure BFD support specifically for one or more interfaces, see the "Configuring BFD Support for OSPF for One or More Interfaces" section.

- OSPF must be running on all participating routers.
- The baseline parameters for BFD sessions on the interfaces over which you want to run BFD sessions to BFD neighbors must be configured. See the "Configuring BFD Session Parameters on the Interface" section for more information.

Follow these steps to configure BFD support for OSPF for all interfaces:

1. Use the **configure terminal** command to enter global configuration mode.

```
Device#configure terminal
```

2. Use the **router ospf** *process-id* command to specify an OSPF process and enter the router configuration mode.

```
Device(config)# router ospf 4
```

3. Use the **bfd all-interfaces** command to enable BFD globally on all interfaces associated with the OSPF routing process.

```
Device(config-router)# bfd all-interfaces
```

4. (Optional) Use the **exit** command to return the device to global configuration mode. Enter this command only if you want to perform the Step to disable BFD for one or more interfaces.

```
Device(config-router)# exit
```

5. (Optional) Use the **interface** *type number* command to enter interface configuration mode. Enter this command only if you want to perform the Step to disable BFD for one or more interfaces.

```
Device(config)# interface gig 1/1
```

6. (Optional) Use the **ip ospf bfd disable** command to disable BFD on a per-interface basis for one or more interfaces associated with the OSPF routing process.

```
Device(config-if)# ip ospf bfd disable
```

> **Note**
>
> You should use the **disable** keyword only if you enabled BFD on all of the interfaces that OSPF is associated with using the **bfd all-interfaces** command in router configuration mode.

7. Use the **end** command to exit nterface configuration mode and return the router to privileged EXEC mode.

```
Device(config-if)# end
```

8. (Optional) Use the **show bfd neighbors details** command to view information that can help verify if the BFD neighbor is active and displays the routing protocols that BFD has registered.

```
Device#show bfd neighbors detail
```

9. (Optional) Use the **show ip ospf** command to view information that can help verify if BFD for OSPF has been enabled.

```
Device#show ip ospf
```

Configure OSPF support for BFD over IPv4 for one or more interfaces

Configure BFD on one or more OSPF interfaces to enable bidirectional forwarding detection.

This task enables BFD (Bidirectional Forwarding Detection) on OSPF interfaces to provide fast failure detection and improve network convergence times.

Use this procedure when you need to configure BFD support for OSPF on specific interfaces in your network topology.

Follow these steps to configure OSPF support for BFD over IPv4 for one or more interfaces:

1. Use the **configure terminal** command to enter global configuration mode.

```
Device#configure terminal
```

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

```
Device(config)# interface gig 1/1
```

3. Use the **ip ospf bfd disable** command to disable BFD on a per-interface basis for one or more interfaces associated with the OSPF routing process.

```
Device(config-if)# ip ospf bfd
```

> **Note**
>
> Use the **disable** keyword only if you enable BFD on all of the interfaces that OSPF is associated with using the **BFD all-interfaces** command in router configuration mode.

4. Use the **end** command to exit nterface configuration mode and return the router to privileged EXEC mode.

```
Device(config-if)# end
```

5. (Optional) Use the **show bfd neighbors details** command to view information that can help verify if the BFD neighbor is active and displays the routing protocols that BFD has registered.

```
Device#show bfd neighbors detail
```

> **Note**
>
> If hardware-offloaded BFD sessions are configured with Tx and Rx intervals that are not multiples of 50 ms, the hardware intervals are changed. However, output from the **show BFD neighbors details** command displays only the configured intervals, not the interval values that change.

6. (Optional) Use the **show ip ospf** command to view information that can help verify if BFD for OSPF has been enabled.

```
Device#show ip ospf
```

Configure BFD support for static routing

Configure BFD support for static routing to enable faster convergence and failure detection.

This task configures BFD support for static routing to provide faster detection of path failures and improve network convergence times.

Perform this task to configure BFD support for static routing. Repeat the steps in this procedure on each BFD neighbor. For more information, see the "Example: Configuring BFD Support for Static Routing" section.

Follow these steps to configure BFD support for static routing:

1. Use the **configure terminal** command to enter global configuration mode.

```
Device#configure terminal
```

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

```
Device(config)# interface gig 1/1
```

3. Use the **interface** *type number* command to disable the switchport.

```
Device(config-if)# no switchport
```

4. Perform one of the following steps:
   - Use the **ip address** *ipv4-address mask* command to configure an IPv4 address for the interface.
   - Use the **ipv6 address** *ipv6-address/mask* command to configure an IPv6 address for the interface.

   Configuring an IPv4 address for the interface:

```
Device(config-if)#ip address 10.201.201.1 255.255.255.0
```

   Configuring an IPv6 address for the interface:

```
Device(config-if)#ipv6 address 2001:db8:1:1::1/32
```

5. Use the **BFD interval** *milliseconds* **min_rx** *milliseconds* **multiplier** *interval-multiplier* command to enable BFD on the interface.

```
Device(config-if)# bfd interval 100 min_rx 100 multiplier 3
```

   The BFD interval configuration is removed when the subinterface on which it is configured is removed.

   The BFD interval configuration is not removed when:
   - An interface removes an IPv4 address.
   - An interface removes an IPv6 address is removed from an interface.
   - An interface disables IPv6.
   - An interface is shutdown
   - An interface globally or locally disables IPv4 CEF.
   - An interface globally or locally disables IPv6 CEF.

6. Use the **exit** command to return to global configuration mode.

```
Device(config-if)# exit
```

7. Use the **IP route static BFD** *interface-type interface-number IP-address* **group** *group-name* **passive**]] command to specify a static route BFD neighbor.

```
Device(config)# ip route static bfd TenGigabitEthernet1/0/1 10.10.10.2 group
 group1 passive
```

> **Note**
>
> The *interface-type*, *interface-number*, and *IP-address* arguments are required because BFD support exists only for directly connected neighbors.

8. Use the **IP route vrf** *vrf-name prefix mask IP-address interface-type interface-number IP-address* **dhcp** *distance* **name** *next-hop-name* **permanent track** *number* **tag** *tag*] command to specify a static route BFD neighbor.

```
Device(config)#ip route 10.0.0.0 255.0.0.0
```

9. Use the **exit** command to exit global configuration mode and returns to privileged EXEC mode.

```
Device(config)#exit
```

Use the **show ip static route** command to view static route database information.

Use the **show ip static route bfd** command to view information about the static BFD configuration from the configured BFD groups and nongroup entries.

BFD echo mode

Describes a BFD feature that enables faster failure detection by using echo packets forwarded through the remote system.

BFD echo mode is a feature that sends echo packets through the forwarding engine. These packets are then returned along the same path for detection. It reduces the number of BFD control packets sent between BFD neighbors and enables faster failure detection times compared to BFD Version 0 with BFD control packets.

BFD echo mode characteristics

BFD echo mode is enabled by default, but you can disable it such that it can run independently in each direction.

BFD echo mode works with asynchronous BFD. The BFD session at the other end does not participate in the actual forwarding of the echo packets. The echo function and the forwarding engine are responsible for the detection process.

Because the forwarding engine tests the forwarding path on the remote (neighbor) system without involving the remote system, it is possible to improve the interpacket delay variance. This improvement can achieve quicker failure detection times compared to using BFD Version 0 with BFD control packets for the BFD session.

Echo mode is described as without asymmetry when it is running on both sides (both BFD neighbors are running echo mode).

Prerequisites for BFD configuration

Outlines the mandatory requirements you must meet before configuring BFD sessions and echo mode.

Ensure that BFD is running on all participating devices and that proper configuration prerequisites are met before implementing BFD sessions.

- BFD must be running on all participating devices.

- Before using BFD echo mode, you must disable the sending of Internet Control Message Protocol (ICMP) redirect messages by entering the **no ip redirects** command, in order to avoid high CPU utilization.
- The baseline parameters for BFD sessions on the interfaces over which you want to run BFD sessions to BFD neighbors must be configured. See the Configuring BFD Session Parameters on the Interface section for more information.

Disable BFD echo mode without asymmetry

Configure BFD settings on an interface and disable BFD echo mode to prevent asymmetric configurations.

This task disables BFD echo mode on an interface to avoid asymmetric BFD configurations that can cause detection issues.

BFD is enabled on an interface only after the "BFD interval" command is configured. There is no global command to configure BFD echo mode and it is configured only under the interface context.

1. Use the **configure terminal** command to enter global configuration mode.

```
Device#configure terminal
```

2. Use the **interface** *type number* command to configure an interface and enter interface configuration mode.

```
Device(config)# interface gigabitEthernet 1/1
```

3. Use the **no switchport** command to disable the switchport.

```
Device(config-if)# no switchport
```

4. Use the **bfd interval** *milliseconds* **min_rx** *milliseconds* **multiplier** *interval-multiplier* command to enable BFD on the interface.

```
Device(config-if)# bfd interval 100 min_rx 100 multiplier 3
```

5. Use the **bfd echo** command to enable the BFD echo.

```
Device(config-if)# bfd echo
```

You can use the **no bfd echo** command to disable the BFD echo mode.

BFD templates

Describes BFD templates that specify interval values for single-hop configurations.

A BFD template is a configuration template that specifies a set of BFD interval values for single-hop configurations. It provides BFD interval values that are not specific to a single interface, and disables echo mode when configured.

Note that configuring BFD-template will disable echo mode.

Configure a single-hop template

Configure a BFD single-hop template and BFD interval timers.

This task creates a BFD single-hop template that allows you to configure BFD interval timers for monitoring network connectivity.

BFD (Bidirectional Forwarding Detection) single-hop templates provide a way to standardize BFD configuration parameters across multiple interfaces or sessions. Use this template when you need to configure consistent BFD timing parameters for single-hop scenarios.

1. Use the **configure terminal** command to enter global configuration mode.

```
Device# configure terminal
```

2. Use the **BFD-template single-hop** *template-name* command to create a single-hop BFD template and enters BFD configuration mode.

```
Device(config)# bfd-template single-hop bfdtemplate1
```

3. Use the **intervalmin_tx** *milliseconds***min_rx** *milliseconds* **multiplier** *interval-multiplier* command to configure the transmit and receive intervals between BFD packets, and the number of consecutive BFD control packets that must be missed before BFD declares that a peer is unavailable

```
Device(bfd-config)# interval min-tx 120 min-rx 100 multiplier 3
```

4. Use the **end** comand to exit BFD configuration mode and returns the device to privileged EXEC mode.

```
Device(bfd-config)# end
```

Monitor and troubleshoot BFD

Retrieves BFD information for system maintenance and diagnostic purposes. These commands are entered as needed to support network troubleshooting.

This information applies to monitoring and troubleshooting BFD for Cisco platforms.

1. Use the **show bfd neighbors** [**details**] command to displays the BFD adjacency database.

```
Device# show bfd neighbors details
```

The **details** keyword shows all BFD protocol parameters and timers per neighbor.

2. Use the **debug bfd** [**packet** | **event**] command to displays debugging information about BFD packets.

```
Device# debug bfd packet
```

2 Configure IPv4 Policy-Based Routing

Topics:

Policy-Based Routing

Describes a routing technique that makes forwarding decisions based on configured policies rather than destination IP addresses.

Policy-based routing (PBR) is a routing technique that

- makes routing decisions based on configured policies
- forwards packets based on criteria other than destination IP address, such as source IP address
- permits routing of packets from different sources to different networks even when destinations are the same, and
- classifies traffic using access control lists (ACLs) and route maps to direct traffic through different paths.

How PBR works

When a router or switch receives a packet, it uses the packet's destination IP address to look up an entry in a routing table and make a forwarding decision. However, in some cases, you may need to forward the packet based on other criteria, such as the source IP address instead of the destination IP address. This approach lets you route packets from different sources to different networks, even if the destinations are the same. It is useful when interconnecting several private networks.

With PBR, you classify traffic using access control lists (ACLs) and direct it through a different path. The system applies PBR to incoming packets. Packets arriving on an interface with PBR enabled are processed by route maps. Route maps forward packets to the appropriate next hop based on defined criteria.

Route map statement processing rules:

- Route map statement marked as permit is processed as described:
  - A match command can match on multiple ACLs. A route map statement can contain multiple match commands. Logical or algorithm function is performed across all the match commands to reach a permit or deny decision.

    For example:

    ```
    match IP address acl1 acl2
    match IP address acl3
    ```

    Note

    IPv6 is not supported.

    A packet is permitted if it is permitted by acl1 or acl2 or acl3.
  - If the decision reached is permit, then the action specified by the **set** command is applied on the packet.
  - If the decision reached is deny, then the PBR action (specified in the **set** command) is not applied. Instead the processing logic moves forward to look at the next route-map statement in the sequence (the statement with the next higher sequence number). If no next statement exists, PBR processing terminates, and the packet is routed using the default IP routing table.
- For PBR, route-map statements and ACLs marked as deny are not supported.

You can use standard IP ACLs to specify match criteria for a source address or extended IP ACLs to specify match criteria based on an end station. The process proceeds through the route map until a match is found. If no match is found, normal destination-based routing occurs. There is an implicit deny at the end of the list of match statements.

If match clauses are satisfied, you can use a set clause to specify the IP addresses identifying the next hop router in the path. You can also set an IP precedence value using the precedence number or name.

Restrictions and limitations for PBR

Consider these restrictions and limitations when implementing Policy-Based Routing on your switch.
License and default state requirements

- To use Policy-Based Routing (PBR) , you must have the Network Advantage license enabled on the switch.
- By default, PBR is disabled on the switch. PBR becomes enabled when you configure and apply a route-map to an interface.
- You can enable PBR on a routed port or a Switch Virtual Interface (SVI).
- Packets that are generated by the switch (CPU), or local packets, are not normally policy-routed. When you globally enable local PBR on the switch, all unicast packets that originate on the switch are subject to local PBR. The protocols that are supported for local PBR are NTP, DNS, MSDP, SYSLOG and TFTP. Local PBR is DISABLED by default.

Supported traffic types and local PBR

- PBR applies only to unicast traffic. Multicast traffic is not policy-routed.
- Packets that are generated by the switch (CPU), or local packets, are not normally policy-routed. When you globally enable local PBR on the switch, all unicast packets that originate on the switch are subject to local PBR. Local PBR supports the following protocols: NTP, DNS, MSDP, SYSLOG, and TFTP. Local PBR is disabled by default.

Scale and hardware resource limits

- You can define a maximum of 64 IP policy route maps on the switch.
- You can define a maximum of 64 access control entries (ACEs) for PBR on the switch.
- The number of hardware entries used by PBR depends on the route map itself, the ACLs used, and the order of the ACLs and route-map entries. The maximum number of entries is 256.

Mutual exclusivity restrictions

- When a policy route map is applied to a physical interface, that interface cannot become a member of an EtherChannel.
- VRF and PBR are mutually exclusive on a switch interface. You cannot enable VRF when PBR is enabled on an interface, and conversely, you cannot enable PBR when VRF is enabled on an interface.
- Web Cache Communication Protocol (WCCP) and PBR are mutually exclusive on a switch interface. You cannot enable WCCP when PBR is enabled on an interface, and conversely, you cannot enable PBR when WCCP is enabled on an interface.

Precedence and feature coexistence

- When a policy route map is applied on an interface along with ACL and QoS, ACL and QoS have higher precedence.
- IP Source Guard has a higher precedence if a rule matches the PBR rule on the same interface.
- On an SVI, IP Source Guard and PBR rules are merged based on operational requirements.

Route-map configuration rules

- Policy-maps with no **set** actions are supported. Matching packets are routed normally.

- Policy-maps with no **match** clauses are supported. **Set** actions are applied to all packets.
- Only one **set** clause is supported at a time in a single sequence route-map. In a route-map with multiple sequences, only **set** clauses of the same type are allowed. For example, if **set IP next-hop** is used in the first sequence, then the second sequence should also have the same set clause **set IP next-hop**.

Limitations and unsupported features

- The switch does not support route map **deny** statements for PBR.
- Matching ACLs with **deny** ACEs is not supported.
- The switch does not support PBR based on Type of Service (TOS), DSCP, or IP Precedence.
- The **set interface**, **set default next-hop**, and **set default interface** actions are not supported.
- The **ip next-hop recursive** and **ip next-hop verify availability** features are not available; the next-hop must be directly connected.

PBR supported ACL match field options

This table summarizes the PBR support for ACL match field options on the switch.

Table 1: PBR Supported ACL Match field Options

Match Field	Supported (Y/N)
Source IP address	Y
Destination IP address	Y
Next Header (ICMP, IGMP, etc.)	N
TCP/UDP Port	N
Type of Service (TOS)	N
Fragmentation Bit	N

PBR feature support

This table lists the PBR feature support on the switch.

Table 2: PBR Feature Support

Feature	Support/Scale
PBR on Ingress Traffic	Y
PBR on Egress Traffic	N
PBR on Physical Interface (L2 Port)	N
PBR on Physical Interface (Routed Port)	Y
PBR on SVI Interface	Y
PBR on Port Channel (L2)	N
PBR on Port Channel (L3)	N
PBR with VRF	N

Feature	Support/Scale
Match on IPv4 ACL	Y **Note** Refer to table above for PBR Supported ACL Match field Options.
Match on Extended/Standard IPv4 ACL	Y
Match Based on Packet Length	N
Match with DENY ACE	N
Action Set Fragment Bit	N
Action to Set Precedence	N
Action to Set Next-Hop	Y
Recursive Next-Hop Action	N
Action to Set Interface	N
Action to Set Default Interface	N
Action to Set IP Precedence	Y
Action to Set IP VRF	Y
Set IP default Next-Hop	N
Set IP Default VRF	N
PBR on Multicast Traffic	N
PBR on IPv6 Traffic	N
Route-Map Deny	N
MAX number of route-map supported	64
MAX number of ACL policies supported	64
Local PBR	Y

Configure PBR

Configure Policy-Based Routing (PBR) to control where packets are output based on defined criteria and actions.

Configure PBR to enable policy-based routing on the switch. This feature allows you to control packet forwarding based on specific match criteria and routing policies.

By default, PBR is disabled on the switch. To enable PBR, you must create a route map that specifies the match criteria and the resulting action. Then, enable PBR for that route map on an interface. Packets that arrive on the specified interface and match the match clauses are subject to PBR.

1. Use the **configure terminal** command to enter global configuration mode.

```
Device# configure terminal
```

2. Use the **route-map** *map-tag* [**permit**] [*sequence number*] command to define route maps that are used to control where packets are output, and enters route-map configuration mode.

```
Device(config)# route-map pbr-map permit
```

   - *map-tag* A meaningful name for the route map. The **IP policy route-map**
   - (Optional) **permit** If **permit** is specified and the match criteria are met for this route map, the route is policy routed as defined by the set actions.
   - (Optional) *sequence number* The sequence number shows the position of the route-map statement in the given route map.

3. Use the **match IP address** {*access-list-number* | *access-list-name*} [*access-list-number* |...*access-list-name*] command to match the source and destination IP addresses that are permitted by one or more standard or extended access lists. ACLs can match multiple source and destination IP addresses.

```
Device(config-route-map)# match ip address 110 140
```

   If you do not specify a **match** command, the route map is applicable to all packets.

4. Configure the actions within the policy route map to define next-hop routing, VRF interfaces, or IP header precedence values for matching traffic.
   a) Use the **set ip next-hop** *ip-address* [...*IP-address*] command to specify the action taken on packets that match the criteria. This command sets the next hop to which to route the packet. The next hop must directly neighbor the switch.

```
Device(config-route-map)# set ip next-hop 10.1.6.2
```

   b) Use the **set ip vrf** *vrf-name***next-hop** *IP-address* [...*ip-address*] command to apply policy-based routing to a VRF interface.

```
Device(config-route-map)# set ip vrf myvrf next-hop 10.5.5.5
```

   c) Use the **set ip precedence** [*number* / *name*] command to set the precedence value in the IP header.

```
Device(config-route-map)# set ip precedence 5
```

      - 0–Routine
      - 1–Priority
      - 2–Immediate
      - 3–Flash
      - 4–Flash-override
      - 5–Critical
      - 6–Internet

- 7—Network

Use the **exit** command to return to global configuration mode.

5. Use the **interface***interface-id* command to enter interface configuration mode, and specify the interface to be configured.

```
Device(config)# interface gigabitethernet 1/1
```

6. Use the **ip policy route-map***map-tag* command to enable PBR on a Layer 3 interface, and identify the route map to use.

```
Device(config-if)# ip policy route-map pbr-map
```

You can configure only one route map on an interface. However, you can have multiple route map entries with different sequence numbers. These entries are evaluated in the order of sequence number until the first match. If there is no match, packets are routed as usual.

Use the **exit** command to return to global configuration mode.

7. (Optional) Use the **ip local policy route-map** *map-tag*command to enable local PBR to perform policy-based routing on packets originating at the switch.

```
Device(config)# ip local policy route-map local-pbr
```

This applies only to packets generated by the switch, not incoming packets.

Use the **end** command to return to privileged EXEC mode.

8. (Optional) Use the **show** commands to verify the configuration.
   a) Use the **show route-map** [*map-name*] command to view all the route maps configured or only the one specified to verify configuration.

```
Device# show route-map
```

   b) Use the **show ip policy** command to view policy route maps attached to the interface.

```
Device# show ip policy
```

   c) Use the **show ip local policy** command to view whether or not local policy routing is enabled and, if so, the route map being used.

```
Device# show ip local policy
```

3 Configure IPv6 Unicast Routing

Topics:

IPv6 unicast routing

Describes how to configure IPv6 unicast routing on the switch.

IPv6 unicast routing is a network routing mechanism that enables switches to forward IPv6 packets between networks using unicast addresses.

IPv6

Describes the Internet Protocol version 6 and its benefits for network addressing and services.

IPv6 is an Internet Protocol that

- provides end-to-end security, quality of service (QoS), and globally unique addresses
- reduces the need for private addresses and Network Address Translation (NAT) processing by border routers at network edges, and
- enables IPv4 users to move to IPv6 and receive enhanced services.

Additional resources

For information about how Cisco Systems implements IPv6, go to:

http://www.cisco.com/en/US/products/ps6553/products_ios_technology_home.html

For information about IPv6 and other features in this chapter:

- See the *Cisco IOS IPv6 Configuration Library*.
- Use the Search field on Cisco.com to locate the Cisco IOS software documentation. For example, if you want information about static routes, you can enter *Implementing Static Routes for IPv6* in the search field to learn about static routes.

IPv6 addresses

Describes the IPv6 addressing format and representation supported by the switch.

An IPv6 address is a 128-bit network identifier that

- uses eight 16-bit hexadecimal fields separated by colons
- supports unicast addressing only on the switch, and
- allows optional leading zero omission and successive zero field compression.

IPv6 address format and representation

The switch supports only IPv6 unicast addresses. It does not support site-local unicast addresses, or anycast addresses.

The IPv6 128-bit addresses are represented as a series of eight 16-bit hexadecimal fields separated by colons in the format: n:n:n:n:n:n:n:n. This is an example of an IPv6 address:

```
2031:0000:130F:0000:0000:09C0:080F:130B
```

For easier implementation, leading zeros in each field are optional. This is the same address without leading zeros:

```
2031:0:130F:0:0:9C0:80F:130B
```

You can also use two colons (::) to represent successive hexadecimal fields of zeros, but you can use this short version only once in each address:

```
2031:0:130F::09C0:080F:130B
```

For more information about IPv6 address formats, address types, and the IPv6 packet header, see the http://www.cisco.com/c/en/us/td/docs/ios-xml/ios/ipv6_basic/configuration/xe-3e/ip6b-xe-3e-book.html of *Cisco IOS IPv6 Configuration Library* on Cisco.com.

- IPv6 Address Formats
- IPv6 Address Type: Multicast
- IPv6 Address Output Display
- Simplified IPv6 Packet Header

Supported IPv6 unicast routing features

Describes the IPv6 protocol features supported by the switch.

These sections describe the IPv6 protocol features supported by the switch:

128-bit wide unicast addresses

Describes the types of IPv6 unicast addresses supported by the switch and their characteristics.

A 128-bit wide unicast address is an IPv6 addressing scheme. It supports aggregatable global unicast addresses and link-local unicast addresses. This scheme enables strict aggregation of routing prefixes, limits routing table entries, and does not support site-local unicast addresses.

Address types and characteristics

The switch supports two types of 128-bit wide unicast addresses:

- **Aggregatable global unicast addresses:** IPv6 addresses from the aggregatable global unicast prefix. The address structure enables strict aggregation of routing prefixes. It also limits the number of entries in the global routing table. These addresses are used on links that are aggregated through organizations and eventually to the Internet service provider. These addresses are defined by a global routing prefix, a subnet ID, and an interface ID. Current global unicast address allocation uses the range of addresses that start with binary value 001 (2000::/3). Addresses with a prefix of 2000::/3(001) through E000::/3(111) must have 64-bit interface identifiers in the extended unique identifier (EUI)-64 format.
- **Link local unicast addresses:** Can be automatically configured on any interface by using the link-local prefix FE80::/10(1111 1110 10) and the interface identifier in the modified EUI format. Link-local addresses are used in the neighbor discovery protocol (NDP) and the stateless autoconfiguration process. Nodes on a local link use link-local addresses and do not require globally unique addresses to communicate. IPv6 routers do not forward packets with link-local source or destination addresses to other links.

For more information, see the section about IPv6 unicast addresses in the "Implementing IPv6 Addressing and Basic Connectivity" chapter in the *Cisco IOS IPv6 Configuration Library* on Cisco.com.

DNS for IPv6

Describes DNS record types that support IPv6 addresses in name-to-address and address-to-name lookup processes.

DNS for IPv6 is a Domain Name System implementation. It supports DNS record types in IPv6 name-to-address and address-to-name lookup processes. It uses DNS AAAA resource record types, which are equivalent to A address records in IPv4. DNS for IPv6 enables switches to support DNS resolution for both IPv4 and IPv6.

Path MTU discovery for IPv6 unicast

Describes how switches support advertising the system maximum transmission unit to IPv6 nodes and enable dynamic path MTU discovery.

Path MTU discovery for IPv6 unicast is a network mechanism. It allows a host to dynamically discover and adjust to differences in MTU size of each link along the data path. This mechanism enables switches to advertise the system maximum transmission unit (MTU) to IPv6 nodes. The packet source must handle fragmentation when a link along the path cannot accommodate the packet size.

ICMPv6

Describes the Internet Control Message Protocol version 6 and its functions in IPv6 networks.

ICMPv6 is an Internet Control Message Protocol used in IPv6 networks. It generates error messages, such as ICMP destination unreachable messages, to report errors during processing. ICMPv6 also provides diagnostic functions and is used in the neighbor discovery protocol and path MTU discovery.

Neighbor discovery

Describes the IPv6 neighbor discovery process that uses ICMP messages and solicited-node multicast addresses to determine link-layer addresses and verify neighbor reachability.

Neighbor discovery is an IPv6 protocol that runs on top of ICMPv6. It uses ICMP messages and solicited-node multicast addresses to determine the link-layer address of a neighbor on the same network and to verify reachability. The protocol also keeps track of neighboring routers.

Neighbor discovery features

The switch supports NDP for IPv6 and static neighbor entries for IPv6 stations that do not support NDP.

The switch supports ICMPv6 redirect for routes with mask lengths less than 64 bits. ICMP redirect is not supported for host routes or for summarized routes with mask lengths greater than 64 bits.

Neighbor discovery throttling ensures that the switch CPU is not unnecessarily burdened while it is in the process of obtaining the next hop forwarding information to route an IPv6 packet. The switch drops any additional IPv6 packets whose next hop is the same neighbor that the switch is actively trying to resolve. This drop avoids further load on the CPU.

Default router preference

Describes an IPv6 extension that improves host router selection in multihomed environments.

Default router preference (DRP) is an IPv6 extension in router advertisement messages that improves the ability of a host to select an appropriate router. It is especially useful when the host is multihomed and the routers are on different links. DRP allows configuration of an IPv6 host to prefer one router over another, provided both are reachable or probably reachable.

How default router preference works

An IPv6 host maintains a default router list from which it selects a router for traffic to offlink destinations. The selected router for a destination is then cached in the destination cache. NDP for IPv6 specifies that routers that are reachable or probably reachable are preferred over routers whose reachability is unknown or suspect. For reachable or probably reachable routers, NDP can either select the same router every time or cycle through the router list.

The switch supports IPv6 default router preference (DRP), an extension in router advertisement messages. The switch does not support the Route Information Option in RFC 4191.

For configuring DRP for IPv6, see the *Configuring Default Router Preference* section.

For more information about DRP for IPv6, see the *Cisco IOS IPv6 Configuration Library* on Cisco.com.

IPv6 stateless autoconfiguration and duplicate address detection

Describes how the switch autonomously manages addressing changes and configures interfaces using stateless autoconfiguration.

Pv6 stateless autoconfiguration and duplicate address detection is a network mechanism. It enables the switch to autonomously manage link, subnet, and site addressing changes. It also allows hosts to configure their own link-local addresses. Additionally, it facilitates booting nodes to send router solicitations to request router advertisements for configuring interfaces.

Additional information

For more information about autoconfiguration and duplicate address detection, see the "Implementing IPv6 Addressing and Basic Connectivity" chapter of *Cisco IOS IPv6 Configuration Library* on Cisco.com.

IPv6 applications
Lists the applications for which IPv6 is supported.

The switch has IPv6 support for these applications:

- Ping, traceroute, Telnet, and TFTP
- Secure Shell (SSH) over an IPv6 transport
- HTTP server access over IPv6 transport
- DNS resolver for AAAA over IPv4 transport
- Cisco Discovery Protocol (CDP) support for IPv6 addresses

For more information about managing these applications, see the *Cisco IOS IPv6 Configuration Library* on Cisco.com.

DHCP for IPv6 address assignment
Describes a Dynamic Host Configuration Protocol extension that enables automatic IPv6 address assignment and configuration parameter distribution to network clients.

DHCPv6 is a network protocol that enables DHCP servers to pass configuration parameters, such as IPv6 network addresses, to IPv6 clients. It manages non-duplicate address assignment within the correct prefix based on the network where the host is connected. DHCPv6 also assigns addresses from one or multiple prefix pools and passes additional options, such as the default domain and DNS name-server address, to the client.

Address pool assignment options

Address pools can be configured for use in different deployment scenarios:

- On a specific interface
- On multiple interfaces
- Automatic pool selection by the server

For configuring DHCP for IPv6, see the *Configuring DHCP for IPv6 Address Assignment* section.

For more information about configuring the DHCPv6 client, server, or relay agent functions, see the *Cisco IOS IPv6 Configuration Library* on Cisco.com.

Static routes for IPv6
Explains manually configured routes that define explicit paths between networking devices for IPv6 networks.

A static route for IPv6 is a manually configured network path that defines an explicit route between two networking devices. It provides routing for smaller networks with only one path to an outside network. It also helps secure specific types of traffic in larger networks.

Configuration information

For configuring static routes for IPv6, see the *Configuring Static Routing for IPv6* section.

For more information about static routes, see the "Implementing Static Routes for IPv6" chapter in the *Cisco IOS IPv6 Configuration Library* on Cisco.com.

Policy-based routing for IPv6
Describes a flexible routing mechanism that allows configuration of defined policies for traffic flows, reducing reliance on routes derived from routing protocols.

Policy-based routing (PBR) for IPv6 is a flexible mechanism that allows configuration of defined policies for traffic flows. It reduces reliance on routes derived from routing protocols, provides more control by extending and complementing existing routing mechanisms, and allows you to set IPv6 precedence and specify paths for particular traffic, such as sending priority traffic over high-cost links

Implementation and capabilities

PBR for IPv6 may be applied to both forwarded and originated IPv6 packets. For forwarded packets, PBR for IPv6 is implemented as an IPv6 input interface feature, supported in these forwarding paths:

- Process
- Cisco Express Forwarding (formerly known as CEF)
- Distributed Cisco Express Forwarding

Policies can be based on the IPv6 address, port numbers, protocols, or packet size.

PBR allows you to perform these tasks:

- Classify traffic based on extended access list criteria. Access lists, then, establish the match criteria.
- Set IPv6 precedence bits, giving the network the ability to enable differentiated classes of service.
- Route packets to specific traffic-engineered paths. This may be necessary to ensure a specific quality of service (QoS) throughout the network.

PBR allows you to classify and mark packets at the edge of the network. PBR marks a packet by setting precedence value. The precedence value can be used directly by devices in the network core to apply the appropriate QoS to a packet, which keeps packet classification at your network edge.

For enabling PBR for IPv6, see the *Enabling Local PBR for IPv6* section.

For enabling IPv6 PBR for an interface, see the *Enabling IPv6 PBR on an Interface* section.

RIP for IPv6
Describes a distance-vector routing protocol that supports IPv6 addressing and uses hop count as a routing metric.

Routing Information Protocol (RIP) for IPv6 is a distance-vector protocol. It uses hop count as a routing metric, supports IPv6 addresses and prefixes, and sends RIP update messages to the all-RIP-routers multicast group address FF02::9.

Configuration and additional information

For configuring RIP for IPv6, see the *Configuring RIP for IPv6* section.

For more information about RIP for IPv6, see the "Implementing RIP for IPv6" chapter in the *Cisco IOS IPv6 Configuration Library* on Cisco.com.

OSPF for IPv6
Describes OSPF for IPv6 support on switches.

OSPF for IPv6 is a link-state protocol that the switch supports for IPv6 routing.

Configuration and additional information

For configuring OSPF for IPv6, see the *Configuring OSPF for IPv6* section.

For more information, see *Cisco IOS IPv6 Configuration Library* on Cisco.com.

EIGRP IPv6
Describes EIGRP IPv6 that runs on interfaces without requiring a global IPv6 address

EIGRP IPv6 is an Enhanced Interior Gateway Routing Protocol. It runs on interfaces without requiring a global IPv6 address. It is configured on the interfaces where it operates and supports stub routing on switches running Network Essentials.

Router ID requirements

Before running, an instance of EIGRP IPv6 requires an implicit or explicit router ID. An implicit router ID is derived from a local IPv6 address, so any IPv6 node always has an available router ID. However, EIGRP IPv6 might be running in a network with only IPv6 nodes and therefore might not have an available IPv6 router ID.

For configuring EIGRP for IPv6, see the *Configuring EIGRP for IPv6* section.

For more information about EIGRP for IPv6, see the *Cisco IOS IPv6 Configuration Library* on Cisco.com.

EIGRPv6 stub routing

Describes a network routing feature that reduces resource utilization by controlling route advertisements and queries between routers.

EIGRPv6 stub routing is a network routing feature that reduces resource utilization by moving routed traffic closer to the end user, allows only specified routes to be propagated from the stub router, and prevents the stub router from being queried for routes by neighbor routers.

EIGRPv6 stub routing configuration details

In a network using EIGRPv6 stub routing, the only allowable route for IPv6 traffic to the user is through a switch that is configured with EIGRPv6 stub routing. The switch sends the routed traffic to interfaces that are configured as user interfaces or are connected to other devices.

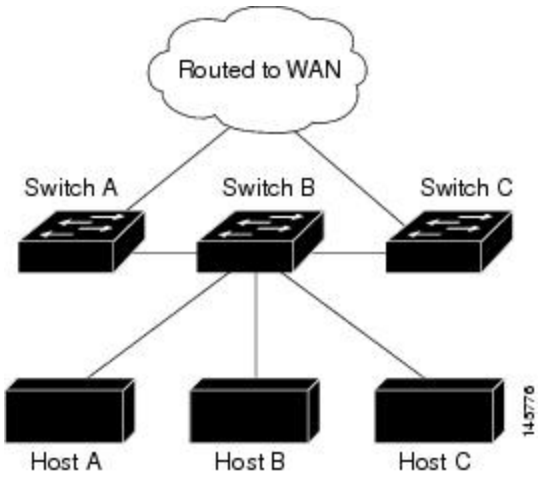
When using EIGRPv6 stub routing, you need to configure the distribution and remote routers to use EIGRPv6 and to configure only the switch as a stub. Only specified routes are propagated from the switch. The switch responds to all queries for summaries, connected routes, and routing updates.

Any neighbor that receives a packet informing it of the stub status does not query the stub router for any routes, and a router that has a stub peer does not query that peer. The stub router depends on the distribution router to send the proper updates to all peers.

EIGRPv6 stub router network topology

In this figure, switch B is configured as an EIGRPv6 stub router. Switches A and C are connected to the rest of the WAN. Switch B advertises connected, static, redistribution, and summary routes to switch A and C. Switch B does not advertise any routes learned from switch A (and the reverse).

Figure 3: EIGRP stub router configuration



For more information about EIGRPv6 stub routing, see "Implementing EIGRP for IPv6" section of the *Cisco IOS IP Configuration Guide, Volume 2 of 3: Routing Protocols, Release 12.4*.

SNMP and syslog over IPv6

Describes network management protocols that provide dual-stack support for IPv4 and IPv6 TRANSPORT mechanisms.

SNMP and syslog over IPv6 are network management protocols that support both IPv4 and IPv6 transport mechanisms. They provide IPv6 transport for SNMP, modify the SNMP agent to support traps for an IPv6 host, and include SNMP and syslog-related MIBs to support IPv6 addressing.

SNMP and syslog capabilities

To support both IPv4 and IPv6, network management requires both IPv6 and IPv4 transports. Syslog over IPv6 supports address data types for these transports.

SNMP and syslog over IPv6 provide these features:

- Support for both IPv4 and IPv6
- IPv6 transport for SNMP and to modify the SNMP agent to support traps for an IPv6 host
- SNMP- and syslog-related MIBs to support IPv6 addressing
- Configuration of IPv6 hosts as trap receivers

For support over IPv6, SNMP modifies the existing IP transport mapping to simultaneously support IPv4 and IPv6. These SNMP actions support IPv6 transport management:

- Opens User Datagram Protocol (UDP) SNMP socket with default settings
- Provides a new transport mechanism called *SR_IPV6_TRANSPORT*
- Sends SNMP notifications over IPv6 transport
- Supports SNMP-named access lists for IPv6 transport
- Supports SNMP proxy forwarding using IPv6 transport
- Verifies SNMP Manager feature works with IPv6 transport

For information on SNMP over IPv6, including configuration procedures, see the "Managing Cisco IOS Applications over IPv6" chapter in the *Cisco IOS IPv6 Configuration Library* on Cisco.com.

For information about syslog over IPv6, including configuration procedures, see the "Implementing IPv6 Addressing and Basic Connectivity" chapter in the *Cisco IOS IPv6 Configuration Library* on Cisco.com.

HTTP(S) over IPv6

Describes how HTTP clients and servers communicate over IPv6 networks using dual-stack environments.

HTTP(S) over IPv6 is a network communication protocol. It enables HTTP clients to send requests to both IPv4 and IPv6 HTTP servers. HTTP servers can also respond to requests from both IPv4 and IPv6 HTTP clients. URLs with literal IPv6 addresses must be specified in hexadecimal using 16-bit values separated by colons.

Network connectivity requirements

The accept socket call chooses an IPv4 or IPv6 address family. The accept socket is either an IPv4 or IPv6 socket. The listening socket continues to listen for both IPv4 and IPv6 signals that indicate a connection. The IPv6 listening socket is bound to an IPv6 wildcard address.

The underlying TCP/IP stack supports a dual-stack environment. HTTP relies on the TCP/IP stack and the sockets for processing network-layer interactions.

Basic network connectivity (**ping**) must exist between the client and the server hosts before HTTP connections can be made.

For more information, see the "Managing Cisco IOS Applications over IPv6" chapter in the *Cisco IOS IPv6 Configuration Library* on Cisco.com.

Unsupported IPv6 unicast routing features

Outlines the IPv6 unicast routing features that are not supported on the switch.

Do not use these IPv6 features as they are not supported on the switch:

- IPv6 packets that are destined to site-local addresses.
- Tunneling protocols, such as IPv4-to-IPv6 or IPv6-to-IPv4.
- The switch as a tunnel endpoint supporting IPv4-to-IPv6 or IPv6-to-IPv4 tunneling protocols.
- IPv6 Web Cache Communication Protocol (WCCP).

IPv6 feature limitations

Consider the IPv6 hardware implementation limitations that affect switch functionality and feature availability.

Because IPv6 is implemented in switch hardware, some limitations occur due to the IPv6 compressed addresses in the hardware memory. These hardware limitations result in some loss of functionality and limits some features.

- The switch cannot apply QoS classification on source-routed IPv6 packets in hardware.

IPv6 and switch stacks

Describes IPv6 forwarding across the stack and IPv6 host functionality on the active switch.

IPv6 and switch stacks provide a networking capability that supports IPv6 forwarding across the stack and enables IPv6 host functionality on the active switch. The active switch can run IPv6 unicast routing protocols and compute routing tables. Member switches can receive tables and create hardware IPv6 routes for forwarding.

IPv6 switch stack operations

The switch supports IPv6 forwarding across the stack and IPv6 host functionality on the active switch. The active switch runs the IPv6 unicast routing protocols and computes the routing tables. The member switches receive the tables and create hardware IPv6 routes for forwarding. The active switch also runs all IPv6 applications.

If a new switch becomes the active switch, it recomputes the IPv6 routing tables. Then, it distributes them to the member switches. During election and reset of the new active switch, the switch stack does not forward IPv6 packets. The stack MAC address changes. This change also updates the IPv6 address. When you specify the stack IPv6 address with an extended unique identifier (EUI) by using the **ipv6 address** *ipv6-prefix/prefix length* **EUI-64** interface configuration command, the address is based on the interface MAC address. See the *Configuring IPv6 Addressing and Enabling IPv6 Routing* section for more information.

If you configure the persistent MAC address feature on the stack and the active switch changes, the stack MAC address does not change for approximately 4 minutes.

These are the functions of the IPv6 active switch and members.

- Active switch:
  - runs IPv6 routing protocols
  - generates routing tables
  - distributes routing tables to member switches that use distributed Cisco Express Forwarding for IPv6
  - runs IPv6 host functionality and IPv6 applications
- Member switch:
  - receives Cisco Express Forwarding for IPv6 routing tables from the active switch
  - programs the routes into hardware

> **Note**
>
> IPv6 packets are routed in hardware across the stack if the packet does not have exceptions (IPv6 Options) and the switches in the stack have not run out of hardware resources.

- flushes the Cisco Express Forwarding for IPv6 tables on active switch re-election

Default IPv6 configuration

Lists the default IPv6 configuration settings for various features including SDM templates, routing, Cisco Express Forwarding, and addresses.

This reference provides the default IPv6 configuration settings for key features and components.

Table 3: Default IPv6 configuration

Feature	Default Setting
IPv6 routing	Disabled globally and on all interfaces **Note** IP routing is enabled by default, but IPv6 unicast-routing must be enabled by the user.
Cisco Express Forwarding for IPv6 or distributed Cisco Express Forwarding for IPv6	Disabled (IPv4 Cisco Express Forwarding and distributed Cisco Express Forwarding are enabled by default) **Note** When IPv6 routing is enabled, Cisco Express Forwarding for IPv6 and distributed Cisco Express Forwarding for IPv6 are automatically enabled.
IPv6 addresses	None configured

How to configure IPv6 unicast routing

Describes the various configuration options available for setting up IPv6 unicast routing on Cisco devices, providing detailed steps and procedures for implementation.

The following sections shows the various configuration options available for IPv6 Unicast Routing

Enable IPv6 Routing

Describes how to assign IPv6 addresses to individual Layer 3 interfaces.

This section describes how to assign IPv6 addresses to individual Layer 3 interfaces and to globally forward IPv6 traffic on the switch.

For more information about configuring IPv6 routing, see the “Implementing Addressing and Basic Connectivity for IPv6” chapter in the "Cisco IOS IPv6 Configuration Library" on Cisco.com.

To assign an IPv6 address to a Layer 3 interface and enable IPv6 routing, perform this procedure:

Before configuring IPv6 on the switch, consider these guidelines:

- Not all features discussed in this chapter are supported by the switch. See the Unsupported IPv6 Unicast Routing Features.
- In the **ipv6 address** interface configuration command, you must enter the *ipv6-address* and *ipv6-prefix* variables with the address specified in hexadecimal using 16-bit values between colons. The *prefix-length* variable (preceded by a slash [/]) is a decimal value that shows how many of the high-order contiguous bits of the address comprise the prefix (the network portion of the address).

To forward IPv6 traffic on an interface, you must configure a global IPv6 address on that interface. Configuring an IPv6 address on an interface automatically configures a link-local address and activates IPv6 for the interface. Configured interfaces automatically join these required multicast groups for the link:

- solicited-node multicast group FF02:0:0:0:0:1:ff00::/104 for each unicast address assigned to the interface (this address is used in the neighbor discovery process.)
- all-nodes link-local multicast group FF02::1
- all-routers link-local multicast group FF02::2

To remove an IPv6 address from an interface, use the **no ipv6 address** *ipv6-prefix/prefix length* **eui-64** or **no ipv6 address** *ipv6-address* **link-local** interface configuration command. To remove all manually configured IPv6 addresses from an interface, use the **no ipv6 address** interface configuration command without arguments. To disable IPv6 processing on an interface that has not been explicitly configured with an IPv6 address, use the **no ipv6 enable** interface configuration command. To globally disable IPv6 routing, use the **no ipv6 unicast-routing** global configuration command.

1. Use the **configure terminal** command to enter global configuration mode.

```
Device# configure terminal
```

2. Use the **interface** *interface-id* command to enter interface configuration mode, and specify the Layer 3 interface to configure. The interface can be a physical interface, a switch virtual interface (SVI), or a Layer 3 EtherChannel.

```
Device(config)# interface gigabitethernet 1/0/1
```

3. Use the **no switchport** command to remove the interface from Layer 2 configuration mode (if it is a physical interface).

```
Device(config-if)# no switchport
```

4. Use these commands to configure IPv6 addresses and enable IPv6 processing on the interface:
   - **ipv6 address** *ipv6-prefix/prefix length* **eui-64**

     This command specifies a global IPv6 address with an extended unique identifier (EUI) in the low-order 64 bits of the IPv6 address. Specify only the network prefix; the switch automatically computes the last 64 bits from the MAC address. IPv6 processing is then enabled on the interface.

- **ipv6 address** *ipv6-address/prefix length*

  This command manually configures an IPv6 address on the interface.

- **ipv6 address** *ipv6-address* **link-local**

  Specifies a link-local address on the interface to be used instead of the link-local address that is automatically configured when IPv6 is enabled on the interface. This command enables IPv6 processing on the interface.

- **ipv6 enable**
- **ipv6 address** *WORD*
- **ipv6 address** *autoconfig*

  Automatically configures an IPv6 link-local address on the interface, and enables the interface for IPv6 processing. The link-local address can only be used to communicate with nodes on the same link.

- **ipv6 address** *dhcp*

```
Device(config-if)# ipv6 address 2001:0DB8:c18:1::/64 eui 64
```

```
Device(config-if)# ipv6 address 2001:0DB8:c18:1::/64
```

```
Device(config-if)# ipv6 address 2001:0DB8:c18:1:: link-local
```

```
Device(config-if)# ipv6 enable
```

5. Use the **exit** command to return to global configuration mode.

```
Device(config-if)# exit
```

6. Use the **ipv6 unicast-routing** command to enable forwarding of IPv6 unicast data packets.

```
Device(config)# ipv6 unicast-routing
```

7. Use the **end** command to return to privileged EXEC mode.

```
Device(config)# end
```

8. (Optional) Use the **show ipv6 interface** *interface-id* command to verify the configuration.

```
Device# show ipv6 interface gigabitethernet 1/0/1
```

Configure IPv4 and IPv6 protocol stacks

Configure a Layer 3 interface to support both IPv4 and IPv6 and enable IPv6 routing.

This task configures a Layer 3 interface to support both IPv4 and IPv6 protocol stacks and enables IPv6 routing functionality on the switch.

Use this procedure when you need to configure dual-stack networking on a Layer 3 interface to support both IPv4 and IPv6 traffic.

Note

To disable IPv6 processing on an interface that has not been configured with an IPv6 address, use the **no ipv6 enable** interface configuration command.

Follow these steps to configure IPv4 and IPv6 protocol stacks:

1. Use the **configure terminal** command to enter global configuration mode.

```
Device# configure terminal
```

2. Use the **ip routing** command to enable routing on the switch.

```
Device(config)# ip routing
```

3. Use the **ipv6 unicast-routing** command to enable forwarding of IPv6 data packets on the switch.

```
Device(config)# ipv6 unicast-routing
```

4. Use the **interface** *interface-id* command to enter interface configuration mode, and specify the Layer 3 interface to configure.

```
Device(config)# interface gigabitethernet 1/0/1
```

5. Use the **no switchport** command to remove the interface from Layer 2 configuration mode (if it is a physical interface).

```
Device(config-if)# no switchport
```

6. Use the **ip address** *ip-address mask* [**secondary**] command to specify a primary or secondary IPv4 address for the interface.

```
Device(config-if)# ip address 10.1.2.3 255.255.255
```

7. Use one of these commands to configure an IPv6 address or enable IPv6 processing on the interface.
   - **ipv6 address** *ipv6-prefix/prefix length* **eui-64**
   - **ipv6 address** *ipv6-address/prefix length*
   - **ipv6 address** *ipv6-address* **link-local**
   - **ipv6 enable**
   - Specifies a global IPv6 address. Specify only the network prefix; the last 64 bits are automatically computed from the switch MAC address.
   - Specifies a link-local address on the interface to be used instead of the automatically configured link-local address when IPv6 is enabled on the interface.
   - Automatically configures an IPv6 link-local address on the interface, and enables the interface for IPv6 processing. The link-local address can only be used to communicate with nodes on the same link.

> **Note**
>
> To remove all manually configured IPv6 addresses from an interface, use the **no ipv6 address** interface configuration command without arguments.

8. Use the **end** command to return to privileged EXEC mode.

```
Device(config)# end
```

9. Use one of these commands to verify the configuration.
   - **show interface** *interface-id*
   - **show ip interface** *interface-id*
   - **show ipv6 interface** *interface-id*

Configure default router preference

Configure default router preference (DRP) for IPv6 router advertisement messages on a switch interface to influence host router selection when multiple routers provide equivalent routing.

Configure default router preference to control which router hosts prefer when multiple routers on a link provide equivalent but not equal-cost routing.

Router advertisement messages are sent with the default router preference (DRP) configured by the **ipv6 nd router-preference** interface configuration command. If no DRP is configured, RAs are sent with a medium preference.

A DRP is useful when two routers on a link might provide equivalent, but not equal-cost routing, and policy might dictate that hosts should prefer one of the routers.

For more information about configuring DRP for IPv6, see the "Implementing IPv6 Addresses and Basic Connectivity" chapter in the *Cisco IOS IPv6 Configuration Library* on Cisco.com.

Use these steps to configure a DRP for a router on an interface:

1. Use the **enable** command to enter privileged EXEC mode.

```
Device> enable
```

2. Use the **configure terminal** command to enter global configuration mode.

```
Device# configure terminal
```

3. Use the **interface** *interface-id* command to enter interface configuration mode, and specify the Layer 3 interface on which to configure DRP.

```
Device(config)# interface gigabitethernet 1/0/1
```

4. Use the **ipv6 nd router-preference** {**high** | **medium** | **low**} command to specify the DRP for the router on the switch interface.

```
Device(config-if)# ipv6 nd router-preference medium
```

5. Use the **end** command to return to privileged EXEC mode.

```
Device(config-if)# end
```

6. Use the **show ipv6 interface** command to verify the configuration.

```
Device# show ipv6 interface
```

7. Use the **copy running-config startup-config** command to save your entries in the configuration file. This step is optional.

```
Device# copy running-config startup-config
```

Configuring IPv6 ICMP rate limiting

Configure the interval and bucket size parameters for IPv6 ICMP error message rate limiting.

This task allows you to customize the IPv6 ICMP rate limiting parameters to control the frequency of ICMP error messages sent by the device.

ICMP rate limiting is enabled by default. The default interval between error messages is 100 milliseconds, and the bucket size (the maximum number of tokens to be stored in a bucket) is 10.

Follow these steps to change the ICMP rate-limiting parameters:

1. Use the **configure terminal** command to enter global configuration mode.

```
Device# configure terminal
```

2. Use the **ipv6 ICMP error-interval** *interval* [*bucketsize*] command to configure the interval and bucket size for IPv6 ICMP error messages.

```
Device(config)# ipv6 icmp error-interval 50 20
```

   - *interval*–The interval (in milliseconds) between tokens being added to the bucket. The range is from 0 to 2147483647 milliseconds.
   - *bucketsize*–(Optional) The maximum number of tokens stored in the bucket. The range is from 1 to 200.

3. Use the **end** command to return to privileged EXEC mode.

```
Device(config)# end
```

4. (Optional) Use the **show ipv6 interface** [*interface-id*] command to verify your entries.

```
Device# show ipv6 interface gigabitethernet0/1
```

5. (Optional) Use the **copy running-config startup-config** command to save your entries in the configuration file.

```
Device# copy running-config startup-config
```

Cisco express forwarding and distributed Cisco express forwarding for IPv6

Describes the Layer 3 IP switching technology for IPv6 that improves network performance through advanced IP lookup and forwarding algorithms.

Cisco Express Forwarding for IPv6 is a Layer 3 IP switching technology that improves network performance through advanced IP lookup and forwarding algorithms. It delivers maximum Layer 3 switching performance with less CPU intensity than fast-switching route-caching. The feature is automatically enabled when IPv6 routing is configured and is disabled when IPv6 routing is unconfigured.

Configuration and operational characteristics

IPv4 Cisco Express Forwarding and distributed Cisco Express Forwarding are enabled by default. IPv6 Cisco Express Forwarding and distributed Cisco Express Forwarding are disabled by default, but automatically enabled when you configure IPv6 routing.

IPv6 Cisco Express Forwarding and distributed Cisco Express Forwarding are automatically disabled when IPv6 routing is unconfigured. Pv6 Cisco Express Forwarding and distributed Cisco Express Forwarding cannot be disabled through configuration. You can verify the IPv6 state by entering the **show ipv6 cef** privileged EXEC command.

To route IPv6 unicast packets, you must first globally configure forwarding of IPv6 unicast packets by using the **ipv6 unicast-routing** global configuration command, and you must configure an IPv6 address and IPv6 processing on an interface by using the **ipv6 address** interface configuration command.

For more information about configuring Cisco Express Forwarding and distributed Cisco Express Forwarding, see *Cisco IOS IPv6 Configuration Library* on Cisco.com.

Configuring static routing for IPv6

Configure static routing for IPv6 to define network paths and enable packet forwarding.

Static IPv6 routing allows you to manually configure routes to specific IPv6 networks. This provides control over packet forwarding paths and enables connectivity to remote networks that are not directly connected.

For more information about configuring static IPv6 routing, see the "Implementing Static Routes for IPv6" chapter in the *Cisco IOS IPv6 Configuration Library* on Cisco.com.

Enable routing by entering the **ip routing** global configuration command, enable the forwarding of IPv6 packets by using the **ipv6 unicast-routing** global configuration command, and enable IPv6 on at least one Layer 3 interface by configuring an IPv6 address on the interface.

Follow these steps to configure static IPv6 routing:

1. Use the **configure terminal** command to enter global configuration mode.

```
Device# configure terminal
```

2. Use the **ipv6 route** *ipv6-prefix/prefix length* {*ipv6-address* | *interface-id* [*ipv6-address*]} [*administrative distance*] command to configure a static IPv6 route.

```
Device(config)# ipv6 route 2001:0DB8::/32 gigabitethernet2/0/1 130
```

- *ipv6-prefix*–The IPv6 network that is the destination of the static route. It can also be a hostname when static host routes are configured.
- */prefix length*–The length of the IPv6 prefix. A decimal value that shows how many of the high-order contiguous bits of the address comprise the prefix (the network portion of the address). A slash mark must precede the decimal value.
- *ipv6-address*–The IPv6 address of the next hop that can be used to reach the specified network. The IPv6 address of the next hop need not be directly connected; recursion is done to find the IPv6 address of the directly connected

next hop. The address must be in the form documented in RFC 2373, specified in hexadecimal using 16-bit values between colons.

- *interface-id*–Specifies direct static routes from point-to-point and broadcast interfaces. With point-to-point interfaces, there is no need to specify the IPv6 address of the next hop. With broadcast interfaces, you should always specify the IPv6 address of the next hop, or ensure that the specified prefix is assigned to the link, specifying a link-local address as the next hop. You can optionally specify the IPv6 address of the next hop to which packets are sent.

> **Note**
>
> You must specify an *interface-id* when using a link-local address as the next hop (the link-local next hop must also be an adjacent router).

- *administrative distance*–(Optional) An administrative distance. The range is 1 to 254; the default value is 1, which gives static routes precedence over any other type of route except connected routes. To configure a floating static route, use an administrative distance greater than that of the dynamic routing protocol.

3. Use the **end** command to return to privileged EXEC mode.

```
Device(config)# end
```

4. (Optional) Use one of these commands to verify your entries by displaying the contents of the IPv6 routing table.
   - **show ipv6 static**[*ipv6-address* | *ipv6-prefix/prefix length*] [**interface***interface-id*] [**detail**]][**recursive**] [**detail**]
   - **show ipv6 route static***updated*

```
Device# show ipv6 static 2001:0DB8::/32 interface gigabitethernet2/0/1
```

or

```
Device# show ipv6 route static
```

   - **interface** *interface-id*–Displays only those static routes with the specified interface as an egress interface.
   - **recursive**–Displays only recursive static routes. The **recursive** keyword is mutually exclusive with the **interface** keyword, but it can be used with or without the IPv6 prefix included in the command syntax.
   - **detail**–Displays this additional information:
     - For valid recursive routes, the output path set, and maximum resolution depth.
     - For invalid routes, the reason why the route is not valid.

5. Use the **copy running-config startup-config** command to save your entries in the configuration file. This step is optional.

```
Device# copy running-config startup-config
```

Enable IPv6 PBR on an interface

Configure Policy-Based Routing (PBR) for IPv6 by creating a route map and associating it with an interface.

Enable Policy-Based Routing (PBR) for IPv6 to allow packet classification and routing based on specific criteria rather than just destination address.

To enable Policy-Based Routing (PBR) for IPv6, you must create a route map that specifies the packet match criteria and desired policy-route action. Next, associate the route map with the required interface. All packets arriving on the specified interface that match the match clauses will be subject to PBR.

In PBR, the **set VRF** command decouples the virtual routing and forwarding (VRF) instance and interface association and allows the selection of a VRF based on access control list (ACL)-based classification using existing PBR or route-map configurations. It provides a single router with multiple routing tables and the ability to select routes based on ACL classification. The router classifies packets based on ACL. It selects a routing table, looks up the destination address, and routes the packet.

1. Use the **configure terminal** command to enter global configuration mode.

```
Device# configure terminal
```

2. Use the **route-map** *map-tag* [**permit** | **deny**] [*sequence-number*] command to define the conditions for redistributing routes from one routing protocol into another, or enable policy routing, and enter route-map configuration mode.

```
Device(config)# route-map rip-to-ospf permit
```

3. Use one of these commands to specify the match criteria.
   - **match length***minimum-length maximum-length*
   - **match ipv6 address** {**prefix-list** *prefix-list-name* | *access-list-name*}

```
Device(config-route-map)# match length 3 200
```

```
Device(config-route-map)# match ipv6 address marketing
```

   - You can specify any or all of these:
     - Matches the Level 3 length of the packet,
     - Matches a specified IPv6 access list, and
     - If you do not specify a **match** command, the route map applies to all packets.

4. Use one of these commands to specify the action or actions to take on the packets that match the criteria.
   - **set ipv6 next-hop***global-ipv6-address* [*global-ipv6-address...*]
   - **set ipv6 default next-hop** *global-ipv6-address* [*global-ipv6-address...*]

```
Device(config-route-map)# set ipv6 next-hop 2001:DB8:2003:1::95
```

```
Device(config-route-map)# set ipv6 default next-hop 2001:DB8:2003:1::95
```

   - You can specify any or all of these:
     - Sets next hop to which to route the packet (the next hop must be adjacent).
     - Sets next hop to which to route the packet, if there is no explicit route for this destination.

5. Use the **exit** command to return to global configuration mode.

```
Device(config-route-map)# exit
```

6. Use the **interface** *type number* command to specify an interface type and number, and enter interface configuration mode.

```
Device(config)# interface fastEthernet 1/0
```

7. Use the **ipv6 policy route-map** *route-map-name* command to identify a route map to use for IPv6 PBR on an interface.

```
Device(config-if)# ipv6 policy route-map interactive
```

8. Use the **end** command to return to privileged EXEC mode.

```
Device(config-if)# end
```

Configure RIP for IPv6

Configure RIP routing for IPv6 to enable IPv6 packet forwarding and routing on Layer 3 interfaces.

RIP for IPv6 enables routing information protocol functionality for IPv6 networks, allowing the exchange of routing information between IPv6-enabled devices and supporting load balancing across multiple equal-cost paths.

For more information about configuring RIP routing for IPv6, see the "Implementing RIP for IPv6" chapter in the *Cisco IOS IPv6 Configuration Library* on Cisco.com.

Before configuring the switch to run IPv6 RIP, you must enable routing by using the **ip routing** global configuration command, enable the forwarding of IPv6 packets by using the **ipv6 unicast-routing** global configuration command, and enable IPv6 on any Layer 3 interfaces on which IPv6 RIP is to be enabled.

Follow these steps to configure RIP routing for IPv6:

1. Use the **configure terminal** command to enter global configuration mode.

```
Device# configure terminal
```

2. Use the **ipv6 router RIP** *name* command to configure an IPv6 RIP routing process, and enter router configuration mode for the process.

```
Device(config)# ipv6 router rip cisco
```

3. Use the **maximum-paths** *number-paths* command to define the maximum number of equal-cost routes that IPv6 RIP can support. This step is optional.

```
Device(config-router)# maximum-paths 6
```

   The range is from 1 to 32, and the default is 16 routes.

4. Use the **exit** command to return to global configuration mode.

```
Device(config-router)# exit
```

5. Use the **interface** *interface-id* command to enter interface configuration mode, and specify the Layer 3 interface to configure.

```
Device(config)# interface gigabitethernet 1/0/1
```

6. Use the **ipv6 RIP** *name* **enable** command to enable the specified IPv6 RIP routing process on the interface.

```
Device(config-if)# ipv6 rip cisco enable
```

7. Use the **ipv6 RIP** *name* **default-information** {**only** | **originate**} command to originate the IPv6 default route (::/0) into the RIP routing process updates sent from the specified interface. This step is optional.

```
Device(config-if)# ipv6 rip cisco default-information only
```

> **Note**
>
> To avoid routing loops after the IPv6 default route (::/0) is originated from any interface, the routing process ignores all default routes received on any interface.

- **only**–Select to originate the default route, but suppress all other routes in the updates sent on this interface.
- **originate**–Select to originate the default route in addition to all other routes in the updates sent on this interface.

8. Use the **end** command to return to privileged EXEC mode.

```
Device(config-if)# end
```

9. Use one of these commands to verify the configuration.
   - **show ipv6 RIP** [*name*] [**interface** *interface-id*] [**database**] [**next-hops**]
   - **show ipv6 RIP**

```
Device# show ipv6 rip cisco interface gigabitethernet 2/0/1
```

or

```
Device# show ipv6 rip
```

- Displays information about current IPv6 RIP processes.
- Displays the current contents of the IPv6 routing table.

Configure OSPF for IPv6

Configure OSPF routing for IPv6 to enable dynamic routing in your network.

You can customize OSPF for IPv6 to fit your network requirements. By default, the settings meet most customer needs.

For more information about configuring OSPF routing for IPv6, see the "Implementing OSPF for IPv6" chapter in the *Cisco IOS IPv6 Configuration Library* on Cisco.com.

ou can customize OSPF for IPv6 for your network. The default settings for OSPF in IPv6 are designed to meet the requirements of most customers and features.

Follow these guidelines:

- Be careful when changing the defaults for IPv6 commands. Changing the defaults might adversely affect OSPF for the IPv6 network.

- Before you enable IPv6 OSPF on an interface, enable routing by using the **ip routing** global configuration command, enable the forwarding of IPv6 packets by using the **ipv6 unicast-routing** global configuration command, and enable IPv6 on Layer 3 interfaces on which you are enabling IPv6 OSPF.

Follow these steps to configure OSPF routing for IPv6:

1. Use the **configure terminal** command to enter global configuration mode.

```
Device# configure terminal
```

2. Use the **ipv6 router OSPF** *process-ID* command to enter OSPF router configuration mode for the process.

```
Device(config)# ipv6 router ospf 21
```

The process ID is a number assigned by an administrator when enabling the OSPF for IPv6 routing process. It is locally assigned and can be any positive integer from 1 to 65,535.

3. Use the **area** *area-ID* **range** {*ipv6-prefix/prefix length*} [**advertise | not-advertise**] [**cost** *cost*] command to consolidate and summarize routes at an area boundary. This step is optional.

```
Device(config-rtr)# area .3 range 2001:0DB8::/32 not-advertise
```

- *area-ID*–Identifier of the area about which routes are to be summarized. It can be specified as either a decimal value or as an IPv6 prefix.
- *ipv6-prefix/prefix length*–This is the destination IPv6 network and a decimal value that specifies how many high-order contiguous bits of the address comprise the prefix (the network portion of the address). The decimal value is preceded by a slash (/).
- **advertise**–(Optional) Sets the address range status to advertise and generate a Type 3 summary link-state advertisement (LSA).
- **not-advertise**–(Optional) Sets the address range status to DoNotAdvertise. The Type 3 summary LSA is suppressed, and component networks remain hidden from other networks.
- **cost** *cost*–(Optional) This option sets the metric or cost for this summary route. The metric is used during OSPF SPF calculations to determine the shortest paths to the destination. The value can range from 0 to 16,777,215.

4. Use the **maximum paths** *number-paths* command to define the maximum number of equal-cost routes to the same destination that IPv6 OSPF should enter in the routing table. This step is optional.

```
Device(config-rtr)# maximum paths 16
```

The range is 1 to 32. The default is 16 paths.

5. Use the **exit** command to return to global configuration mode.

```
Device(config-rtr)# exit
```

6. Use the **interface** *interface-ID* command to enter interface configuration mode, and specify the Layer 3 interface to configure.

```
Device(config)# interface gigabitethernet 1/0/1
```

7. Use the **router** *router-ID* command to enter router configuration mode, and specify the router to configure.

```
Device(config)# router ospfv3 1
```

8. Use the **ipv6 ospf** *process-ID* **area** *area-ID* [**instance** *instance-ID*] command to enable OSPF for IPv6 on the interface.

```
Device(config-if)# ipv6 ospf 21 area .3
```

   - **instance** *instance-ID*–(Optional) Instance identifier.

   Use the **end** command to return to privileged EXEC mode.

9. Use one of these commands to verify the configuration.
   - **show ipv6 ospf** [*process-ID*] [*area-ID*] **interface** [*interface-ID*]

     Displays information about OSPF interfaces.
   - **show ipv6 ospf** [*process-ID*] [*area-ID*]

     Displays general information about OSPF routing processes.

```
Device# show ipv6 ospf 21 interface gigabitethernet2/0/1
```

   or

```
Device# show ipv6 ospf 21
```

Configuring EIGRP for IPv6

Describes the configuration process for enabling IPv6 EIGRP on network switches.

Configuring EIGRP for IPv6 enables IPv6 Enhanced Interior Gateway Routing Protocol functionality on network switches. This is accomplished through specific command sequences and interface configurations.

Configuration requirements

Before configuring the switch to run IPv6 EIGRP, enable routing by entering the **ip routing global configuration** command, enable the forwarding of IPv6 packets by entering the **ipv6 unicast-routing global** configuration command, and enable IPv6 on any Layer 3 interfaces on which you want to enable IPv6 EIGRP.

To set an explicit router ID, use the **show ipv6 EIGRP** command to see the configured router IDs, and then use the **router-ID** command.

As with EIGRP IPv4, you can use EIGRPv6 to specify your EIGRP IPv6 interfaces and to select a subset of those as passive interfaces. Use the **passive-interface** command to make an interface passive, and then use the **no passive-interface** command on selected interfaces to make them active. EIGRP IPv6 does not need to be configured on a passive interface.

For more configuration procedures, see the "Implementing EIGRP for IPv6" chapter in the *Cisco IOS IPv6 Configuration Library* on Cisco.com.

Configuring IPv6 unicast reverse path forwarding

Explains how to configure IPv6 unicast reverse path forwarding to mitigate network security problems.

IPv6 unicast reverse path forwarding is a security feature that

- helps to mitigate problems that are caused by the introduction of malformed or forged (spoofed) IP source addresses into a network by discarding IP packets that lack a verifiable IP source address
- deflects common types of denial-of-service (DoS) attacks, including Smurf and Tribal Flood Network (TFN), by forwarding only packets that have source addresses that are valid and consistent with the IP routing table,
- protects the network of the ISP, its customer, and the rest of the Internet by preventing attackers from using forged or rapidly changing source IP addresses to thwart efforts to locate or filter attacks.

Configuration considerations

Note

- Do not configure Unicast RPF if the switch is in a mixed hardware stack combining more than one switch type.

For detailed IP unicast RPF configuration information, see the *Other Security Features* chapter in the *Cisco IOS Security Configuration Guide, Release 12.4*.

DHCP for IPv6 address assignment

Describes the DHCPv6 address assignment configuration process.

DHCP for IPv6 address assignment is a network configuration method that automatically assigns IPv6 addresses to devices on a network using the DHCPv6 protocol.

Additional information

This section describes only the DHCPv6 address assignment. For more information about configuring the DHCPv6 client, server, or relay agent functions, see the "Implementing DHCP for IPv6" chapter in the *Cisco IOS IPv6 Configuration Library* on Cisco.com.

Default DHCPv6 address assignment configuration

Describes the default DHCPv6 address assignment configuration on Cisco switches, where no DHCPv6 features are enabled or configured by default.

By default, no DHCPv6 features are configured on the switch.

Best practice for DHCPv6 address assignment configuration

Consider these guidelines when configuring DHCPv6 address assignment.

When configuring DHCPv6 address assignment, consider these guidelines

- In the procedures, the specified interface must be one of these Layer 3 interfaces:
  - DHCPv6 IPv6 routing must be enabled on a Layer 3 interface.
  - SVI: a VLAN interface created by using the **interface VLAN** *vlan_id* command.
  - EtherChannel port channel in Layer 3 mode: a port-channel logical interface created by using the **interface port-channel port-channel-number** command.
- The switch can act as a DHCPv6 client, server, or relay agent. The DHCPv6 client, server, and relay function are mutually exclusive on an interface.
- The DHCPv6 client, server, or relay agent runs only on the active switch. If a new switch becomes active after a re-election, it keeps the DHCPv6 configuration, but the local RAM copy of the DHCP server database lease information does not persist.

Enable DHCPv6 server function (CLI)

Configure DHCPv6 server functionality on an interface using CLI commands to enable IPv6 address assignment and pool configuration.

Enable DHCPv6 server functionality on network interfaces to provide IPv6 address assignment. Configure address pools and manage client address allocation in IPv6 networks.

Use the **no** form of the DHCP pool configuration mode commands to change the characteristics of the DHCPv6 pool. Use the **no ipv6 dhcp server** interface configuration command to disable the DHCPv6 server function on an interface.

Follow these steps to enable the DHCPv6 server function on an interface:

1. Use the **configure terminal** command to enter global configuration mode.

```
Device# configure terminal
```

2. Use the **ipv6 dhcp pool** *poolname* command to enter DHCP pool configuration mode, and define the name for the IPv6 DHCP pool.

```
Device(config)# ipv6 dhcp pool 7
```

   The pool name can be a symbolic string, such as Engineering, or an integer, such as 0

3. (Optional) Use the **address prefix** *IPv6-prefix* {**lifetime**} {*t1 t1* | **infinite**} command to specify an address prefix for address assignment.

```
Device(config-dhcpv6)# address prefix 2001:1000::0/64 lifetime 3600
```

   This address must be in hexadecimal, using 16-bit values between colons.

   lifetime *t1 t1*–Specifies a time interval (in seconds) that an IPv6 address prefix remains in the valid state. The range is 5 to 4294967295 seconds. Specify **infinite** for no time interval.

4. (Optional) Use the **link-address** *IPv6-prefix* command to specify a link-address IPv6 prefix.

```
Device(config-dhcpv6)# link-address 2001:1002::0/64
```

   If an address on the incoming interface or a link-address in the packet matches the specified IPv6 prefix, the server uses the corresponding configuration information pool.

   The address must be in hexadecimal and must use 16-bit values separated by colons.

5. Use the **vendor-specific** *vendor-id* command to enter vendor-specific configuration mode and specify a vendor-specific identification number. This step is optional.

```
Device(config-dhcpv6)# vendor-specific 9
```

   This number is the vendor IANA Private Enterprise Number. The range is 1 to 4294967295.

6. (Optional) Use the **suboption** *number* {**address** *IPv6-address* | **ASCII** *ASCII-string* | **hex** *hex-string*} command to enter a vendor-specific suboption number.

```
Device(config-dhcpv6-vs)# suboption 1 address 1000:235D::
```

   The range is 1 to 65535. Enter an IPv6 address, ASCII text, or a hex string as defined by the suboption parameters.

   Use the **exit** command twice to return to global configuration mode.

7. Use the **interface** *interface-id* command to enter interface configuration mode, and specify the interface to configure.

```
Device(config)# interface gigabitethernet 1/0/1
```

8. Use the **ipv6 dhcp server** [*poolname* | **automatic**] [**rapid-commit**] [**preference** *value*] [**allow-hint**] command to enable the DHCPv6 server function on an interface.

```
Device(config-if)# ipv6 dhcp server automatic
```

   - *poolname*–(Optional) User-defined name for the IPv6 DHCP pool. The pool name can be a symbolic string (such as Engineering) or an integer (such as 0).

- **automatic**–(Optional) Enables the system to automatically determine which pool to use when allocating addresses for a client.
- **rapid-commit**–(Optional) Allows two-message exchange method.
- **preference** *value*–(Optional) Configures the preference value carried in the preference option in the advertise message sent by the server. The range is from 0 to 255. The preference value default is 0.
- **allow-hint**–(Optional) Specifies whether the server should consider client suggestions in the SOLICIT message. By default, the server ignores client hints.

Use the **end** command to return to privileged EXEC mode.

9. (Optional) Use one of these commands to verify the configuration.
   - **show ipv6 dhcp pool**
   - **show ipv6 dhcp interface**

```
Device# show ipv6 dhcp pool
```

or

```
Device# show ipv6 dhcp interface
```

- Verifies DHCPv6 pool configuration.
- Verifies that the DHCPv6 server function is enabled on an interface.

Enable DHCPv6 client function

Configure an interface to acquire an IPv6 address from a DHCPv6 server and enable client functionality.

Enable the DHCPv6 client function on an interface to allow automatic IPv6 address assignment from a DHCPv6 server.

Use this procedure when you need to configure an interface to automatically obtain IPv6 addressing information from a DHCPv6 server instead of using static configuration.

Follow these steps to enable the DHCPv6 client function on an interface:

1. Use the **configure terminal** command to enter global configuration mode.

```
Device# configure terminal
```

2. Use the **interface** *interface-id* command to enter interface configuration mode and specify the interface.

```
Device(config)# interface gigabitethernet 1/0/1
```

3. Use the **ipv6 address dhcp** [**rapid-commit**] command to configure the interface for IPv6 address acquisition from the DHCPv6 server.

```
Device(config-if)# ipv6 address dhcp rapid-commit
```

rapid-commit–(Optional) Allow two-message exchange method for address assignment.

4. Use the **ipv6 dhcp client request** [**vendor-specific**] command to request the vendor-specific option. This step is optional.

```
Device(config-if)# ipv6 dhcp client request vendor-specific
```

5. Use the **end** command to return to privileged EXEC mode.

```
Device(config-if)# end
```

6. (Optional) Use the **show ipv6 dhcp interface** command to verify that the DHCPv6 client is enabled on an interface.

```
Device# show ipv6 dhcp interface
```

Display IPv6

Lists commands for monitoring IPv6 and displaying EIGRP IPv6 information.

For complete syntax and usage information on these commands, see the Cisco IOS command reference publications.

Table 4: Command for monitoring IPv6

Command	Purpose
show ipv6 access-list	Displays a summary of access lists.
show ipv6 cef	Displays Cisco Express Forwarding for IPv6.
show ipv6 interface *interface-id*	Displays IPv6 interface status and configuration.
show ipv6 MTU	Displays IPv6 MTU per destination cache.
show ipv6 neighbors	Displays IPv6 neighbor cache entries.
show ipv6 prefix-list	Displays a list of IPv6 prefix lists.
show ipv6 protocols	Displays a list of IPv6 routing protocols on the switch.
show ipv6 RIP	Displays IPv6 RIP routing protocol status.
show ipv6 route	Displays IPv6 route table entries.
show ipv6 static	Displays IPv6 static routes.
show ipv6 traffic	Displays IPv6 traffic statistics.

Configuration examples for IPv6 unicast routing

Describes configuration examples for implementing IPv6 unicast routing on Cisco devices. Explains step-by-step procedures for enabling IPv6 routing protocols and configuring unicast routing scenarios.

Configure IPv6 addressing and enable IPv6 routing

Describes the configuration of IPv6 addressing using a link-local address and a global address based on an IPv6 prefix with the EUI-64 interface ID. This process enables IPv6 unicast routing and verifies the interface configuration.

This configuration enables IPv6 with both a link-local address and a global address based on the IPv6 prefix 2001:0DB8:c18:1::/64, utilizing the EUI-64 interface ID in the low-order 64 bits of both addresses.

The example demonstrates how the interface ID (20B:46FF:FE2F:D940) is appended to the link-local prefix FE80::/64 of the interface. The following commands are used to enable IPv6 unicast routing and configure the interface:

```
Device(config)# ipv6 unicast-routing
Device(config)# interface gigabitethernet0/11
Device(config-if)# ipv6 address 2001:0DB8:c18:1::/64 eui-64
Device(config-if)# end
```

Verification

The interface is configured with IPv6 enabled, displaying both the link-local address (FE80::20B:46FF:FE2F:D940) and the global unicast address (2001:0DB8:c18:1:20B:46FF:FE2F:D940) with the EUI-64 interface ID appended to the specified prefix. Use the show ipv6 interface command to verify:

```
Device# show ipv6 interface gigabitethernet0/11

GigabitEthernet0/11 is up, line protocol is up
IPv6 is enabled, link-local address is FE80::20B:46FF:FE2F:D940
Global unicast address(es):
2001:0DB8:c18:1:20B:46FF:FE2F:D940, subnet is 2001:0DB8:c18:1::/64 [EUI]
Joined group address(es):
FF02::1
FF02::2
FF02::1:FF2F:D940
MTU is 1500 bytes
ICMP error messages limited to one every 100 milliseconds
ICMP redirects are enabled
ND DAD is enabled, number of DAD attempts: 1
ND reachable time is 30000 milliseconds
ND advertised reachable time is 0 milliseconds
ND advertised retransmit interval is 0 milliseconds
ND router advertisements are sent every 200 seconds
ND router advertisements live for 1800 seconds
Hosts use stateless autoconfig for addresses.
```

Configure default router preference: example

Provides an example of configuring a default router preference (DRP) of high for the router on an interface.

This example shows how to configure a DRP of *high* for the router on an interface.

```
Device# configure terminal
Device(config)# interface gigabitethernet1/0/1
Device(config-if)# ipv6 nd router-preference high
Device(config-if)# end
```

Configure IPv4 and IPv6 protocol stacks: example

Provides an example of how to enable IPv4 and IPv6 routing on an interface.

This example shows how to enable IPv4 and IPv6 routing on an interface.

```
Device(config)# ip routing
Device(config)# ipv6 unicast-routing
Device(config)# interface fastethernet1/0/11
Device(config-if)# no switchport
Device(config-if)# ip address 192.168.99.1 255.255.255.0
Device(config-if)# ipv6 address 2001:0DB8:c18:1::/64 eui 64
Device(config-if)#   end
```

Enable DHCPv6 server function: example

Provides configuration examples for enabling DHCPv6 server functionality with different pool configurations and options.

This reference provides configuration examples demonstrating how to enable DHCPv6 server functionality with different pool configurations, including IPv6 address prefixes, link addresses, and vendor-specific options.

This example shows how to configure a pool called *engineering* with an IPv6 address prefix:

```
Device# configure terminal
Device(config)# ipv6 dhcp pool engineering
Device(config-dhcpv6)# address prefix 2001:1000::0/64
Device(config-dhcpv6)# end
```

This example shows how to configure a pool called *testgroup* with three link-addresses and an IPv6 address prefix:

```
Device# configure terminal
Device(config)# ipv6 dhcp pool testgroup
Device(config-dhcpv6)# link-address 2001:1001::0/64
Device(config-dhcpv6)# link-address 2001:1002::0/64
Device(config-dhcpv6)# link-address 2001:2000::0/48
Device(config-dhcpv6)# address prefix 2001:1003::0/64
Device(config-dhcpv6)# end
```

This example shows how to configure a pool called *350* with vendor-specific options:

```
Device# configure terminal
Device(config)# ipv6 dhcp pool 350
Device(config-dhcpv6)# address prefix 2001:1005::0/48
Device(config-dhcpv6)# vendor-specific 9
Device(config-dhcpv6-vs)# suboption 1 address 1000:235D::1
Device(config-dhcpv6-vs)# suboption 2 ascii "IP-Phone"
Device(config-dhcpv6-vs)# end
```

Enable DHCPv6 client function: example

Provides an example of how to acquire an IPv6 address and enable the rapid-commit option.

This example demonstrates how to acquire an IPv6 address and enable the rapid-commit option using the DHCPv6 client function.

This example shows how to acquire an IPv6 address and to enable the rapid-commit option:

```
Device(config)# interface gigabitethernet2/0/1
Device(config-if)# ipv6 address dhcp rapid-commit
```

Configure IPv6 ICMP rate limiting: example

Provides an example of configuring IPv6 ICMP error message rate limiting with specific interval and bucket size parameters.

This example shows how to configure an IPv6 ICMP error message interval of 50 milliseconds and a bucket size of 20 tokens.

```
Device(config)# ipv6 icmp error-interval 50 20
```

Configure static routing for IPv6: example

Provides an example of configuring a floating static route to an interface with an administrative distance.

This example demonstrates how to configure a floating static route to an interface with an administrative distance of 130.

```
Device(config)# ipv6 route 2001:0DB8::/32 gigabitethernet 1/0/1 130
```

Enable PBR on an interface: example

Provides an example configuration that enables policy-based routing (PBR) on a GigabitEthernet interface.

This example demonstrates how to create and configure a route map named PBR-dest-1, specify packet match criteria and desired policy-route action, and enable PBR on GigabitEthernet interface 0/0/1.

```
ipv6 access-list match-dest-1
  permit ipv6 any 2001:DB8:2001:1760::/32
route-map pbr-dest-1 permit 10
  match ipv6 address match-dest-1
  set interface GigabitEthernet 0/0/0
interface GigabitEthernet0/0/1
  ipv6 policy-route-map interactive
```

Enable local PBR for IPv6: example

Provides an example of enabling local policy-based routing for IPv6 traffic using access lists and route maps.

This example demonstrates how to configure local policy-based routing for IPv6 by creating access lists and route maps to control packet forwarding based on destination addresses.

In this example, packets with a destination IPv6 address that match the IPv6 address range allowed by access list PBR-src-90 are sent to the device at IPv6 address 2001:DB8:2003:1::95:

```
ipv6 access-list src-90
  permit ipv6 host 2001:DB8:2003::90 2001:DB8:2001:1000::/64
route-map pbr-src-90 permit 10
  match ipv6 address src-90
  set ipv6 next-hop 2001:DB8:2003:1::95
ipv6 local policy route-map pbr-src-90
```

Configure RIP for IPv6: Example

Provides an example of how to enable the RIP routing process with multiple equal-cost routes and configure it on an interface.

This example shows how to enable the RIP routing process *cisco* with a maximum of eight equal-cost routes and to enable it on an interface.

```
Device(config)# ipv6 router rip cisco
Device(config-router)# maximum-paths 8
Device(config)# exit
Device(config)# interface gigabitethernet2/0/11
Device(config-if)# ipv6 rip cisco enable
```

Display IPv6: example

Provides an example of the output from the show ipv6 interface privileged EXEC command.

This example demonstrates the output format and information displayed when using the **show ipv6 interface** privileged EXEC command to view IPv6 interface configuration and status details.

```
Device# show ipv6 interface
Vlan1 is up, line protocol is up
  IPv6 is enabled, link-local address is FE80::20B:46FF:FE2F:D940
  Global unicast address(es):
    3FFE:C000:0:1:20B:46FF:FE2F:D940, subnet is 3FFE:C000:0:1::/64 [EUI]
  Joined group address(es):
    FF02::1
    FF02::2
```

```
    FF02::1:FF2F:D940
  MTU is 1500 bytes
  ICMP error messages limited to one every 100 milliseconds
  ICMP redirects are enabled
  ND DAD is enabled, number of DAD attempts: 1
  ND reachable time is 30000 milliseconds
  ND advertised reachable time is 0 milliseconds
  ND advertised retransmit interval is 0 milliseconds
  ND router advertisements are sent every 200 seconds
  ND router advertisements live for 1800 seconds
<output truncated>
```

Additional references

Provides supplementary documentation, standards, RFCs, and MIB resources for related topics and implementations.

This reference provides links to supplementary documentation, standards, RFCs, and MIB resources that support the implementation and understanding of related networking topics and Cisco IOS functionality.

Related documents

Related Topic	Document Title
Cisco IOS commands	Cisco IOS Master Commands List, All Releases

Standards and RFCs

Standard/RFC	Title
RFC 5453	*Reserved IPv6 Interface Identifiers*

MIBs

MIB	MIBs Link
All the supported MIBs for this release.	To locate and download MIBs for selected platforms, Cisco IOS releases, and feature sets, use Cisco MIB Locator found at this URL: http://www.cisco.com/go/mibs

4 Configure RIP

Topics:

Information about Routing Information Protocol

Describes the Routing Information Protocol (RIP), an interior gateway protocol designed for small networks.

The Routing Information Protocol (RIP) is an interior gateway protocol (IGP) designed for use in small, homogeneous networks. RIP is a distance-vector routing protocol that uses broadcast User Datagram Protocol (UDP) data packets to exchange routing information. It is documented in RFC 1058.

RIP operation and characteristics

You can find detailed information about RIP in *IP Routing Fundamentals,* published by Cisco Press.

Using RIP, the switch sends routing information updates (advertisements) every 30 seconds. If a router does not receive an update from another router for 180 seconds or more, it marks the routes served by that router as unusable. If there is still no update after 240 seconds, the router removes all routing table entries for the non-updating router.

RIP uses hop counts to rate the value of different routes. The hop count is the number of routers that can be traversed in a route. A directly connected network has a hop count of zero; a network with a hop count of 16 is unreachable. This small range (0 to 15) makes RIP unsuitable for large networks.

If the router has a default network path, RIP advertises a route that links the router to the pseudonetwork 0.0.0.0. The 0.0.0.0 network does not exist; it is treated by RIP as a network to implement the default routing feature. The switch advertises the default network if a default was learned by RIP or if the router has a gateway of last resort and RIP is configured with a default metric. RIP sends updates to the interfaces in specified networks. If an interface's network is not specified, it is not advertised in any RIP update.

RIP for IPv6

Describes a distance-vector routing protocol that provides IPv6 address support and uses hop count as a routing metric.

RIP for IPv6 is a distance-vector protocol that uses hop count as a routing metric. It includes support for IPv6 addresses and prefixes. The all-RIP-routers multicast group address FF02::9 is the destination address for RIP update messages.

Additional information

For configuring RIP for IPv6, see the *Configuring RIP for IPv6* section.

For more information about RIP for IPv6, see the "Implementing RIP for IPv6" chapter in the *Cisco IOS IPv6 Configuration Library* on Cisco.com.

Summary addresses and split horizon

Describes the split-horizon mechanism used by routers on broadcast-type IP networks with distance-vector routing protocols.

Split horizon is a routing mechanism that blocks information about routes from being advertised by a router on any interface from which that information originated. It also optimizes communication among multiple routers, especially when links are broken.

Split horizon functionality

Routers connected to broadcast-type IP networks and using distance-vector routing protocols normally use the split-horizon mechanism to reduce the possibility of routing loops.

How to Configure RIP

Default RIP configuration

Lists the default configuration settings for RIP features and parameters.

This topic provides the default configuration settings for Routing Information Protocol (RIP) features and parameters on the device.

Table 5: Default RIP configuration

Feature	Default Setting
Auto summary	Enabled.
Default-information originate	Disabled.
Default metric	Built-in; automatic metric translations.
IP RIP authentication key-chain	No authentication. Authentication mode: clear text.
IP RIP triggered	Disabled
IP split horizon	Varies with media.
Neighbor	None defined.
Network	None specified.
Offset list	Disabled.
Output delay	0 milliseconds.
Timers basic	• Update: 30 seconds. • Invalid: 180 seconds. • Hold-down: 180 seconds. • Flush: 240 seconds.
Validate-update-source	Enabled.
Version	Receives RIP Version 1 and 2 packets; sends Version 1 packets.

Configure basic RIP parameters

Configure RIP routing by enabling RIP for a network and configuring optional parameters.

Configure RIP routing to enable network routing capabilities and establish routing protocols on your device.

To configure RIP, you enable RIP routing for a network and optionally configure other parameters. On the switch, RIP configuration commands are ignored until you configure the network number.

1. Use the **configure terminal** command to enter global configuration mode.

```
Device# configure terminal
```

2. Use the **ip routing** command to enable IP routing.

```
Device(config)# ip routing
```

This step is required only if IP routing is disabled.

3. Use the **router rip** command to enable a RIP routing process and enter router configuration mode.

```
Device(config)# router rip
```

4. Use the **network** *network-number* command to associate a network with the RIP routing process.

```
Device(config-router)# network 12.0.0.0
```

You must configure a network number for the RIP commands to take effect.

5. Use the **neighbor** *ip-address* command to define a neighboring router with which to exchange routing information. This step is optional.

```
Device(config-router)# neighbor 10.2.5.1
```

6. Configure these optional steps, if required.
   a) Use the **offset-list** [*access-list-number* | *name*] {**in** | **out**} *offset* [*type-number*] command to apply an offset list to RIP route metrics.

```
Device(config-router)# offset-list 103 in 10
```

   b) Use the **timers basic** *update invalid holddown flush* command to adjust routing protocol timers.

```
Device(config-router)# timers basic 45 360 400 300
```

   The values define the update, invalid, holddown, and flush timers, respectively.

   c) Use the **version** {**1** | 2} command to configure the switch to receive and send only RIP Version 1 or RIP Version 2 packets. This step is optional.

```
Device(config-router)# version 2
```

   d) Use the **no auto summary** command to disable automatic summarization.

```
Device(config-router)# no auto summary
```

   e) Use the **output-delay** *delay* command to add an interpacket delay for RIP updates.

```
Device(config-router)# output-delay 8
```

7. Use the **end** command to return to privileged EXEC mode.

```
Device(config-router)# end
```

8. Use the **show ip protocols** command to verify the configuration.

```
Device# show ip protocols
```

Configure RIP authentication

Configure RIP authentication on an interface to enable authentication for RIP Version 2 packets using either plain text or MD5 digest authentication.

This task enables RIP authentication on interfaces to secure RIP Version 2 packet exchanges and prevent unauthorized routing updates.

RIP Version 1 does not support authentication. If you are sending and receiving RIP Version 2 packets, you can enable RIP authentication on an interface. The key chain specifies the set of keys that can be used on the interface. If a key chain is not configured, the interface does not perform authentication, including the default option.

The switch supports two modes of authentication on interfaces for which RIP authentication is enabled: plain text and MD5. The default is plain text.

1. Use the **enable** command to enter privileged EXEC mode.

```
Device> enable
```

2. Use the **configure terminal** command to enter global configuration mode.

```
Device# configure terminal
```

3. Use the **interface** *interface-id* command to enter interface configuration mode, and specify the interface to configure.

```
Device(config)# interface gigabitethernet 1/0/1
```

4. Use the **ip rip authentication key-chain** *name-of-chain* command to enable RIP authentication on the interface.

```
Device(config-if)# ip rip authentication key-chain trees
```

5. Use the **ip rip authentication mode {text | md5}** command to configure the interface to use plain text authentication or MD5 digest authentication.

```
Device(config-if)# ip rip authentication mode md5
```

6. Use the **end** command to return to privileged EXEC mode.

```
Device(config-if)# end
```

7. Use the **show running-config** command to verify the configuration.

```
Device# show running-config
```

Configure RIP for IPv6

Configure RIP routing for IPv6 on a Layer 3 switch to enable IPv6 packet forwarding and routing processes.

This task configures IPv6 RIP routing on a switch to enable IPv6 packet forwarding and establish RIP routing processes for IPv6 networks.

For more information about configuring RIP routing for IPv6, see the "Implementing RIP for IPv6" chapter in the *Cisco IOS IPv6 Configuration Library* on Cisco.com.

Before configuring the switch to run IPv6 RIP:

- Enable routing by using the **ip routing** command in global configuration mode.
- Enable the forwarding of IPv6 packets by using the **ipv6 unicast-routing** command in global configuration mode.

- Enable IPv6 on any Layer 3 interface that will use IPv6 RIP.

Follow these steps to configure RIP routing for IPv6:

1. Use the **configure terminal** command to enter global configuration mode.

```
Device# configure terminal
```

2. Use the **ipv6 router rip** *name* command to configure an IPv6 RIP routing process and enter router configuration mode for the process.

```
Device(config)# ipv6 router rip cisco
```

3. (Optional) Use the **maximum-paths** *number-paths* command to define the maximum number of equal-cost routes that IPv6 RIP can support.

```
Device(config-router)# maximum-paths 6
```

4. Use the **exit** command to return to global configuration mode.

```
Device(config-router)# exit
```

5. Use the **interface** *interface-id* command to enter interface configuration mode, and specify the Layer 3 interface to configure.

```
Device(config)# interface gigabitethernet 1/0/1
```

6. Use the **ipv6 rip** *name* **enable** command to enable the specified IPv6 RIP routing process on the interface.

```
Device(config-if)# ipv6 rip cisco enable
```

7. (Optional) Use the **ipv6 rip** *name* **default-information** {**only** | **originate**} command to originate the IPv6 default route (::/0) into RIP updates sent from the specified interface.

```
Device(config-if)# ipv6 rip cisco default-information only
```

8. Use the **end** command to return to privileged EXEC mode.

```
Device(config-if)# end
```

9. Use one of the following commands to verify the IPv6 RIP configuration:
   - **show ipv6 rip** [*name*] [**interface** *interface-id*] [**database**] [**next-hops**]
   - **show ipv6 rip**

```
Device# show ipv6 rip cisco interface gigabitethernet 2/0/1
```

or

```
Device# show ipv6 rip
```

Configure summary addresses and split horizon

Configure an interface running RIP to advertise summarized local IP address pools and manage split horizon settings.

This task configures RIP summary addresses to advertise summarized local IP address pools on network access servers for dial-up clients and manages split horizon settings on interfaces.

If you want to configure an interface running RIP to advertise a summarized local IP address pool on a network access server for dial-up clients, use the **ip summary-address rip** interface configuration command.

> **Note**
>
> In general, disabling split horizon is not recommended unless you are certain that your application requires it to properly advertise routes.

> **Note**
>
> If split horizon is enabled, neither autosummary nor interface IP summary addresses are advertised.

1. Use the **configure terminal** command to enter global configuration mode.

```
Device# configure terminal
```

2. Use the **interface** *interface-id* command to enter interface configuration mode, and specify the Layer 3 interface to configure.

```
Device(config)# interface gigabitethernet 1/0/1
```

3. Use the **ip address** *ip-address subnet-mask* command to configure the IP address and subnet mask for the interface.

```
Device(config-if)# ip address 10.1.1.10 255.255.255.0
```

4. Use the **ip summary-address rip ip address** *ip-network mask* command to configure the IP address to be summarized and the IP network mask.

```
Device(config-if)# ip summary-address rip ip address 10.1.1.30 255.255.255.0
```

5. Use the **no ip split horizon** command to disable split horizon on the interface.

```
Device(config-if)# no ip split horizon
```

6. Use the **end** command to return to privileged EXEC mode.

```
Device(config-if)# end
```

7. Use the **show ip interface** *interface-id* command to verify the configuration.

```
Device# show ip interface gigabitethernet 1/0/1
```

Configure split horizon

Configure split horizon to reduce the possibility of routing loops on routers connected to broadcast-type IP networks using distance-vector routing protocols.

Split horizon blocks information about routes from being advertised by a router on any interface from which that information originated. This feature can optimize communication among multiple routers, especially when links are broken.

Routers connected to broadcast-type IP networks and using distance-vector routing protocols normally use the split-horizon mechanism to reduce the possibility of routing loops.

> **Note**
>
> In general, we do not recommend disabling split horizon unless you are certain that your application requires it to properly advertise routes.
>
> If split horizon is enabled, neither autosummary nor interface IP summary addresses are advertised.

1. Use the **configure terminal** command to enter global configuration mode.

```
Device# configure terminal
```

2. Use the **interface** *interface-id* command to enter interface configuration mode, and specify the interface to configure.

```
Device(config)# interface gigabitethernet 1/0/1
```

3. Use the **ip address** *ip-address subnet-mask* command to configure the IP address and subnet mask for the interface.

```
Device(config-if)# ip address 10.1.1.10 255.255.255.0
```

4. Use the **no ip split-horizon** command to disable split horizon on the interface.

```
Device(config-if)# no ip split-horizon
```

5. Use the **end** command to return to privileged EXEC mode.

```
Device(config-if)# end
```

6. Use the **show ip interface** *interface-id* command to verify the configuration.

```
Device# show ip interface gigabitethernet 1/0/1
```

Example: configure RIP for IPv6

Provides an example of how to enable the RIP routing process with maximum equal-cost routes on an interface for IPv6.

This example shows how to enable the RIP routing process *cisco* with a maximum of eight equal-cost routes and to enable it on an interface.

```
Devce# configure terminal
Device(config)# ipv6 router rip cisco
Device(config-router)# maximum-paths 8
```

```
Device(config)# exit
Device(config)# interface gigabitethernet2/0/11
Device(config-if)# ipv6 rip cisco enable
```

Example: summary addresses and split horizon

Provides a configuration example demonstrating how summary addresses override autosummary addresses and the effect of split horizon on RIP advertisements.

In this example, the major net is 10.0.0.0. The summary address 10.2.0.0 overrides the autosummary address of 10.0.0.0 so that 10.2.0.0 is advertised out interface Gigabit Ethernet port 2, and 10.0.0.0 is not advertised. In the example, if the interface is still in Layer 2 mode (the default), you must enter a **no switchport** interface configuration command before entering the **ip address** interface configuration command.

Note

If split horizon is enabled, neither autosummary nor interface summary addresses (those configured with the **ip summary-address rip** router configuration command) are advertised.

```
Device(config)# router rip
Device(config-router)# interface gigabitethernet1/0/2
Device(config-if)# ip address 10.1.5.1 255.255.255.0
Device(config-if)# ip summary-address rip 10.2.0.0 255.255.0.0
Device(config-if)# no ip split-horizon
Device(config-if)# exit
Device(config)# router rip
Device(config-router)# network 10.0.0.0
Device(config-router)# neighbor 2.2.2.2 peer-group mygroup
Device(config-router)# end
```

5 Configure VRF-lite

Topics:

VRF-lite

Describes a feature that enables service providers to support multiple VPNs with overlapping IP addresses using virtual routing and forwarding tables.

VRF-lite allows service providers to support multiple VPNs with overlapping IP addresses. The system distinguishes routes for each VPN using input interfaces. It creates virtual packet-forwarding tables by associating Layer 3 interfaces with each VRF. Each VRF interface can be physical (for example, an Ethernet port) or logical (for example, a VLAN SVI). A Layer 3 interface cannot be assigned to more than one VRF simultaneously.

VRF-lite devices and network components

Note

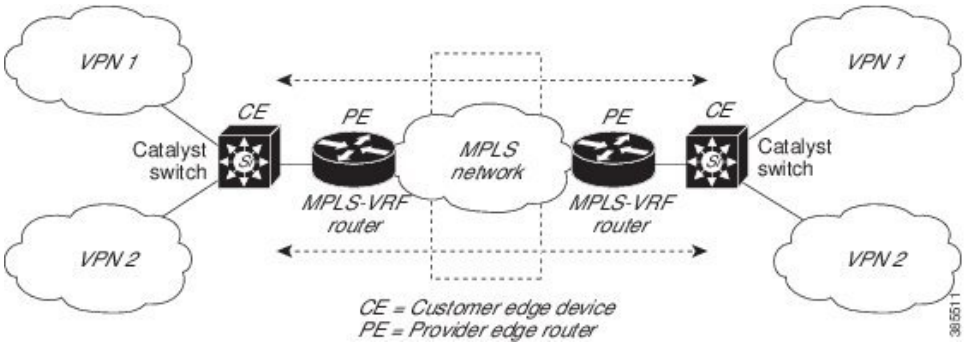
VRF-lite interfaces must be Layer 3 interfaces.

VRF-lite includes these devices:

- Customer edge (CE) devices provide access to the service provider network using a data link to provider edge routers. CE devices advertise local routes to the provider edge router and learn remote VPN routes. Cisco Catalyst switches can function as CE devices.
- Provider routers (also called core routers) operate within the service provider network but do not connect to CE devices.

Each Cisco Catalyst switch can act as multiple virtual CE devices. Since VRF-lite operates at Layer 3, every interface in a VRF must be a Layer 3 interface.

Figure 4: Cisco Catalyst switches acting as multiple virtual CEs



This figure illustrates the packet-forwarding process in a VRF-lite CE-enabled network.

The packet-forwarding process in a VRF-lite CE-enabled network includes these:

1. When the CE receives a packet from a VPN, it looks up the routing table based on the input interface. When a route is found, the CE forwards the packet to the PE.
2. When the ingress PE receives a packet from the CE, it performs a VRF lookup. When a route is found, the router adds a corresponding MPLS label to the packet and sends it to the MPLS network.
3. When an egress PE receives a packet from the network, it strips the label and uses the label to identify the correct VPN routing table. The egress PE then performs the normal route lookup. When a route is found, it forwards the packet to the correct adjacency.
4. When a CE receives a packet from an egress PE, it uses the input interface to look up the correct VPN routing table. If a route is found, the CE forwards the packet within the VPN.

To configure VRF, create a VRF table and specify the Layer 3 interface associated with the VRF. Configure the routing protocols in the VPN and between the CE and the PE. BGP is the recommended routing protocol for distributing VPN routing information across the provider backbone. VRF-lite includes three major components:

- VPN route target communities–Lists all other members of a VPN community. You need to configure VPN route targets for each VPN community member.
- Multiprotocol BGP peering of VPN community PE routers–Propagates VRF reachability information to all members of a VPN community. You need to configure BGP peering in all PE routers within a VPN community.
- VPN forwarding–Transports all traffic between all VPN community members across a VPN service-provider network.

Guidelines for configuring VRF-lite

Consider these guidelines when configuring VRF-lite to ensure proper implementation and avoid common configuration issues.

IPv4 and IPv6 considerations

A switch with VRF-lite is shared by multiple customers, and all customers have their own routing tables.

- Because customers use different VRF tables, you can reuse the same IP addresses. Overlapped IP addresses are allowed in different VPNs.
- VRF-lite lets multiple customers share the same physical link between the PE and the CE. Trunk ports with multiple VLANs separate packets among customers. All customers have their own VLANs.
- For the PE router, there is no difference between using VRF-lite or using multiple CEs. In both cases the multiple virtual Layer 3 interfaces are connected to the VRF-lite device.
- The Cisco Catalyst switch supports configuring VRF by using physical ports, VLAN SVIs, or a combination of both. You can connect SVIs through an access port or a trunk port.
- A customer can use multiple VLANs if they do not overlap with those of other customers. A customer's VLANs are mapped to a specific routing table ID, which is used to identify the appropriate routing tables stored on the switch.
- The Layer 3 TCAM resource is shared between all VRFs.
- A Cisco Catalyst switch using VRF can support one global network and multiple VRFs. The total number of routes supported is limited by the size of the TCAM.
- A single VRF can be configured for both IPv4 and IPv6.
- If an incoming packet's destination address is not found in the VRF table, the packet is dropped. Also, if insufficient TCAM space exists for a VRF route, hardware switching for that VRF is disabled and the corresponding data packets are sent to software for processing.

IPv4 specific considerations

You can use most routing protocols (BGP, OSPF, EIGRP, RIP and static routing) between the CE and the PE. However, we recommend using external BGP (EBGP) for these reasons:

- BGP does not require multiple algorithms to communicate with multiple CEs.
- BGP is designed for passing routing information between systems run by different administrations.
- BGP simplifies passing route attributes to the CE.
- Multicast VRF-lite is not supported.
- The **capability VRF-lite** subcommand under **router OSPF** should be used when configuring OSPF as the routing protocol between the PE and the CE.

IPv6 specific considerations

- VRF-aware OSPFv3, BGPv6, EIGRPv6, and IPv6 static routing are supported.
- VRF-aware IPv6 route applications include ping, telnet, ssh, tftp, ftp, and traceroute. The management interface is not included in this list because it is handled differently, even though both IPv4 and IPv6 VRF can be configured under it.

IPv4 VRF-lite Configuration

VRF-aware services

Describes IP services that can run on multiple routing instances within Virtual Routing and Forwarding (VRF) environments.

VRF-aware services are IP services that can be configured on global interfaces and within the global routing instance. These services are enhanced to run on multiple routing instances. They allow any configured VRF in the system to be specified for the service.

VRF-aware service characteristics

VRF-aware services are implemented in platform-independent modules. VRF provides multiple routing instances in Cisco IOS. Each platform has its own limit on the number of VRFs it supports.

VRF-aware services have these characteristics:

- The user can ping a host in a user-specified VRF.
- ARP entries are learned in separate VRFs. The user can display Address Resolution Protocol (ARP) entries for specific VRFs.

Configure the user interface for ARP

Configure ARP table management and static ARP entries in specified VRF environments.

This task allows you to view ARP table entries and create static ARP mappings within specific VRF instances for network management and troubleshooting purposes.

ARP (Address Resolution Protocol) configuration in VRF environments enables you to manage address resolution within isolated routing domains. This is particularly useful in multi-tenant network environments where traffic isolation is required.

1. Use the **arp vrf** *vrf-name ip-address mac-address ARPA* command to create a static ARP entry in the specified VRF.

```
Switch(config)# arp vrf customer-a 192.0.2.10 0001.0203.0405 ARPA
```

2. Use the **show ip arp vrf** *vrf-name* command to display the ARP table (static and dynamic entries) in the specified VRF.

```
Switch# show ip arp vrf customer-a
```

Configure per-VRF for TACACS+ servers

Configure per-virtual route forwarding (per-VRF) authentication, authorization, and accounting (AAA) on TACACS+ servers.

Configure per-VRF for TACACS+ servers to enable per-virtual route forwarding (per-VRF) authentication, authorization, and accounting (AAA) on TACACS+ servers.

The per-VRF for TACACS+ servers feature enables you to configure per-virtual route forwarding (per-VRF) authentication, authorization, and accounting (AAA).

Create the VRF routing table, configure the interface, then configure per-VRF on the TACACS+ server.

Before configuring per-VRF on a TACACS+ server, you must have configured AAA and a server group.

1. Use the **vrf definition** *vrf-name* command to name the VRF and enter VRF configuration mode.

```
Switch(config)# vrf definition tacacs-vrf
```

2. Use the **rd** *route-distinguisher* command to create routing and forwarding tables for a VRF instance.

```
Switch(config-vrf)# rd 65000:100
```

   Use the **exit** command to exit VRF configuration mode.

3. Use the **interface** *interface-name* command to configure an interface and enter interface configuration mode.

```
Switch(config)# interface Loopback100
```

4. Use the **vrf forwarding** *vrf-name* command to configure a VRF for the interface.

```
Switch(config-if)# vrf forwarding tacacs-vrf
```

5. Use the **ip address** *IP-address mask* [**secondary**] command to set a primary or secondary IP address for an interface.

```
Switch(config-if)# ip address 192.0.2.1 255.255.255.255
```

   Use the **exit** command to exit interface configuration mode.

6. Use the **aaa group server tacacs+** *group-name* command to group different TACACS+ server hosts into distinct lists and distinct methods and enter server-group configuration mode.

```
Switch(config)# aaa group server tacacs+ tacacs-servers
```

7. Use the **server-private** {*IP-address* | *name*} [**nat**] [**single-connection**] [**port** *port-number*] [**timeout** *seconds*] [**key** [**0** | **7**] *string*] command to configure the IP address of the private TACACS+ server for the group server.

```
Switch(config-sg-tacacs+)# server-private 198.51.100.10 port 49 timeout 5 key
 0 tacacs-key
```

8. Use the **vrf forwarding** *vrf-name* command to configure the VRF reference of a AAA TACACS+ server group.

```
Switch(config-sg-tacacs+)# vrf forwarding tacacs-vrf
```

9. Use the **ip tacacs source-interface** *subinterface-name* command to use the IP address of a specified interface for all outgoing TACACS+ packets.

```
Switch(config-sg-tacacs+)# ip tacacs source-interface Loopback100
```

Configure a VPN routing session

Configure routing within the VPN with any supported routing protocol (RIP, OSPF, or BGP) or with static routing. The configuration shown here is for OSPF, but the process is the same for other protocols.

Configure routing within a VPN to enable communication between network devices using OSPF or other supported routing protocols.

Routing within the VPN can be configured with any supported routing protocol (RIP, OSPF, or BGP) or with static routing. The configuration shown here is for OSPF, but the process is the same for other protocols.

1. Use the **configure terminal** command to enter global configuration mode.

```
Switch# configure terminal
```

2. Use the **router OSPF** *process-ID* **vrf vrf-name** command to enable OSPF routing, specify a VPN forwarding table, and enter router configuration mode.

```
Switch(config)# router ospf 10 vrf customer-a
```

3. Use the **capability vrf-lite** command to enable OSPF VRF-lite capability.

```
Switch(config-router)# capability vrf-lite
```

4. Use the **log-adjacency-changes** command to log changes in the adjacency state (the default state). This step is optional.

```
Switch(config-router)# log-adjacency-changes
```

5. Use the **redistribute BGP** *autonomous-system-number subnets* command to set the switch to redistribute information from the BGP network to the OSPF network.

```
Switch(config-router)# redistribute bgp 65000 subnets
```

6. Use the **network** *network-number* **area** *area-ID* command to define a network address and mask on which OSPF runs and the area ID for that network address.

```
Switch(config-router)# network 10.10.10.0 area 0
```

7. Use the **end** command to return to privileged EXEC mode.

```
Switch(config-router)# end
```

8. (Optional) Use the **show ip OSPF** *process-ID* command to verify the configuration of the OSPF network.

```
Switch# show ip ospf 10
```

9. (Optional) Use the **copy running-config startup-config** command to save your entries in the configuration file.

```
Switch# copy running-config startup-config
```

Use the **no router OSPF** *process-ID* **vrf** *vrf-name* global configuration command to disassociate the VPN forwarding table from the OSPF routing process.

Configure BGP PE to CE routing sessions

Configure BGP routing sessions between Provider Edge (PE) and Customer Edge (CE) routers to enable communication between VRF networks.

This task establishes BGP routing sessions between PE and CE routers. These sessions enable routing between VPN sites and facilitate communication across MPLS VPN networks.

Service providers use BGP PE to CE routing sessions in MPLS VPN environments to exchange routing information with customer networks. This configuration enables the PE router to learn customer routes and advertise them to other PE routers in the MPLS backbone.

Follow these steps to configure BGP PE to CE routing sessions:

1. Use the **router bgp** *autonomous-system-number* command to configure the BGP routing process with the AS number passed to other BGP routers and enter router configuration mode.

```
Switch(config)# router bgp 65000
```

   Use the **no router bgp** *autonomous-system-number* global configuration command tto delete the BGP routing process. Use the command with keywords to delete routing characteristics.

2. Use the **network** *network-number* **mask** *network-mask* command to specify a network and mask to announce using BGP.

```
Switch(config-router)# network 10.10.10.0 mask 255.255.255.0
```

3. Use the **redistribute ospf** *process-ID match internal* command to set the switch to redistribute OSPF internal routes.

```
Switch(config-router)# redistribute ospf 10 match internal
```

4. Use the **network** *network-number* **area** *area-ID* command to define a network address and mask on which OSPF runs and the area ID for that network address.

```
Switch(config-router)# network 10.20.20.0 area 0
```

5. Use the **address-family ipv4 vrf** *vrf-name* command to define BGP parameters for PE to CE routing sessions and enter VRF address-family mode.

```
Switch(config-router-af)# address-family ipv4 vrf customer-a
```

6. Use the **neighbor** *address* **remote-AS** *AS-number* command to define a BGP session between PE and CE routers.

```
Switch(config-router-af)# neighbor 192.0.2.2 remote-as 65001
```

7. Use the **neighbor** *address* **activate** command to activate the advertisement of the IPv4 address family.

```
Switch(config-router-af)# neighbor 192.0.2.2 activate
```

8. Use the **end** command to return to privileged EXEC mode.

```
Switch(config-router-af)# end
```

9. Use the **show ip BGP** [**ipv4**] [**neighbors**] command to verify BGP configuration.

```
Switch# show ip bgp ipv4 neighbors
```

Configure IPv4 VRFs

Configure IPv4 Virtual Routing and Forwarding (VRF) instances to create separate routing tables for network traffic isolation.

Configure IPv4 Virtual Routing and Forwarding (VRF) instances to create separate routing tables that enable network traffic isolation and support multiple routing domains on a single device.

VRFs provide network segmentation by creating isolated routing tables. Each VRF maintains its own set of routing information and interfaces, allowing you to implement multiple virtual networks on a single physical infrastructure.

1. Use the **configure terminal** command to enter global configuration mode.

```
Switch# configure terminal
```

2. Use the **ip routing** command to enable IP routing.

```
Switch(config)# ip routing
```

3. Use the **vrf definition** *vrf-name* command to name the VRF and enter VRF configuration mode.

```
Switch(config)# vrf definition customer-a
```

Use the **no vrf definition***vrf-name* global configuration command to delete a VRF and to remove all interfaces from it. Use the **no vrf forwarding** interface configuration command to remove an interface from the VRF.

4. Use the **rd** *route-distinguisher* command to create a VRF table by specifying a route distinguisher.

```
Switch(config-vrf)# rd 65000:10
```

Enter either an Autonomous System number number and an arbitrary number (xxx:y) or an IP address and arbitrary number (A.B.C.D:y).

5. Use the **route-target** {**export** | **import** | **both**} *route-target-ext-community* command to create a list of import, export, or import and export route target communities for the specified VRF.

```
Switch(config-vrf)# route-target both 65000:10
```

Enter either an AS system number and an arbitrary number (xxx:y) or an IP address and an arbitrary number (A.B.C.D:y).

> **Note**
>
> This command is effective only if BGP is running.

6. Use the **import map** *route-map* command to associate a route map with the VRF. This step is optional.

```
Switch(config-vrf)# import map CUSTOMER-A-IN
```

7. Use the **interface** *interface-id* command to enter interface configuration mode and specify the Layer 3 interface to associate with the VRF. The interface can be a routed port or SVI.

```
Switch(config)# interface GigabitEthernet1/0/1
```

8. Use the **vrf forwarding** *VRF-name* command to associate the VRF with the Layer 3 interface.

```
Switch(config-if)# vrf forwarding customer-a
```

Use the **end** command to return to privileged EXEC mode.

9. (Optional) Use the **show IP VRF** [**brief** | **detail** | **interfaces**] [*VRF-name*] command to verify the configuration and display information about the configured VRFs.

```
Switch# show ip vrf detail customer-a
```

IPv6 VRF-lite Configuration

VRF-aware services

Describes IPv6 services that are enhanced to run on multiple routing instances and are VRF-aware.

VRF-aware services are IPv6 services that can be configured on global interfaces and within the global routing instance. These services are enhanced to run on multiple routing instances, and they allow any configured VRF in the system to be specified for the service.

VRF-aware service characteristics

VRF-aware services are implemented in platform-independent modules. VRF provides multiple routing instances in Cisco IOS. Each platform has its own limit on the number of VRFs it supports.

VRF-aware services have these characteristics:

- The user can ping a host in a user-specified VRF.
- Neighbor Discovery entries are learned in separate VRFs. The user can display Neighbor Discovery (ND) entries for specific VRFs.

These services are VRF-aware:

- Ping
- Unicast Reverse Path Forwarding (uRPF)
- Traceroute
- FTP and TFTP
- Telnet and SSH
- NTP

Configure the user interface for PING

Configure a VRF-aware PING to test IPv6 connectivity within a specified virtual routing and forwarding instance.

Configure a VRF-aware PING to test IPv6 connectivity within a specified virtual routing and forwarding instance.

Use this task when you need to verify IPv6 network connectivity within a specific VRF context, which allows for isolated routing tables and separate network domains.

Use the **ping vrf** *vrf-name* **ipv6-host** command to ping an IPv6 host or address in the specified VRF.

```
Switch# ping vrf customer-a 2001:db8:10::20
```

Configure the user interface for uRPF

Configure uRPF on an interface assigned to a VRF where source lookup is performed in the VRF table.

This task enables unicast Reverse Path Forwarding (uRPF) on a network interface to prevent IP address spoofing by verifying that incoming packets arrive on the interface expected based on the routing table.

You can configure uRPF on an interface assigned to a VRF. Source lookup is performed in the VRF table.

1. Use the **configure terminal** command to enter global configuration mode.

```
Switch# configure terminal
```

2. Use the **interface** *interface-id* command to enter interface configuration mode and specify the Layer 3 interface to configure.

```
Switch(config)# interface GigabitEthernet1/0/1
```

3. Use the **no switchport** command to remove the interface from Layer 2 switchport mode if the interface is a physical interface.

```
Switch(config-if)# no switchport
```

4. Use the **vrf forwarding** *vrf-name* command to configure VRF on the interface.

```
Switch(config-if)# vrf forwarding customer-a
```

5. Use the **ipv6 address** *ip-address* **subnet-mask** command to configure the IPv6 address for the interface.

```
Switch(config-if)# ipv6 address 2001:db8:10::1/64
```

6. Use the **ipv6 verify unicast source reachable-via rx allow-default** command to enable uRPF on the interface.

```
Switch(config-if)# ipv6 verify unicast source reachable-via rx allow-default
```

7. Use the **end** command to return to privileged EXEC mode.

```
Switch(config-if)# end
```

Configure the user interface for traceroute

Describes how to configure the user interface for traceroute by specifying a VPN VRF name to locate the destination IPv6 address within that virtual routing and forwarding instance.

Use the **traceroute vrf** *vrf-name ipv6address* command to specify the name of a VPN VRF in which to find the destination address.

```
Switch# traceroute vrf customer-a 2001:db8:10::20
```

Configure the user interface for telnet and SSH

Configure Telnet and SSH connections to IPv6 hosts or addresses in specified VRFs through the user interface.

This task enables you to establish Telnet and SSH connections to IPv6 hosts or addresses within specified Virtual Routing and Forwarding (VRF) instances.

Use this configuration when you need to connect to IPv6 hosts or addresses that are located in specific VRF instances on your network. This is particularly useful in environments where network segmentation requires connections through designated VRF contexts.

1. Use the **telnet** *ipv6-address*|**vrf** *vrf-name* command to connect through Telnet to an IPv6 host or address in the specified VRF.

```
Switch# telnet 2001:db8:10::20 /vrf customer-a
```

2. Use the **ssh** *username* **vrf** *vrf-name* **ipv6-host** command to connect through SSH to an IPv6 host or address in the specified VRF.

```
Switch# ssh admin vrf customer-a 2001:db8:10::20
```

Configure the user interface for NTP
Configure NTP server and peer settings in a specified VRF.

Configure the NTP (Network Time Protocol) server and peer settings within a Virtual Routing and Forwarding (VRF) instance. This ensures accurate time synchronization for the device.

NTP configuration allows the device to synchronize its system clock with external time sources. In VRF environments, you must specify the VRF instance to ensure proper routing of NTP traffic.

1. Use the **configure terminal** command to enter global configuration mode.

```
Device# configure terminal
```

2. Use the **ntp server vrf** *vrf-name ipv6-host* command to configure the NTP server in the specified VRF.

```
Device(config)# ntp server vrf management-vrf 2001:db8:100::10
```

3. Use the **ntp peer vrf** *vrf-name ipv6-host* command to configure the NTP peer in the specified VRF.

```
Device(config)# ntp peer vrf management-vrf 2001:db8:100::11
```

Configure IPv6 VRFs

Configure IPv6 Virtual Routing and Forwarding (VRF) instances. Create separate routing tables and enable multicast routing.

Configure IPv6 VRFs to create isolated routing tables for network segmentation and enable IPv6 multicast routing functionality.

VRFs provide network virtualization by creating separate routing instances on a single physical device. They are essential for implementing IPv6 multicast routing and network segmentation in enterprise environments.

1. Use the **configure terminal** command to enter global configuration mode.

```
Switch# configure terminal
```

2. Use the **vrf definition** *vrf-name* command to name the VRF and enter VRF configuration mode.

```
Switch(config)# vrf definition ipv6-vrf
```

3. (Optional) Use the **rd** *route-distinguisher* command to create a VRF table by specifying a route distinguisher.

```
Switch(config-vrf)# rd 65000:20
```

Enter either an Autonomous System number and an arbitrary number (xxx:y) or an IP address and an arbitrary number (A.B.C.D:y).

4. (Optional) Use the **address-family** *ipv4* | *ipv6* command to specify the address family to configure.

```
Switch(config-vrf)# address-family ipv6
```

IPv4 is configured by default. This command is required for IPv6.

5. Use the **route-target** {**export** | **import** | **both**} *route-target-ext-community* command to create a list of import, export, or import and export route target communities for the specified VRF.

```
Switch(config-vrf-af)# route-target both 65000:20
```

Enter either an AS system number and an arbitrary number (xxx:y) or an IP address and an arbitrary number (A.B.C.D:y).

> **Note**
>
> This command is effective only if BGP is running.

6. Use the **exit** command to exit VRF address-family configuration mode and return to VRF configuration mode.

```
Switch(config-vrf-af)# exit
```

7. Use the **vrf definition** *vrf-name* command to enter VRF configuration mode.

```
Switch(config)# vrf definition ipv6-vrf
```

8. Use the **ipv6 multicast multitopology** command to enable multicast specific RPF topology.

```
Switch(config-vrf)# ipv6 multicast multitopology
```

9. Use the **address-family ipv6 multicast** command to enter multicast IPv6 address-family configuration mode.

```
Switch(config-vrf)# address-family ipv6 multicast
```

Associate interfaces to the defined VRFs

Configure interface association with VRFs to enable VRF forwarding on Layer 3 interfaces.

This task associates Layer 3 interfaces with Virtual Routing and Forwarding (VRF) instances to enable VRF forwarding functionality.

After defining VRFs, associate interfaces with them. This enables traffic forwarding within the VRF domain. The interface can be a routed port or SVI.

1. Use the **interface** *interface-id* command to enter interface configuration mode and specify the Layer 3 interface to be associated with the VRF.

```
Switch(config)# interface GigabitEthernet1/0/1
```

   The interface can be either a routed port or an SVI.

2. Use the **no switchport** command to remove the interface from Layer 2 switchport mode if the interface is a physical interface.

```
Switch(config-if)# no switchport
```

3. Use the **vrf forwarding** *vrf-name* command to associate the VRF with the Layer 3 interface.

```
Switch(config-if)# vrf forwarding customer-a
```

4. Use the **ipv6 enable** command to enable IPv6 on the interface.

```
Switch(config-if)# ipv6 enable
```

5. Use the **ipv6 address** *ip-address subnet-mask* command to configure the IPv6 address for the interface.

```
Switch(config-if)# ipv6 address 2001:db8:10::1/64
```

6. (Optional) Use the **show ipv6 VRF** [**brief | detail | interfaces**] [*VRF-name*] command to verify the configuration and display information about the configured VRFs.

```
Switch# show ipv6 vrf interfaces customer-a
```

IPv6 Routing Processes

Configure VRF static routes

Configure IPv6 static routes specific to VRF to enable routing within virtual routing and forwarding instances.

This task allows you to create static routes for a Virtual Routing and Forwarding (VRF) instance. This action enables isolated routing tables for different network segments.

Use VRF static routes to maintain separate routing tables for different customers, services, or network segments. This configuration is essential for environments that require network isolation while sharing the same physical infrastructure.

1. Use the **configure terminal** command to enter global configuration mode.

```
Switch# configure terminal
```

2. Use the **ipv6 route** [**VRF** *VRF-name*] *ipv6-prefix*/*prefix-length* {*ipv6-address* | **interface-type** *interface-number* [*ipv6-address*]} command to configure static routes specific to a VRF.

```
Switch(config)# ipv6 route vrf customer-a 2001:db8:200::/64
TenGigabitEthernet32 2001:db8:10::2
```

Configure OSPFv3 router process

Configure OSPFv3 router process to enable routing for IPv6 networks.

Configure the OSPFv3 router process to enable IPv6 routing. Then, establish OSPFv3 areas with the required redistribution settings.

OSPFv3 is the IPv6 version of the OSPF routing protocol that provides dynamic routing capabilities for IPv6 networks. This configuration establishes the basic router process and area settings.

1. Use the **configure terminal** command to enter global configuration mode.

```
Switch# configure terminal
```

2. Use the **router ospfv3** *process-ID* command to enable OSPFv3 router configuration mode for the IPv6 address family.

```
Switch(config)# router ospfv3 10
```

3. Use the **area** *area-ID* [**default-cot** | **nssa** | **stub**] command to configure the OSPFv3 area.

```
Switch(config-router)# area 0 stub
```

4. Use the **router-id** *router-ID* command to set a fixed router ID.

```
Switch(config-router)# router-id 192.0.2.10
```

5. Use the **address-family ipv6 unicast vrf** *vrf-name* command o enter IPv6 address family configuration mode for OSPFv3 in the specified VRF.

```
Switch(config-router)# address-family ipv6 unicast vrf customer-a
```

6. Use the **redistribute source-protocol** [*process-ID*] **options** command to redistribute IPv6 routes from one routing domain into another routing domain.

```
Switch(config-router)# redistribute connected metric 10
```

7. Use the **end** command to return to privileged EXEC mode.

```
Switch(config-router)# end
```

Enable OSPFv3 on an interface

Configure OSPFv3 on a specific interface to enable IPv6 routing protocol functionality.

Enable OSPFv3 on an interface to provide IPv6 routing protocol functionality and allow the interface to participate in OSPFv3 routing operations.

OSPFv3 is the IPv6 version of the OSPF routing protocol. You configure OSPFv3 on interfaces to enable IPv6 routing within your network topology.

1. Use the **configure terminal** command to enter global configuration mode.

```
Switch# configure terminal
```

2. Use the **interface** *type-number* command to specify an interface type and number, and place the switch in interface configuration mode.

```
Switch(config)# interface GigabitEthernet1/0/1
```

3. Use the **ospfv3** *process-id* **area** *area-ID* **ipv6** [**instance** *instance-id*] command to enable OSPFv3 on an interface with IPv6 AF.

```
Switch(config-if)# ospfv3 10 area 0 ipv6 instance 1
```

4. Use the **end** command to return to privileged EXEC mode.

```
Switch(config-if)# end
```

Configure EIGRPv6 routing process

Configure an EIGRPv6 routing process with VRF-Lite support and topology settings.

Configure an EIGRP routing process for IPv6 with VRF-Lite support to enable routing in virtual routing and forwarding instances.

Use this procedure to set up an EIGRPv6 routing process with virtual routing and forwarding (VRF) instances, and to configure topology settings.

1. Use the **configure terminal** command to enter global configuration mode.

```
Switch# configure terminal
```

2. Use the **router eigrp** *virtual-instance-name* command to configure the EIGRP routing process and enter router configuration mode.

```
Switch(config)# router eigrp CUSTOMER-EIGRP
```

3. Use the **address-family ipv6 vrf** *vrf-name* **autonomous-system** *autonomous-system-number* command to enable EIGRP IPv6 VRF-Lite and enter address family configuration mode.

```
Switch(config-router)# address-family ipv6 vrf customer-a autonomous-system
100
```

4. Use the **topology** {**base** | **topology-name tid number**} command to configure an EIGRP process to route IP traffic under the specified topology instance. This command also enters address family topology configuration mode.

```
Switch(config-router-af)# topology base
```

5. Use the **exit-af-topology** command to exit address family topology configuration mode.

```
Switch(config-router-af-topology)# exit-af-topology
```

6. Use the **eigrp router-id** *IP-address* command to enable the use of a fixed router-id.

```
Switch(config-router)# eigrp router-id 192.0.2.10
```

Configure EBGPv6 routing process

Configure an External Border Gateway Protocol version 6 (EBGPv6) routing process to enable IPv6 routing between autonomous systems.

Configure an EBGPv6 routing process to establish IPv6 connectivity and routing between different autonomous systems using Border Gateway Protocol.

EBGPv6 allows routers in different autonomous systems to exchange IPv6 routing information. This configuration enables the switch to participate in inter-domain IPv6 routing.

Follow these steps to configure an EBGPv6 routing process:

1. Use the **configure terminal** command to enter global configuration mode.

```
Switch# configure terminal
```

2. Use the **router bgp** *as-number* command to enter router configuration mode for the specified routing process.

```
Switch(config)# router bgp 65000
```

3. Use the **neighbor** *peer-group-name peer-group* command to create a multiprotocol BGP peer group.

```
Switch(config-router)# neighbor ce-ipv6 peer-group
```

4. Use the **neighbor** {*ip-address* | *ipv6-address*[*%*] | *peer-group-name*} **remote-as** *autonomous-system-number* [**alternate-as** *autonomous-system-number* …] command to add the IPv6 address of the neighbor in the specified autonomous system to the IPv6 multiprotocol BGP neighbor table of the local router.

```
Switch(config-router)# neighbor 2001:db8:1::2 remote-as 65001
```

5. Use the **address-family ipv6** [**vrf** *vrf-name*] [**unicast** | **multicast** | **vpnv6**] command to specify the IPv6 address family and enter address family configuration mode.

```
Switch(config-router)# address-family ipv6 vrf customer-a unicast
```

   - The unicast keyword specifies the IPv6 unicast address family. By default, the switch is placed in configuration mode for the IPv6 unicast address family if the unicast keyword is not specified with the address-family ipv6 command.
   - The multicast keyword specifies IPv6 multicast address prefixes.

6. Use the **neighbor** *ipv6-address* **peer-group** *peer-group-name* command to assign the IPv6 address of a BGP neighbor to a peer group.

```
Switch(config-router-af)# neighbor 2001:db8:1::2 peer-group ce-ipv6
```

7. Use the **neighbor** {*ip-address* | *peer-group-name* | *ipv6-address*[*%*]} **route-map** *map-name* {**in** | **out**} command to apply a route map to incoming or outgoing routes.

```
Switch(config-router-af)# neighbor 2001:db8:1::2 route-map IPV6-IN in
```

   Changes to the route map do not affect existing peers until the peering is reset or a soft reset is performed. Entering the clear BGP ipv6 command with the soft and in keywords performs a soft reset.

VPN Co-existence

VPN co-existence between IPv4 and IPv6

Describes how IPv4 and IPv6 VPN configurations can coexist on the same interface with backward compatibility between CLI versions.

VPN co-existence between IPv4 and IPv6 is a networking capability with several benefits. It enables backward compatibility between older IPv4 CLI and newer IPv6 CLI configurations. It also allows configurations to contain both IPv4 and IPv6 CLI on the same system. Lastly, it permits an interface to have an IP address defined within a VRF alongside an IPv6 address defined in the global routing table.

Configuration behavior

Mixed VRF assignment configuration

The IPv4 CLI allows the same interface to have an IP address defined within a VRF and an IPv6 address defined in the global routing table.

This configuration demonstrates how IPv4 and IPv6 addresses can be assigned to different VRFs on the same interface:

```
vrf definition red
 rd 100:1
 address family ipv6
 route-target both 200:1
 exit-address-family
!
ip vrf blue
 rd 200:1
 route-target both 200:1
!
interface Ethernet0/0
 vrf forwarding red
```

```
 ip address 50.1.1.2 255.255.255.0
 ipv6 address 4000::72B/64
!
interface Ethernet0/1
 vrf forwarding blue
 ip address 60.1.1.2 255.255.255.0
 ipv6 address 5000::72B/64
```

In this example, all addresses (v4 and v6) defined for Ethernet0/0 refer to VRF red whereas for Ethernet0/1, the IP address refers to VRF blue but the ipv6 address refers to the global IPv6 routing table.

Displaying VRF-lite Status

Display IPv4 VRF-lite status

Lists commands used to display information about VRF-lite configuration and status.

To display information about VRF-lite configuration and status, perform one of these tasks:

Command	Purpose
`Switch# show ip protocols vrf vrf-name`	Displays routing protocol information associated with a VRF.
`Switch# show ip route vrf vrf-name [connected] [protocol [as-number]] [list] [mobile] [odr] [profile] [static] [summary][supernets-only]`	Displays IP routing table information associated with a VRF.
`Switch# show ip vrf [brief | detail | interfaces] [vrf-name]`	Displays information about the defined VRF instances.
`Switch# bidir vrf instance-name a.b.c.d | active | bidriectional| count | interface | proxy | pruned | sparse | ssm | static | summary`	Displays information about the defined VRF instances.

This example shows how to display multicast route table information within a VRF instance:

```
Switch# show ip mroute 226.0.0.2
IP Multicast Routing Table
Flags: S - Sparse, B - Bidir Group, s - SSM Group, C - Connected,
       L - Local, P - Pruned, R - RP-bit set, F - Register flag,
       T - SPT-bit set, J - Join SPT, M - MSDP created entry, E - Extranet,
       X - Proxy Join Timer Running, A - Candidate for MSDP Advertisement,
       U - URD, I - Received Source Specific Host Report,
       Z - Multicast Tunnel, z - MDT-data group sender,
       Y - Joined MDT-data group, y - Sending to MDT-data group,
       G - Received BGP C-Mroute, g - Sent BGP C-Mroute,
       N - Received BGP Shared-Tree Prune, n - BGP C-Mroute suppressed,
       Q - Received BGP S-A Route, q - Sent BGP S-A Route,
       V - RD & Vector, v - Vector, p - PIM Joins on route,
       x - VxLAN group, c - PFP-SA cache created entry
Outgoing interface flags: H - Hardware switched, A - Assert winner, p - PIM
Join
```

```
Timers: Uptime/Expires
Interface state: Interface, Next-Hop or VCD, State/Mode

(*, 226.0.0.2), 00:01:17/stopped, RP 1.11.1.1, flags: SJCF
  Incoming interface: Null, RPF nbr 0.0.0.0
  Outgoing interface list:
    Vlan100, Forward/Sparse, 00:01:17/00:02:36

(5.0.0.11, 226.0.0.2), 00:01:17/00:01:42, flags: FT
  Incoming interface: Vlan5, RPF nbr 0.0.0.0
  Outgoing interface list:
    Vlan100, Forward/Sparse, 00:01:17/00:02:36
```

Display IPv6 VRF-lite status

Lists commands to display information about VRF-lite configuration and status.

To display information about VRF-lite configuration and status, perform one of these tasks:

Command	Purpose
Switch# **show ipv6 mroute vrf** *instance-name* [*X:X:X:X::X/<0-128>*] [**bgp** | **connected** | **eigrp** | **interface** | **isis** | **local** | **nd** | **nsf** | **ospf** | **repair** | **rip** | **shortcut** | **static** | **summary** | **tag** | **updated** | **watch**]	Displays routing protocol information associated with a VRF.
Switch# **show ipv6 mfib vrf** *instance-name a.b.c.d* [**active** | **all** | **count** | **linkscope**| **route** | **summary** | **update-sets** | **verbose**]	Displays information about the defined VRF instances.

This example shows how to display multicast route table information within a VRF instance:

```
show ipv6 mroute vrf vrf1 FF05:ABCD:0:0:0:0:0:1
Multicast Routing Table
Flags: S - Sparse, B - Bidir Group, s - SSM Group,
C - Connected, L - Local, I - Received Source Specific Host Report,
P - Pruned, R - RP-bit set, F - Register flag, T - SPT-bit set,
J - Join SPT, Y - Joined MDT-data group,
y - Sending to MDT-data group

g - BGP signal originated, G - BGP Signal received,
N - BGP Shared-Tree Prune received, n - BGP C-Mroute suppressed,
q - BGP Src-Active originated, Q - BGP Src-Active received
E - Extranet
Timers: Uptime/Expires
Interface state: Interface, State

(*, FF05:ABCD::1), 00:06:22/never, RP 1010:ABCD::10, flags: SCJ
Incoming interface: Port-channel33
RPF nbr: FE80::2E31:24FF:FE06:134A
Immediate Outgoing interface list:
TenGigabitEthernet4/0/18, Forward, 00:06:22/never

(3232:ABCD::2, FF05:ABCD::1), 00:04:54/00:02:16, flags: SJT
Incoming interface: Port-channel33
RPF nbr: FE80::2E31:24FF:FE06:134A
```

```
Inherited Outgoing interface list:
TenGigabitEthernet4/0/18, Forward, 00:06:22/never
```

This example displays the output of **show ipv6 mfib** command:

```
Switch# show ipv6 mfib vrf vrf1 FF05:ABCD:0:0:0:0:0:1
Entry Flags:    C - Directly Connected, S - Signal, IA - Inherit A flag,
                ET - Data Rate Exceeds Threshold, K - Keepalive
                DDE - Data Driven Event, HW - Hardware Installed
                ME - MoFRR ECMP entry, MNE - MoFRR Non-ECMP entry, MP - MFIB

                MoFRR Primary, RP - MRIB MoFRR Primary, P - MoFRR Primary
                MS  - MoFRR  Entry in Sync, MC - MoFRR entry in MoFRR Client.
I/O Item Flags: IC - Internal Copy, NP - Not platform switched,
                NS - Negate Signalling, SP - Signal Present,
                A - Accept, F - Forward, RA - MRIB Accept, RF - MRIB Forward,

                MA - MFIB Accept, A2 - Accept backup,
                RA2 - MRIB Accept backup, MA2 - MFIB Accept backup

Forwarding Counts: Pkt Count/Pkts per second/Avg Pkt Size/Kbits per second
Other counts:      Total/RPF failed/Other drops
I/O Item Counts:   FS Pkt Count/PS Pkt Count
VRF testvrf1
 (*,FF05:ABCD::1) Flags: C HW
   SW Forwarding: 0/0/0/0, Other: 0/0/0
   HW Forwarding:   295/0/512/0, Other: 0/0/0
   Port-channel33 Flags: A NS
   TenGigabitEthernet4/0/18 Flags: F NS
     Pkts: 0/0
 (3232:ABCD::2,FF05:ABCD::1) Flags: HW
   SW Forwarding: 50/0/512/0, Other: 111/0/111
   HW Forwarding:   4387686/14849/512/59398, Other: 0/0/0
   Port-channel33 Flags: A
   TenGigabitEthernet4/0/18 Flags: F NS
     Pkts: 0/50

Switch#
```

Configuration Examples

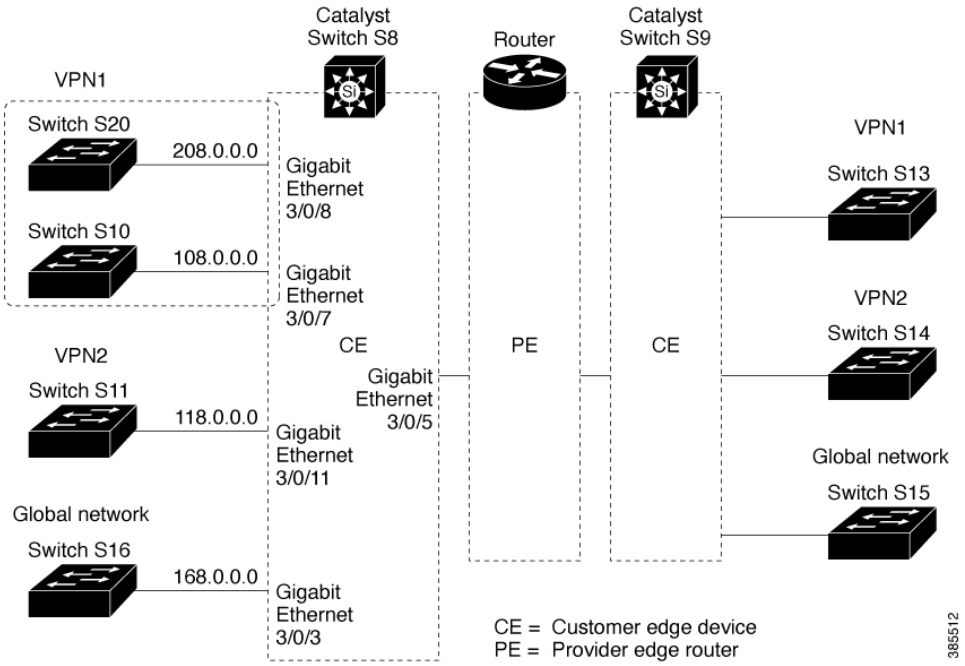
Configuration example for IPv4 VRF-lite

Describes the configuration of IPv4 VRF-lite on switches and PE routers to enable virtual routing and forwarding across multiple VPNs using OSPF and BGP protocols. This configuration isolates traffic between different VPNs while sharing the same physical infrastructure.

This reference provides the configuration steps and command examples for setting up IPv4 VRF-lite to create virtual routing instances that isolate traffic between different VPNs.

OSPF operates in VPN1, VPN2, and the global network. BGP manages the CE to PE connections. The content provides commands to configure the CE switch S8, the VRF setups for switches S20 and S11, and the commands for the PE router that enable traffic with switch S8.

Figure 5: VRF-lite configuration example




```
Switch# configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
Switch(config)# ip routing
Switch(config)# ip vrf v11
Switch(config-vrf)# rd 800:1
Switch(config-vrf)# route-target export 800:1
Switch(config-vrf)# route-target import 800:1
Switch(config-vrf)# exit
Switch(config)# ip vrf v12
Switch(config-vrf)# rd 800:2
Switch(config-vrf)# route-target export 800:2
Switch(config-vrf)# route-
```

Configure the loopback and physical interfaces on switch S8. Fast Ethernet interface 3/5 acts as a trunk connection to the PE. Interfaces 3/7 and 3/11 connect to the VPNs.

```
Switch(config)# interface loopback1
Switch(config-if)# vrf forwarding v11
Switch(config-if)# ip address 8.8.1.8 255.255.255.0
Switch(config-if)# exit

Switch(config)# interface loopback2
Switch(config-if)# vrf forwarding v12
Switch(config-if)# ip address 8.8.2.8 255.255.255.0
Switch(config-if)# exit

Switch(config)# interface GigabitEthernet3/5
Switch(config-if)# switchport trunk encapsulation dot1q
Switch(config-if)# switchport mode trunk
Switch(config-if)# no ip address
Switch(config-if)# exit

Switch(config)# interface GigabitEthernet3/8
Switch(config-if)# switchport access vlan 208
Switch(config-if)# no ip address
Switch(config-if)# exit
```

```
Switch(config)# interface GigabitEthernet3/11
Switch(config-if)# switchport trunk encapsulation dot1q
Switch(config-if)# switchport mode trunk
Switch(config-if)# no ip
```

Configure the VLANs on switch S8. VLAN 10 operates with VRF 11 between the CE and the PE. VLAN 20 operates with VRF 12 between the CE and the PE. VLANs 118 and 208 provide VRF access for VPNs with switch S11 and switch S20, respectively.

```
Switch(config)# interface Vlan10
Switch(config-if)# vrf forwarding v11
Switch(config-if)# ip address 38.0.0.8 255.255.255.0
Switch(config-if)# exit

Switch(config)# interface Vlan20
Switch(config-if)# vrf forwarding v12
Switch(config-if)# ip address 83.0.0.8 255.255.255.0
Switch(config-if)# exit

Switch(config)# interface Vlan118
Switch(config-if)# vrf forwarding v12
Switch(config-if)# ip address 118.0.0.8 255.255.255.0
Switch(config-if)# exit

Switch(config)# interface Vlan208
Switch(config-if)# vrf forwarding v11
Switch(config-if)# ip address 208.0.0.8 255.255.255.0
Switch(config-if)# exit
```

Configure OSPF routing in VPN1 and VPN2.

```
Switch(config)# router ospf 1 vrf vl1
Switch(config-router)# redistribute bgp 800 subnets
Switch(config-router)# network 208.0.0.0 0.0.0.255 area 0
Switch(config-router)# exit

Switch(config)# router ospf 2 vrf vl2
Switch(config-router)# redistribute bgp 800 subnets
Switch(config-router)# network 118.0.0.0 0.0.0.255 area 0
Switch(config-router)# exit
```

Configure BGP for CE to PE routing.

```
Switch(config)# router bgp 800
Switch(config-router)# address-family ipv4 vrf vl2
Switch(config-router-af)# redistribute ospf 2 match internal
Switch(config-router-af)# neighbor 83.0.0.3 remote-as 100
Switch(config-router-af)# neighbor 83.0.0.3 activate
Switch(config-router-af)# network 8.8.2.0 mask 255.255.255.0
Switch(config-router-af)# exit

Switch(config-router)# address-family ipv4 vrf vl1
Switch(config-router-af)# redistribute ospf 1 match internal
Switch(config-router-af)# neighbor 38.0.0.3 remote-as 100
Switch(config-router-af)# neighbor 38.0.0.3 activate
Switch(config-router-af)# network 8.8.1.0 mask 255.255.255.0
Switch(config-router-af)# end
```

Configuring Switch S20

Configure S20 to connect to CE.

```
Switch# configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
Switch(config)# ip routing
Switch(config)# interface Fast Ethernet 0/7
Switch(config-if)# no switchport
Switch(config-if)# ip address 208.0.0.20 255.255.255.0
Switch(config-if)# exitS
witch(config)# router ospf 101
Switch(config-router)# network 208.0.0.0 0.0.0.255 area 0
Switch(config-router)# end
```

Configuring Switch S11

Configure S11 to connect to CE.

```
Switch# configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
Switch(config)# ip routing
Switch(config)# interface Gigabit Ethernet 0/3
Switch(config-if)# switchport trunk encapsulation dot1q
Switch(config-if)# switchport mode trunk
Switch(config-if)# no ip address
Switch(config-if)# exit

Switch(config)# interface Vlan118
Switch(config-if)# ip address 118.0.0.11 255.255.255.0
Switch(config-if)# exit

Switch(config)# router ospf 101
Switch(config-router)# network 118.0.0.0 0.0.0.255 area 0
Switch(config-router)# end
```

Configuring the PE Switch S3

On switch S3 (the router), use these commands to configure only the connections to switch S8.

```
Router# configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
Router(config)# ip vrf v1
Router(config-vrf)# rd 100:1
Router(config-vrf)# route-target export 100:1
Router(config-vrf)# route-target import 100:1
Router(config-vrf)# exit

Router(config)# ip vrf v2
Router(config-vrf)# rd 100:2
Router(config-vrf)# route-target export 100:2
Router(config-vrf)# route-target import 100:2
Router(config-vrf)# exit
```

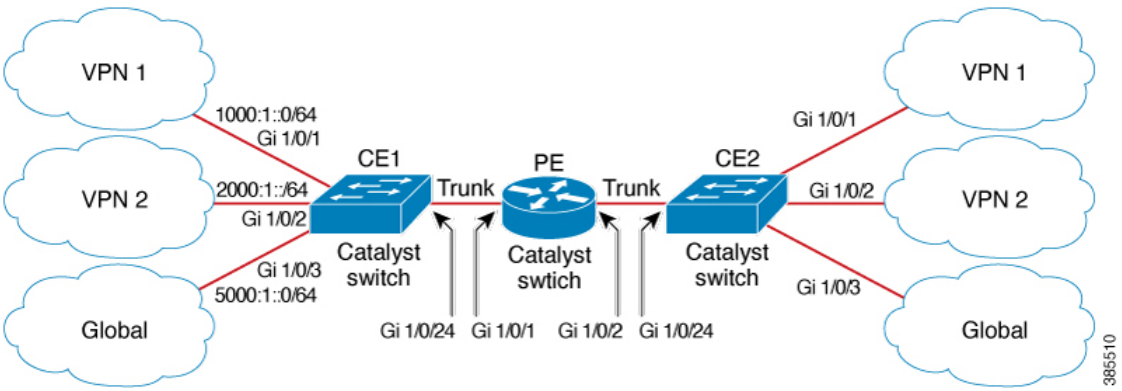
```
Router(config)# ip cef
Router(config)# interface Loopback1
Router(config-if)# vrf forwarding v1
Router(config-if)# ip address 3.3.1.3 255.255.255.0
Router(config-if)# exit

Router(config)# interface Loopback2
Router(config-if)# vrf forwarding v2
Router(config-if)# ip address 3.3.2.3 255.255.255.0
Router(config-if)# exit

Router(config)# interface Fast Ethernet3/0.10
Router(config-if)# encapsulation dot1q 10
Router(config-if)# vrf forwarding v1
Router(config-if)# ip address 38.0.0.3 255.255.255.0
Router(config-if)# exit

Router(config)# interface Fast Ethernet3/0.20
Router(config-if)# encapsulation dot1q 20
Router(config-if)# vrf forwarding v2
Router(config-if)# ip address 83.0.0.3 255.255.255.0
Router(config-if)# exit

Router(config)# router bgp 100
Router(config-router)# address-family ipv4 vrf v2
Router(config-router-af)# neighbor 83.0.0.8 remote-as 800
Router(config-router-af)# neighbor 83.0.0.8 activate
Router(config-router-af)# network 3.3.2.0 mask 255.255.255.0
Router(config-router-af)# exit

Router(config-router)# address-family ipv4 vrf vl
Router(config-router-af)# neighbor 83.0.0.8 remote-as 800
Router(config-router-af)# neighbor 83.0.0.8 activate
Router(config-router-af)# network 3.3.1.0 mask 255.255.255.0
Router(config-router-af)# end
```

Configuration example for IPv6 VRF-lite

Provides a configuration example that illustrates how to use OSPFv3 for CE-PE routing in an IPv6 VRF-lite topology.

This topology illustrates how to use OSPFv3 for CE-PE routing.

Figure 6: VRF-lite configuration example



Configuring CE1 switch

```
ipv6 unicast-routing
vrf definition v1
 rd 100:1
 !
address-family ipv6
 exit-address-family
!

vrf definition v2
 rd 200:1
 !
address-family ipv6
 exit-address-family
!

interface Vlan100
 vrf forwarding v1
 ipv6 address 1000:1::1/64
 ospfv3 100 ipv6 area 0
!

interface Vlan200
 vrf forwarding v2
 ipv6 address 2000:1::1/64
 ospfv3 200 ipv6 area 0
!

interface GigabitEthernet 1/0/1
switchport access vlan 100
end

interface GigabitEthernet 1/0/2
switchport access vlan 200
end

interface GigabitEthernet 1/0/24
 switchport trunk encapsulation dot1q

switchport mode trunk
end

router ospfv3 100
 router-id 10.10.10.10
 !
 address-family ipv6 unicast vrf v1
  redistribute connected
  area 0 normal
 exit-address-family
!

router ospfv3 200
 router-id 20.20.20.20
 !
 address-family ipv6 unicast vrf v2
  redistribute connected
  area 0 normal
 exit-address-family
!
```

Configuring PE switch

```
ipv6 unicast-routing

vrf definition v1
 rd 100:1
 !
address-family ipv6
 exit-address-family
!

vrf definition v2
 rd 200:1
 !
address-family ipv6
 exit-address-family
!
interface Vlan600
 vrf forwarding v1
 no ipv6 address
 ipv6 address 1000:1::2/64
 ospfv3 100 ipv6 area 0
!

interface Vlan700
 vrf forwarding v2
 no ipv6 address
 ipv6 address 2000:1::2/64
 ospfv3 200 ipv6 area 0
!

interface Vlan800
 vrf forwarding v1
 ipv6 address 3000:1::7/64
 ospfv3 100 ipv6 area 0
!
interface Vlan900
 vrf forwarding v2
 ipv6 address 4000:1::7/64
 ospfv3 200 ipv6 area 0
!

interface GigabitEthernet 1/0/1
 switchport trunk encapsulation dot1q
 switchport mode trunk
 exit

interface GigabitEthernet 1/0/2
 switchport trunk encapsulation dot1q

switchport mode trunk
 exit

router ospfv3 100
 router-id 30.30.30.30
 !
 address-family ipv6 unicast vrf v1
  redistribute connected
  area 0 normal
 exit-address-family
 !
address-family ipv6 unicast vrf v2
  redistribute connected
```

```
  area 0 normal
 exit-address-family
 !
```

Configuring CE2 switch

```
ipv6 unicast-routing

vrf definition v1
 rd 100:1
 !
address-family ipv6
 exit-address-family
!

vrf definition v2
 rd 200:1
 !
address-family ipv6
 exit-address-family
!

interface Vlan100
 vrf forwarding v1

ipv6 address 1000:1::3/64
 ospfv3 100 ipv6 area 0
!

interface Vlan200
 vrf forwarding v2
 ipv6 address 2000:1::3/64
 ospfv3 200 ipv6 area 0
!

interface GigabitEthernet 1/0/1
switchport access vlan 100
end

interface GigabitEthernet 1/0/2
switchport access vlan 200
end

interface GigabitEthernet 1/0/24
 switchport trunk encapsulation dot1q
 switchport mode trunk
 end

router ospfv3 100
 router-id 40.40.40.40
 !
 address-family ipv6 unicast vrf v1
  redistribute connected
  area 0 normal
 exit-address-family
!

router ospfv3 200
 router-id 50.50.50.50
 !
 address-family ipv6 unicast vrf v2
```

```
  redistribute connected

area 0 normal
 exit-address-family
!
```

6 Configure BGP

Topics:

Border Gateway Protocol

Describes the Border Gateway Protocol used for interdomain routing in the Internet.

The Border Gateway Protocol (BGP) is an exterior gateway protocol. It sets up an interdomain routing system that guarantees the loop-free exchange of routing information between autonomous systems. BGP serves as the standard EGP for interdomain routing in the Internet. RFCs 1163, 1267, and 1771 define BGP Version 4.

Autonomous systems and gateway protocols

Autonomous systems consist of routers that operate under the same administration. They run Interior Gateway Protocols (IGPs), such as RIP or OSPF, within their boundaries. These routers interconnect by using an Exterior Gateway Protocol (EGP).

BGP network topologies

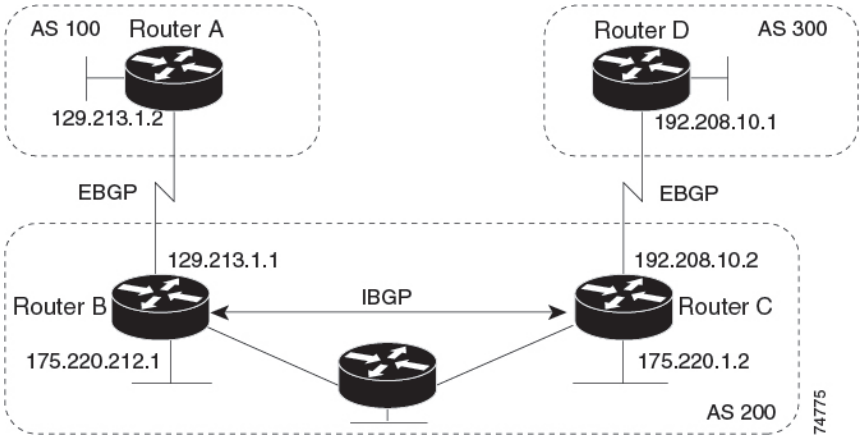
Describes the network topology structures that BGP uses for routing updates between autonomous systems.

A BGP network topology is a routing structure that determines whether routing updates are exchanged internally within an autonomous system (AS) or externally between different autonomous systems. It establishes peer relationships between BGP speakers using TCP connections for routing information exchange. It also constructs loop-free maps of autonomous systems using AS path information.

BGP topology characteristics

Routers that belong to the same autonomous system (AS) and that exchange BGP updates run internal BGP (IBGP), and routers that belong to different autonomous systems and that exchange BGP updates run external BGP (EBGP). Most configuration commands are the same for configuring EBGP and IBGP. The difference is that the routing updates are exchanged either between autonomous systems (EBGP) or within an AS (IBGP).

Figure 7: EBGP, IBGP, and multiple autonomous systems



Before exchanging information with an external AS, BGP ensures that networks within the AS can be reached. It does this by defining internal BGP peering among routers within the AS and by redistributing BGP routing information to IGPs that run within the AS, such as IGRP and OSPF.

Routers that run a BGP routing process are often referred to as BGP speakers. BGP uses the Transmission Control Protocol (TCP), specifically port 179, as its transport protocol. Two BGP speakers that exchange routing information over a TCP connection are known as peers or neighbors. The routing information is a series of AS numbers that describe the full path to the destination network. BGP uses this information to construct a loop-free map of autonomous systems.

Network topology characteristics

BGP network topologies have these characteristics:

- EBGP peers are typically directly connected, while IBGP peers do not have to be directly connected. As long as there is an IGP running that allows the two neighbors to reach one another, IBGP peers can establish connections.
- All BGP speakers within an AS must establish a peer relationship with each other. That is, the BGP speakers within an AS must be fully meshed logically. BGP4 provides two techniques that reduce the requirement for a logical full mesh: confederations and route reflectors.
- Transit autonomous systems can be used to transfer packets between other autonomous systems.

Peering communication and update exchange

BGP peers initially exchange their full BGP routing tables and then send only incremental updates. BGP peers also exchange keepalive messages (to ensure that the connection is up) and notification messages (in response to errors or special conditions).

Route composition and path policy

In BGP, each route consists of a network number, a list of autonomous systems through which information has passed (the autonomous system path), and a list of other path attributes. The primary function of a BGP system is to exchange network reachability information, including information about the list of AS paths, with other BGP systems. This information can be used to determine AS connectivity, to prune routing loops, and to enforce AS-level policy decisions.

Route selection and IGP synchronization

A router or device running Cisco IOS does not select or use an IBGP route unless it has a route available to the next-hop router and it has received synchronization from an IGP (unless IGP synchronization is disabled). When multiple routes are available, BGP bases its path selection on attribute values.

CIDR support and routing table aggregation

BGP Version 4 supports classless interdomain routing (CIDR) so you can reduce the size of your routing tables by creating aggregate routes, resulting in supernets. CIDR eliminates the concept of network classes within BGP and supports the advertising of IP prefixes.

Nonstop forwarding awareness

Explains how nonstop forwarding awareness allows a layer 3 device to forward packets between the interval when a primary RP reloads, or fails and the secondary RP takes over.

Nonstop forwarding awareness is a network feature that allows a Layer 3 device to continue forwarding packets during Route Processor (RP) transitions. The device forwards packets during the interval when a primary RP reloads or fails and the secondary RP takes over. This feature requires Graceful Restart to be enabled with BGP routing. It operates when the neighboring router is NSF-capable during RP failover or during manual reloads for nondisruptive software upgrades.

BGP routing requirements

To enable this feature with BGP routing, you need to enable Graceful Restart. When the neighboring router is NSF-capable and this feature is enabled, the Layer 3 device continues to forward packets from the neighboring router. It forwards packets during the interval when the primary Route Processor (RP) in a device is failing and the backup RP is taking over. It also continues forwarding while the primary RP is manually reloaded for a nondisruptive software upgrade.

BGP routing

Describes the foundational requirements and characteristics of BGP routing configuration.

BGP routing is a network protocol configuration that requires establishing a BGP routing process with defined local networks. The configuration must specify BGP neighbors to recognize network relationships. This protocol supports both internal neighbors (within the same AS) and external neighbors (in different autonomous systems).

BGP neighbor types and private AS support

BGP supports two kinds of neighbors: internal and external. Internal neighbors are in the same AS; external neighbors are in different autonomous systems. External neighbors are usually adjacent and share a subnet. Internal neighbors can be located anywhere within the same AS.

Private AS numbers and AS path manipulation

The switch supports the use of private AS numbers, usually assigned by service providers and given to systems whose routes are not advertised to external neighbors. Private AS numbers range from 64512 to 65535. You can configure external neighbors to remove private AS numbers from the AS path by using the **neighbor remove-private-AS** router configuration command. Then when an update is passed to an external neighbor, if the AS path includes private AS numbers, these numbers are dropped.

BGP and IGP synchronization

If your AS will be passing traffic between other autonomous systems, it is important to be consistent about the routes it advertises. To avoid routing issues, BGP should wait until IGP has propagated route information across the AS so that BGP and IGP remain synchronized. Synchronization is enabled by default.

If your AS does not pass traffic from one AS to another AS, or if all routers in your autonomous system are running BGP, you can disable synchronization. Disabling synchronization allows your network to carry fewer routes in the IGP and lets BGP converge more quickly.

Routing policy changes

Explains the requirement for hardware or software resets when routing policy changes are made in BGP configurations.

Routing policy changes are BGP configuration modifications that affect inbound or outbound routing table updates between BGP neighbors. They include changes to BGP filters, weights, distances, versions, or timers, and these changes require either a hardware or software reset to take effect.

BGP reset types

When you have defined two routers as BGP neighbors, they form a BGP connection and exchange routing information. If you later change a BGP filter, weight, distance, version, or timer, or make a similar configuration change, you must reset the BGP sessions so that the configuration changes take effect.

Hard and soft BGP resets

There are two types of reset: hard reset and soft reset. To use a soft reset without preconfiguration, both BGP peers must support the soft route refresh capability, which is advertised in the OPEN message sent when the peers establish a TCP session. A soft reset allows the dynamic exchange of route refresh requests and routing information between BGP routers and the subsequent re-advertisement of the respective outbound routing table.

Soft reset types:

- Dynamic inbound soft reset: When soft reset generates inbound updates from a neighbor
- Outbound soft reset: When soft reset sends a set of updates to a neighbor

A soft inbound reset causes the new inbound policy to take effect. A soft outbound reset causes the new local outbound policy to take effect without resetting the BGP session. As a new set of updates is sent during outbound policy reset, a new inbound policy can also take effect.

This table lists the advantages and disadvantages of hard reset and soft reset.

Table 6: Advantages and disadvantages of hard and soft resets

Type of reset	Advantages	Disadvantages
Hard reset	No memory overhead	The prefixes in the BGP, IP, and FIB tables provided by the neighbor are lost. Not recommended.

Type of reset	Advantages	Disadvantages
Outbound soft reset	No configuration, no storing of routing table updates	Does not reset inbound routing table updates.
Dynamic inbound soft reset	Does not clear the BGP session and cache Does not require storing of routing table updates and has no memory overhead	Both BGP routers must support the route refresh capability (in Cisco IOS Release 12.1 and later).

BGP decision attributes

Describes the attributes that BGP evaluates for choosing the best path based on multiple factors in a certain order.

BGP decision attributes are evaluation criteria that determine the single best path when a BGP speaker receives updates from multiple autonomous systems describing different paths to the same destination, control how the selected path is entered into the BGP routing table and propagated to neighbors, and include values from the update contents and other BGP-configurable factors.

BGP decision attributes are evaluation criteria that determine the single best path when a BGP speaker receives updates from multiple autonomous systems that describe different paths to the same destination. These attributes also control how the selected path is entered into the BGP routing table and propagated to neighbors. BGP decision attributes include values from the update contents and other BGP-configurable factors.

BGP path evaluation order

When a BGP peer learns two EBGP paths for a prefix from a neighboring AS, it chooses the best path and inserts that path in the IP routing table. If BGP multipath support is enabled and the EBGP paths are learned from the same neighboring autonomous systems, instead of a single best path, multiple paths are installed in the IP routing table. Then, during packet switching, per-packet or per-destination load-balancing is performed among the multiple paths. The **maximum-paths** router configuration command controls the number of paths allowed.

These factors summarize the order in which BGP evaluates the attributes for choosing the best path:

1. If the path specifies a next hop that is inaccessible, drop the update. The BGP next-hop attribute, automatically determined by the software, is the IP address of the next hop that is going to be used to reach a destination. For EBGP, this is usually the IP address of the neighbor that is specified by the **neighbor remote-AS router** configuration command. You can disable next-hop processing by using route maps or the **neighbor next-hop-self** router configuration command.
2. Prefer the path with the largest weight (a Cisco proprietary parameter). The weight attribute is local to the router and not propagated in routing updates. By default, the weight attribute is 32768 for paths that the router originates and zero for other paths. Routes with the largest weight are preferred. You can use access lists, route maps, or the **neighbor weight** router configuration command to set weights.
3. Prefer the route with the highest local preference. Local preference is part of the routing update and exchanged among routers in the same AS. The default value of the local preference attribute is 100. You can set local preference by using the **BGP default local-preference** router configuration command or by using a route map.
4. Prefer the route that was originated by BGP running on the local router, the route with the shortest AS path, and the external (EBGP) path over the internal (IBGP) path.
5. Prefer the route with the lowest origin type. An interior route or IGP is lower than a route learned by EGP, and an EGP-learned route is lower than one of unknown origin or learned in another way.
6. Prefer the route with the lowest multi-exit discriminator (MED) metric attribute if the neighboring AS is the same for all routes considered. You can configure the MED by using route maps or by using the **default-metric** router configuration command. When an update is sent to an IBGP peer, the MED is included.
7. Prefer the route that can be reached through the closest IGP neighbor (the lowest IGP metric). This means that the router will prefer the shortest internal path within the AS to reach the destination (the shortest path to the BGP next-hop).

8. If the conditions are all true, insert the route for this path into the IP routing table:
   - Both the best route and this route are external.
   - Both the best route and this route are from the same neighboring autonomous system.
   - Maximum-paths is enabled.
9. If multipath is not enabled, prefer the route with the lowest IP address value for the BGP router ID. The router ID is usually the highest IP address on the router or the loopback (virtual) address, but might be implementation-specific.

Route maps

Describes how route maps control and modify routing information and define the conditions by which routes are redistributed between routing domains.

A route map is a BGP mechanism that controls and modifies routing information, defines the conditions by which routes are redistributed between routing domains, and has a name that identifies the route map (*map tag*) and an optional sequence number.

BGP filtering

Explains how to filter BGP advertisements using AS-path filters, access lists, or route maps.

BGP filtering is a network control mechanism that filters BGP advertisements using AS-path filters, access lists, or route maps. It controls routing updates, modifies various attributes on a per-neighbor basis, and applies matching criteria based on AS path, community, and network numbers.

BGP filtering methods

BGP filtering can be implemented using these methods:

- AS-path filters using the **AS-path access-list** global configuration command and the **neighbor filter-list** router configuration command
- Access lists with the **neighbor distribute-list** router configuration command, where distribute-list filters are applied to network numbers
- Route maps filter updates and modify various attributes on a per-neighbor basis.

Route maps can be applied to either inbound or outbound updates. Only the routes that pass the route map are sent or accepted in updates. On both inbound and outbound updates, matching is supported based on AS path, community, and network numbers.

- Autonomous system path matching: Uses the **match AS-path access-list** route-map command
- Community based matching: Uses the **match community-list** route-map command
- Network-based matching: Uses the **ip access-list** global configuration command

Prefix lists for BGP filtering

Describes filtering by a prefix list that involves matching the prefixes of routes with those listed in the prefix list.

A prefix list is a BGP route filtering mechanism that matches the prefixes of routes with those listed in the prefix list. It improves performance for loading and searching large lists, offers incremental update support, allows for easier CLI configuration, and provides greater flexibility. The prefix list also serves as an alternative to access lists in many BGP route filtering commands, such as the **neighbor distribute-list router** configuration command.

Prefix list matching rules

Whether a prefix is permitted or denied is based on these rules.

- An empty prefix list permits all prefixes.
- An implicit deny is assumed if a given prefix does not match any entries in a prefix list.
- When multiple entries of a prefix list match a given prefix, the sequence number of a prefix list entry identifies the entry with the lowest sequence number.

By default, sequence numbers are generated automatically and incremented in units of five. If you disable the automatic generation of sequence numbers, you must specify the sequence number for each entry. You can specify sequence values in any increment. If you specify increments of one, you cannot insert additional entries into the list; if you choose large increments, you might run out of values.

BGP community filtering

Describes BGP community attribute that allows to group destinations into communities and to apply routing decisions based on the communities.

BGP community filtering is a routing control mechanism. It groups destinations into communities based on the COMMUNITIES attribute. It applies routing decisions based on the communities and simplifies the configuration of a BGP speaker to control the distribution of routing information.

BGP community attributes

A community is a group of destinations that share some common attribute. Each destination can belong to multiple communities. AS administrators can define to which communities a destination belongs to. By default, all destinations belong to the general Internet community. The community is identified by the COMMUNITIES attribute, an optional, transitive, global attribute in the numerical range from 1 to 4294967200. These are some predefined, well-known communities:

- **internet**: Advertise this route to the Internet community. All routers belong to it.
- **no-export**: Do not advertise this route to EBGP peers.
- **no-advertise**: Do not advertise this route to any peer (internal or external).
- **local-AS**: Do not advertise this route to peers outside the local autonomous system.

Based on the community, you can control which routing information to accept, prefer, or distribute to other neighbors. A BGP speaker can set, append, or modify the community of a route when learning, advertising, or redistributing routes. When routes are aggregated, the resulting aggregate has a communities attribute that contains all communities from all the initial routes.

You can use community lists to create groups of communities to use in a match clause of a route map. AS with an access list, a series of community lists can be created. Statements are checked until a match is found. AS soon AS one statement is satisfied, the test is concluded.

BGP neighbors and peer groups

Explains how BGP neighbors with the same update policies can be grouped into peer groups to simplify configuration and to make updating more efficient.

A BGP peer group is a configuration grouping mechanism that groups BGP neighbors with the same update policies to simplify configuration. It improves update efficiency when many peers are configured. Members can inherit all configuration options, such as remote-as, version, update-source, out-route-map, out-filter-list, out-dist-list, minimum-advertisement-interval, and next-hop-self.

BGP peer group configuration

To configure a BGP peer group, you create the peer group, assign options to the peer group, and add neighbors as peer group members. You configure the peer group by using the **neighbor** router configuration commands.

By default, peer group members inherit all configuration options of the peer group. These options include remote-as (if configured), version, update-source, out-route-map, out-filter-list, out-dist-list, minimum-advertisement-interval, and next-hop-self.

All peer group members also inherit changes that are made to the peer group. Members can also be configured to override the options that do not affect outbound updates.

Aggregate routes

Explains how aggregate routes minimize the size of routing tables.

An aggregate route is a CIDR routing mechanism that minimizes the size of routing tables by creating supernets. You can configure an aggregate route in BGP by redistributing an aggregate route into BGP or by creating an aggregate entry in the BGP routing table. The aggregate route is added to the BGP table when there is at least one more specific entry in the BGP table.

CIDR implementation

Classless interdomain routing (CIDR) enables you to create aggregate routes (or supernets) to minimize the size of routing tables.

Routing domain confederations

Explains how confederations reduce the IBGP mesh by grouping multiple sub-autonomous systems into a single autonomous system.

A routing domain confederation is a BGP architecture technique that reduces the IBGP mesh by dividing an autonomous system into multiple sub-autonomous systems. This technique groups these sub-autonomous systems into a single confederation that appears as a single autonomous system, and preserves routing information such as next hop, MED, and local preference between peers in different autonomous systems.

Confederation architecture details

Each autonomous system is fully meshed within itself and has a few connections to other autonomous systems in the same confederation. Peers in different autonomous systems have EBGP sessions but exchange routing information as if they were IBGP peers. Specifically, the next hop, MED, and local preference information are preserved. You can then use a single IGP for all of the autonomous systems.

BGP route reflectors

Explains BGP route reflectors that allow passing learned routes to neighbors without the need to be fully meshed.

A BGP route reflector is a BGP router that passes IBGP learned routes to a set of IBGP neighbors, eliminating the need for all IBGP speakers to be fully meshed. It divides internal peers into client peers and nonclient peers, reflects routes between these groups, and forms a cluster with its client peers. The clients in this cluster do not need to be fully meshed with each other.

Route reflection behavior

BGP requires that all of the IBGP speakers be fully meshed. When a router receives a route from an external neighbor, it must advertise it to all internal neighbors. To prevent a routing information loop, all IBGP speakers must be connected. The internal neighbors do not send routes that are learned from internal neighbors to other internal neighbors.

With route reflectors, IBGP speakers do not need to be fully meshed. Route reflectors use a method to pass learned routes to neighbors. When you configure an internal BGP peer to be a route reflector, it is responsible for passing IBGP learned routes to a set of IBGP neighbors. The internal peers of the route reflector are divided into two groups: client peers and nonclient peers. The nonclient peers include all other routers in the autonomous system. The route reflector reflects routes between these two groups. The route reflector and its client peers form a cluster. The nonclient peers must be fully meshed with each other. The client peers do not need to be fully meshed. The clients in the cluster do not communicate with IBGP speakers outside their cluster.

When the route reflector receives an advertised route, it takes one of these actions, depending on the neighbor:

- A route from an external BGP speaker is advertised to all clients and nonclient peers.
- A route from a nonclient peer is advertised to all clients.
- A route from a client is advertised to all clients and nonclient peers. Hence, the clients need not be fully meshed.

Typically, a cluster of clients has a single route reflector. The cluster is identified by the route reflector router ID. To increase redundancy and to avoid a single point of failure, a cluster might have more than one route reflector. In this case, all route reflectors in the cluster must be configured with the same 4-byte cluster ID. This configuration enables a route reflector to recognize updates from other route reflectors in the same cluster. All route reflectors serving a cluster should be fully meshed. They should also have identical sets of client and nonclient peers.

Route dampening

Describes a BGP feature designed to minimize the propagation of flapping routes across an internetwork.

Route flap dampening is a BGP feature that minimizes the propagation of flapping routes across an internetwork. It assigns a numeric penalty value to a route when it flaps, and suppresses advertisements of the route when accumulated penalties reach a configurable limit.

Route dampening operation

A route is considered to be flapping when its status repeatedly alternates between available and unavailable. When route dampening is enabled, the system assigns a numeric penalty value to a route each time it flaps. If a route's accumulated penalties reach a configurable limit, BGP suppresses advertisements of the route, even if it is operational. The system compares the penalty value with a configurable reuse limit. If the penalty falls below the reuse limit, BGP advertises the suppressed route again if it is available.

Dampening is not applied to routes that are learned by IBGP. This policy prevents the IBGP peers from having a higher penalty for routes external to the AS.

Conditional BGP route injection

Describes a BGP feature that allows you to originate a prefix into a BGP routing table without the corresponding match.

Conditional BGP route injection is a BGP feature that allows you to originate a prefix into a BGP routing table without the corresponding match. It enables the generation of more specific routes based on administrative policy or traffic engineering information. It also provides greater control over packet forwarding by allowing conditional injection or replacement of less specific prefixes with more specific prefixes.

Implementation details

Routes that are advertised through the BGP are commonly aggregated to minimize the number of routes that are used and reduce the size of global routing tables. However, common route aggregation can obscure more specific routing information that is more accurate but not necessary to forward packets to their destinations. Routing accuracy is obscured by common route aggregation. A prefix that represents multiple addresses or hosts across a large topological area cannot be accurately represented in a single route.

Prefix origination in Cisco software

Cisco software provides several methods by which you can originate a prefix into BGP. Prior to the BGP conditional route injection feature, the existing methods included redistribution and using the **network** or **aggregate-address** command. However, these methods assume the existence of more specific routing information (matching the route to be originated) in either the routing table or the BGP table.

BGP conditional route injection

More specific prefixes are injected into the BGP routing table only if the configured conditions are met. Only prefixes that are equal to or more specific than the original prefix may be injected. You enable BGP conditional route injection with the **BGP inject-map exist-map** command. The feature uses two route maps (inject map and exist map) to install one or more more-specific prefixes into a BGP routing table. The exist map specifies the prefixes that the BGP speaker will track. The inject map defines the prefixes that will be created and installed into the local BGP table.

> **Note**
>
> Inject maps and exist maps will only match a single prefix per route map clause. To inject additional prefixes, you must configure additional route map clauses. If multiple prefixes are used, the first prefix that is matched will be used.

BGP peer templates

Explains BGP peer templates that allow the network operator to define complex configuration patterns through the capability of a peer template to inherit a configuration from another peer template.

A BGP peer template is a configuration pattern that can be applied to neighbors that share policies. This pattern is reusable and supports inheritance. Network operators can group and apply distinct neighbor configurations for BGP neighbors with shared policies. With peer templates, operators can also define complex configuration patterns because a peer template can inherit configurations from another peer template.

BGP peer template types and characteristics

To address some of the limitations of peer groups such as configuration management, BGP peer templates were introduced to support the BGP update group configuration.

There are two types of peer templates:

- Peer session templates are used to group and apply the configuration of general session commands that are common to all address family and NLRI configuration modes.
- Peer policy templates are used to group and apply the configuration of commands that are applied within specific address families and NLRI configuration modes.

Benefits of BGP peer templates

Peer templates improve the flexibility and capability of neighbor configuration. They also provide an alternative to peer group configuration and address certain limitations of peer groups. When BGP peer devices use peer templates, they can benefit from automatic update group configuration. With BGP peer templates and BGP dynamic update peer groups, network operators no longer need to configure peer groups in BGP. The network benefits from improved configuration flexibility and faster convergence.

> **Note**
>
> A BGP neighbor cannot be configured to work with both peer groups and peer templates. A BGP neighbor can be configured to belong only to a peer group or to inherit policies from peer templates.

Peer template restrictions

Restrictions apply to the peer policy templates:

- A peer policy template can directly or indirectly inherit up to eight peer policy templates.
- A BGP neighbor cannot be configured to work with both peer groups and peer templates. A BGP neighbor can be configured to belong only to a peer group or to inherit policies only from peer templates.

Inheritance in peer templates

Describes how a peer template can directly or indirectly inherit the configuration from another peer template.

Inheritance in peer templates is a configuration capability that allows a peer template to directly or indirectly inherit configuration from another peer template. This feature uses node and tree structures similar to file and directory trees in general computing. It eliminates the need to repeat configuration statements by allowing common configurations to be applied once and inherited by multiple templates. This approach also expands the scalability and flexibility of neighbor configuration by chaining peer template configurations.

Inheritance structure and operation

The inheritance capability is a key component of peer template operation. The directly inherited peer template represents the trunk or root of the structure, while the indirectly inherited peer template represents a node on the tree. Because each node also supports inheritance, branches can be created that apply the configurations of all indirectly inherited peer templates within a chain back to the directly inherited peer template or the source of the tree.

Configuration statements that are duplicated separately within a node and a tree are filtered out at the source of the tree by the directly inherited template. A directly inherited template overwrites any indirectly inherited statements that are duplicated in the directly inherited template.

Scalability and configuration flexibility

Inheritance expands the scalability and flexibility of neighbor configuration by letting you chain peer template configurations. You can create simple configurations that inherit common configuration statements or complex configurations that apply specific statements in addition to inherited configurations.

Verifying inherited policies and configuration details

When BGP neighbors use inherited peer templates, it can be difficult to determine which policies are associated with a specific template. The **detail** keyword of the **show ip BGP template peer-policy** command displays the detailed configuration of local and inherited policies that are associated with a specific template.

Peer session templates

Describes peer session templates used to group and apply the configuration of general session commands to groups of neighbors that share session configuration elements.

A peer session template is a BGP configuration feature that

- groups and applies the configuration of general session commands to groups of neighbors that share session configuration elements
- supports only general session commands that are common for neighbors configured in different address families, and
- simplifies the configuration of general session commands commonly applied to all neighbors within an autonomous system.

Supported commands

Peer session templates are created and configured in peer session configuration mode. The general session commands supported by peer session templates are:

- **description**
- **disable-connected-check**
- **ebgp-multihop**
- **exit peer-session**
- **inherit peer-session**
- **local-as**

- **password**
- **remote-as**
- **shutdown**
- **timers**
- **translate-update**
- **update-source**
- **version**

General session commands can be configured once in a peer session template and then applied to many neighbors through the direct application of a peer session template or through indirect inheritance from a peer session template.

Session template inheritance and constraints

Peer session templates support direct and indirect inheritance. A peer can be configured with only one peer session template at a time. That peer session template can contain only one indirectly inherited peer session template.

> **Note**
>
> If you attempt to configure more than ONE inherit statement with a single peer session template, an error message will be displayed.

This behavior allows a BGP neighbor to directly inherit only ONE session template and indirectly inherit up to seven additional peer session templates. This allows you to apply up to eight peer session configurations to a neighbor: one configuration from the directly inherited peer session template and up to seven configurations from indirectly inherited peer session templates.

Evaluation order and configuration precedence

Inherited peer session configurations are evaluated first. They are applied starting with the last node in the branch and ending with the directly applied peer session template configuration at the source of the tree. The directly applied peer session template has priority over inherited peer session template configurations.

Any configuration statements that are duplicated in inherited peer session templates will be overwritten by the directly applied peer session template. So, if a general session command is reapplied with a different value, the subsequent value will have priority and overwrite the previous value that was configured in the indirectly inherited template.

Command scope and configuration example

Peer session templates support only general session commands. BGP policy configuration commands that are configured only for a specific address family or NLRI configuration mode are configured with peer policy templates.

In this example, the general session command **remote-as 1** is applied in the peer session template named session-template-ONE:

```
template peer-session session-template-ONE
 remote-as 1
 exit peer-session
```

Peer policy templates

Describes BGP policy configuration templates that group and apply commands within specific address families and NLRI configuration mode.

A peer policy template is a BGP configuration template that

- groups and applies BGP policy commands within specific address families and NLRI configuration mode,
- is created and configured in peer policy configuration mode, and
- supports inheritance and can be applied to multiple neighbors or neighbor groups.

Supported BGP policy commands

Peer policy templates support these BGP policy commands:

- **advertisement-interval**
- **allowas-in**
- **as-override**
- **capability**
- **default-originate**
- **distribute-list**
- **dmzlink-bw**
- **exit-peer-policy**
- **filter-list**
- **inherit peer-policy**
- **maximum-prefix**
- **next-hop-self**
- **next-hop-unchanged**
- **prefix-list**
- **remove-private-as**
- **route-map**
- **route-reflector-client**
- **send-community**
- **send-label**
- **soft-reconfiguration**
- **unsuppress-map**
- **weight**

Peer policy templates configure BGP policy commands for neighbors in specific address families. Like peer session templates, peer policy templates are configured once and then applied to many neighbors either directly or through inheritance. Peer policy templates simplify applying BGP policy commands to all neighbors within an autonomous system.

Policy template inheritance and evaluation mechanics

Like a peer session template, a peer policy template supports inheritance; however, minor differences exist.

A directly applied peer policy template can directly or indirectly inherit configurations from up to seven peer policy templates. So, a total of eight peer policy templates can be applied to a neighbor or neighbor group. Like route maps, inherited peer policy templates are configured with sequence numbers. Also like a route map, an inherited peer policy template is evaluated starting with the **inherit peer-policy** statement that has the lowest sequence number and ending with the highest sequence number.

However, a peer policy template does not collapse like a route map. Every sequence is evaluated. If a BGP policy command is reapplied with a different value, it overwrites any previous value from a lower sequence number.

Precedence and priority rules

The directly applied peer policy template and the **inherit peer-policy** statement with the highest sequence number always have priority and are applied last. Commands that are reapplied in subsequent peer templates will always overwrite the previous values. This behavior is designed to allow you to apply common policy configurations to large neighbor groups and specific policy configurations only to certain neighbors and neighbor groups without duplicating individual policy configuration commands.

Command scope and configuration flexibility

Peer policy templates support only policy configuration commands. BGP policy configuration commands that are configured only for specific address families are configured with peer policy templates.

The configuration of peer policy templates simplifies and improves the flexibility of BGP configuration. A specific policy can be configured once and referenced many times. Because a peer policy supports up to eight levels of inheritance, very specific and very complex BGP policies can also be created.

How to Configure BGP

Default BGP configuration

Lists the default BGP configuration settings for the router. This information helps administrators understand the baseline state of BGP features before applying custom network policies.

Provides a comprehensive reference table detailing the default settings for various BGP features and parameters.

Table 7: Default BGP configuration

Feature	Default setting
Aggregate address	Disabled: none defined.
AS path access list	None defined.
Auto summary	Disabled.
Best path	• The router considers *AS-path* in choosing a route and does not compare similar routes from external BGP peers. • Compare router ID: Disabled.
BGP community list	• Number: None defined. When you permit a value for the community number, the list defaults to an implicit deny for everything else that has not been permitted. • Format: Cisco default format (32-bit number).

Feature	Default setting
BGP confederation identifier/peers	• Identifier: none configured. • Peers: none identified.
BGP Fast external fallover	Enabled.
BGP local preference	100. The range is 0 to 4294967295 with the higher value preferred.
BGP network	No value is specified and no backdoor route is advertised.
BGP route dampening	Disabled by default. When enabled: • The half-life is 15 minutes. • Re-use is 750 (10-second increments). • Suppress is 2000 (10-second increments). • Max-suppress-time is four times the half-life, which is 60 minutes.
BGP router ID	The IP address of a loopback interface if one is configured or the highest IP address configured for a physical interface on the router.
Default information originate (protocol or network redistribution)	Disabled.
Default metric	The router uses built-in, automatic metric translations.
Distance	• External route administrative distance: 20 (acceptable values are from 1 to 255). • Internal route administrative distance: 200 (acceptable values are from 1 to 255). • Local route administrative distance: 200 (acceptable values are from 1 to 255).
Distribute list	• In (filter networks received in updates): Disabled. • Out (suppress networks from being advertised in updates): Disabled.
Internal route redistribution	Disabled.
IP prefix list	None defined.
Multi exit discriminator (MED)	• Always compare: Disabled. Does not compare MEDs for paths from neighbors in different autonomous systems. • Best path compare: Disabled. • MED missing AS worst path: Disabled. • Deterministic MED comparison is disabled.

Feature	Default setting
Neighbor	• Advertisement interval: 30 seconds for external peers; 5 seconds for internal peers. • Change logging: Enabled. • Conditional advertisement: Disabled. • Default originate: No default route is sent to the neighbor. • Description: None. • Distribute list: None defined. • External BGP multihop: Only directly connected neighbors are allowed. • Filter list: None used. • Maximum number of prefixes received: No limit. • Next hop (router AS next hop for BGP neighbor): Disabled. • Password: Disabled. • Peer group: None defined; no members assigned. • Prefix list: None specified. • Remote AS (add entry to neighbor BGP table): No peers defined. • Private AS number removal: Disabled. • Route maps: None applied to a peer. • Send community attributes: None sent to neighbors. • Shutdown or soft reconfiguration: Not enabled. • Timers: keepalive: 60 seconds; holdtime: 180 seconds. • Update source: Best local address. • Version: BGP Version 4. • Weight: Routes learned through BGP peer: 0; routes sourced by the local router: 32768.
NSF Awareness	Disabled. If enabled, allows Layer 3 switches to continue forwarding packets from a neighboring NSF-capable router during hardware or software changes.
Route reflector	None configured.
Synchronization (BGP and IGP)	Disabled.
Table map update	Disabled.
Timers	Keepalive: 60 seconds; holdtime: 180 seconds.

Enable BGP routing

Configure BGP routing to enable dynamic routing protocol functionality on your network device.

This task enables BGP (Border Gateway Protocol) routing on a network device to facilitate dynamic routing between autonomous systems and within the network infrastructure.

BGP routing is essential for networks that need to exchange routing information with external networks or manage complex internal routing scenarios. This configuration establishes the basic BGP parameters and neighbor relationships.

1. Use the **configure terminal** command to enter global configuration mode.

```
Device# configure terminal
```

2. Use the **ip routing** command to enable IP routing.

```
Device(config)# ip routing
```

3. Use the **router bgp** *autonomous-system* command to enable a BGP routing process, assign it an AS number, and enter router configuration mode.

```
Device(config)# router bgp 45000
```

   The AS number can be from 1 to 65535, with 64512 to 65535 designated AS private autonomous numbers.

4. Use the **network** *network-number* [**mask** *network-mask*] [**route-map** *route-map-name*] command to configure a network AS local to this AS, and enter it in the BGP table.

```
Device(config)# network 192.0.2.0
```

5. Use the **neighbor** {*ip-address* | *peer-group-name*} **remote-as** *number* command to add an entry to the BGP neighbor table specifying that the neighbor identified by the IP address belongs to the specified AS.

```
Device(config)# neighbor 192.0.2.2 remote-as 65200
```

   For EBGP, neighbors are usually directly connected, and the IP address is the address of the interface at the other end of the connection.

   For IBGP, the IP address can be the address of any of the router interfaces.

6. Perform these optional steps, if required.
   a) Use the **neighbor** {*ip-address* | *peer-group-name*} **remove-private-as** command to remove private AS numbers from the AS-path in outbound routing updates.

```
Device(config)# neighbor 192.0.2.33 remove-private-as
```

   b) Use the **synchronization** command to enable synchronization between BGP and an IGP.

```
Device(config)# synchronization
```

   c) Use the **auto-summary** command to enable automatic network summarization.

```
Device(config)# auto-summary
```

   When a subnet is redistributed from an IGP into BGP, only the network route is inserted into the BGP table.

d) Use the **bgp graceful-restart** command to enable NSF awareness on the switch.

```
Device(config)# bgp graceful-start
```

By default, NSF awareness is disabled.

7. Use the **end** command to return to privileged EXEC mode.

```
Device(config)# end
```

8. (Optional) Use the **show ip bgp network** *network-number* command to verify the configuration.

```
Device# show ip bgp network 192.0.2.0
```

9. (Optional) Use the **show ip bgp neighbor** command to verify that NSF awareness (Graceful Restart) is enabled on the neighbor.

```
Device# show ip bgp neighbor
```

Verifies that NSF awareness (Graceful Restart) is enabled on the neighbor. If NSF awareness is enabled on the switch and the neighbor, this message appears: *Graceful Restart Capability: advertised and received*

If NSF awareness is enabled on the switch, but not on the neighbor, this message appears: *Graceful Restart Capability: advertised*

Managing routing policy changes

Configure BGP routing policy changes by checking route refresh capability and resetting BGP sessions.

This task enables you to manage BGP routing policy changes by determining if a BGP peer supports route refresh capability and performing necessary BGP session resets.

Use this procedure to apply routing policy changes or troubleshoot BGP neighbor connectivity issues.

To learn if a BGP peer supports the route refresh capability and to reset the BGP session:

1. Use the **show ip bgp neighbors** command to display whether a neighbor supports the route refresh capability.

```
Device# show ip bgp neighbors
```

When supported, this message appears for the router:

Received route refresh capability from peer.

2. Use the **clear ip bgp** {***** | *address* | *peer-group-name*} command to reset the routing table on the specified connection.

```
Device# clear ip bgp *
```

- Enter an asterisk (*) to specify that all connections be reset.
- Enter an IP address to specify the connection to be reset.
- Enter a peer group name to reset the peer group.

3. Use the **clear ip bgp** {***** | *address* | *peer-group-name*} **soft out** command to perform an outbound soft reset, which resets the inbound routing table on the specified connection.

```
Device# clear ip bgp * soft out
```

Use this command if route refresh is supported.

- Enter an asterisk (*) to specify that all connections be reset.
- Enter an IP address to specify the connection to be reset.
- Enter a peer group name to reset the peer group.

4. Use the **show ip bgp** command to verify the reset by checking information about the routing table and about BGP neighbors.

```
Device# show ip bgp
```

5. Use the **show ip bgp neighbors** command to verify the reset by checking information about the routing table and about BGP neighbors.

```
Device# show ip bgp neighbors
```

Configure BGP decision attributes

Configure BGP decision attributes to control path selection behavior and routing decisions in your BGP network.

Configure BGP decision attributes to customize how BGP selects the best path when multiple routes to the same destination exist. This allows you to influence routing decisions based on your network requirements and policies.

BGP decision attributes help control path selection by modifying how BGP evaluates routes. These attributes include AS path length, next-hop processing, weights, MED values, and local preference settings.

1. Use the **router bgp** *autonomous-system* command to enable a BGP routing process, assign it an AS number, and enter router configuration mode.

```
Device(config)# router bgp 4500
```

2. (Optional) Use the **bgp best-path as-path ignore** command to configure the router to ignore AS path length in selecting a route.

```
Device(config-router)# bgp bestpath as-path ignore
```

3. (Optional) Use the **neighbor** {*ip-address* | *peer-group-name*} **next-hop-self** command to disable next-hop processing on BGP updates to a neighbor by entering a specific IP address to be used instead of the next-hop address.

```
Device(config-router)# neighbor 192.0.2.11 next-hop-self
```

4. (Optional) Use the **neighbor** {*ip-address* | *peer-group-name*} **weight** *weight* command to assign a weight to a neighbor connection.

```
Device(config-router)# neighbor 192.0.2.12 weight 50
```

Acceptable values are from 0 to 65535; the largest weight is the preferred route. Routes learned through another BGP peer have a default weight of 0; routes sourced by the local router have a default weight of 32768.

5. (Optional) Use the **default-metric** *number* command to set a MED metric to set preferred paths to external neighbors.

```
Device(config-router)# default-metric 300
```

All routes without a MED are set to this value. The MED can range from 1 to 4294967295. Lower values indicate more desirable routes.

6. (Optional) Customize BGP path selection by modifying Multi-Exit Discriminator (MED) and Local Preference attribute evaluation.

a) Use the **bgp bestpath med missing-as-worst** command to configure the switch to consider a missing MED AS having a value of infinity, making the path without a MED value the least desirable path.

```
Device(config-router)# bgp bestpath med missing-as-worst
```

b) Use the **bgp always-compare med** command to configure the switch to compare MEDs for paths from neighbors in different autonomous systems. T

```
Device(config-router)# bgp always-compare-med
```

By default, MED comparison is only done among paths in the same AS.

c) Use the **bgp bestpath med confed** command to configure the switch to consider the MED in choosing a path from among those advertised by different subautonomous systems within a confederation.

```
Device(config-router)# bgp bestpath med confed
```

d) Use the **bgp deterministic med** command to configure the switch to consider the MED variable when choosing among routes advertised by different peers in the same AS.

```
Device(config-router)# bgp deterministic med
```

e) Use the **bgp default local-preference** *value* command to change the default local preference value.

```
Device(config-router)# bgp default local-preference 200
```

The range is 0 to 4294967295; the default value is 100. The highest local preference value is preferred.

7. (Optional) Use the **maximum-paths** *number* command to configure the number of paths to be added to the IP routing table.

```
Device(config-router)# maximum-paths 8
```

By default, only the best path is entered in the routing table. You can set the number of paths from 1 to 16. Configuring multiple paths enables load balancing. The switch software allows up to 32 equal-cost routes, but the hardware uses a maximum of 16 paths per route.

Use the **end** command to return to privileged EXEC mode.

8. Use the **show ip bgp** command to verify the reset by checking information about the routing table and about BGP neighbors.

```
Device# show ip bgp
```

9. Use the **show ip bgp neighbors** command to verify the reset by checking information about the routing table and about BGP neighbors.

```
Device# show ip bgp neighbors
```

Configure BGP filtering with route maps

Configure BGP filtering using route maps to control next-hop processing for inbound and outbound routes.

Configure BGP route maps to control next-hop processing and filtering of routes in BGP sessions.

Route maps provide flexible route filtering and next-hop manipulation capabilities for BGP. Use route maps to override third-party next hops or to disable next-hop calculation for specific BGP peers.

Follow these steps to configure BGP filtering with route maps:

1. Use the **configure terminal** command to enter global configuration mode.

```
Device# configure terminal
```

2. Use the **route-map** *map-tag* [**permit | deny**] [*sequence-number*] command to create a route map, and enter route-map configuration mode.

```
Device(config)# route-map set-peer-address permit 10
```

3. Use the **set ip next-hop** *ip-address* [*...ip-address*] [**peer-address**] command to set a route map to disable next-hop processing. This step is optional.

```
Device(config)# set ip next-hop 192.0.2.3
```

   - In an inbound route map, set the next hop of matching routes to be the neighbor peering address, overriding third-party next hops.
   - In an outbound route map of a BGP peer, set the next hop to the peering address of the local router, disabling the next-hop calculation.

4. Use the **end** command to return to privileged EXEC mode.

```
Device(config)# end
```

5. Use the **show route-map** [*map-name*] command tto display all configured route maps or just the specified one to verify configuration.

```
Device# show route-map
```

Configure BGP filtering by neighbor

Configure BGP filtering by neighbor to control routing updates using access lists or route maps.

This task configures BGP filtering by neighbor to control which routing updates are sent to or received from specific BGP neighbors using access lists or route maps.

BGP filtering by neighbor allows you to control routing information exchange with specific BGP peers, providing granular control over route advertisement and acceptance.

1. Use the **configure terminal** command to enter global configuration mode.

```
Device# configure terminal
```

2. Use the **router bgp** *autonomous-system* command to enable a BGP routing process, assign it an AS number, and enter router configuration mode.

```
Device(config)# router bgp 109
```

3. (Optional) Use the **neighbor** {*ip-address* | *peer-group name*} **distribute-list** {**access-list-number** | *name*} {**in** | **out**} command to filter BGP routing updates to or from neighbors as specified in an access list.

```
Device(config-router)# neighbor 192.0.2.41 distribute-list 39 in
```

> **Note**
>
> You can also use the **neighbor prefix-list** router configuration command to filter updates, but you cannot use both commands to configure the same BGP peer.

4. (Optional) Use the **neighbor** {*ip-address* | *peer-group name*} **route-map** *map-tag* {**in** | **out**} command to apply a route map to filter an incoming or outgoing route.

```
Device(config-router)# neighbor 192.0.2.24 route-map internal-map in
```

5. Use the **end** command to return to privileged EXEC mode.

```
Device(config)# end
```

6. Use the **show ip bgp neighbors** command to verify the configuration.

```
Device# show ip bgp neighbors
```

Configure BGP filtering by access lists and neighbors

Configure BGP filtering by specifying access list filters on incoming and outgoing updates based on BGP autonomous system paths.

This task configures BGP filtering to control route advertisements by applying access list filters based on autonomous system paths for specific neighbors.

Another method of filtering is to specify an access list filter on both incoming and outbound updates, based on the BGP autonomous system paths. Each filter is an access list based on regular expressions. To use this method, define an autonomous system path access list, and apply it to updates to and from particular neighbors.

1. Use the **configure terminal** command to enter global configuration mode.

```
Device# configure terminal
```

2. Use the **ip as-path access-list** *access-list-number* {**permit** | **deny**} *as-regular-expressions* command to define a BGP-related access list.

```
Device(config)# ip as-path access-list 1 deny _65535_
```

3. Use the **router bgp** *autonomous-system* command to enter BGP router configuration mode.

```
Device(config)# router bgp 110
```

4. Use the **neighbor** {*ip-address* | *peer-group name*} **filter-list** {*access-list-number* | *name*} {**in** | **out** | **weight** *weight*} command to establish a BGP filter based on an access list.

```
Device(config-router)# neighbor 192.0.2.21 filter-list 1 out
```

5. Use the **end** command to return to privileged EXEC mode.

```
Device(config)# end
```

6. Use the **show ip bgp neighbors** [**paths** *regular-expression*] command to verify the configuration.

```
Device# show ip bgp neighbors
```

Configure prefix lists for BGP filtering

Configure prefix lists to control BGP route filtering by specifying permitted or denied network prefixes.

Prefix lists efficiently filter BGP routes. You can specify which network prefixes should be permitted or denied during BGP updates.

You do not need to specify a sequence number when removing a configuration entry. The **show** commands include the sequence numbers in their output.

Before using a prefix list in a command, you must set up the prefix list.

1. Use the **configure terminal** command to enter global configuration mode.

```
Device# configure terminal
```

2. Use the **ip prefix-list** *list-name* [**seq** *seq-value*] **deny** | **permit** *network/len* [**ge** *ge-value*] [**le** *le-value*] command to create a prefix list with an optional sequence number to deny or permit access for matching conditions.

```
Device(config)# ip prefix-list BLUE permit 192.0.2.0/25
```

   At least one **permit** or **deny** clause must be entered.

   - *network/len* is the network number and length (in bits) of the network mask.
   - (Optional) **ge** and **le** values specify the range of the prefix length to be matched. The specified *ge-value* and *le-value* must satisfy this condition: *len* < *ge-value* < *le-value* < 32

3. Use the **ip prefix-list** *list-name* **seq** *seq-value* **deny** | **permit** *network/len* [**ge** *ge-value*] [**le** *le-value*] command to add an entry to a prefix list, and assign a sequence number to the entry. This step is optional.

```
Device(config)# ip prefix-list BLUE seq 10 permit 192.0.2.128/25
```

4. Use the **end** command to return to privileged EXEC mode.

```
Device(config)# end
```

5. Use the **show ip prefix list** [**detail** | **summary**] *name* [*network/len*] [**seq** *seq-num*] [**longer**] [**first-match**] command to verify the configuration by displaying information about a prefix list or prefix list entries.

```
Device# show ip prefix list summary test
```

Configure BGP community filtering

Configure BGP community filtering to control which BGP communities are sent to neighbors and to filter community attributes.

Configure BGP community filtering to control the exchange of community attributes between BGP neighbors and implement community-based routing policies.

By default, no communities attribute is sent to a neighbor. You can specify that the communities attribute be sent to the neighbor at an IP address by using the **neighbor send-community** router configuration command.

Follow these steps to configure BGP community filtering:

1. Use the **configure terminal** command to enter global configuration mode.

```
Device# configure terminal
```

2. Use the **ip community-list** *community-list-number* {**permit** | **deny**} *community-number* command to create a community list, and assign it a number.

```
Device(config)# ip community-list 1 permit 50000:10
```

   - The *community-list-number* is an integer from 1 to 99 that identifies one or more permit or deny groups of communities.
   - The *community-number* is the number configured by a **set community** route-map configuration command.

3. Use the **router bgp** *autonomous-system* command to enter BGP router configuration mode.

```
Device(config)# router bgp 108
```

4. Use the **neighbor** {*ip-address* | *peer-group name*} **send-community** command to specify that the communities attribute be sent to the neighbor at this IP address.

```
Device(config-router)# neighbor 192.0.2.23 send-community
```

5. (Optional) Use the **set comm-list** *list-num* **delete** command to remove communities from the community attribute of an inbound or outbound update. The standard or extended community list specified by a route map determines which communities are removed.

```
Device(config-router)# set comm-list 500 delete
```

6. Use the **exit** command to return to global configuration mode.

```
Device(config-router)# end
```

7. (Optional) Use the **ip bgp-community new-format** command to display and parse BGP communities in the format AA:NN. This step is optional.

```
Device(config)# ip bgp-community new format
```

   A BGP community is displayed in a two-part format that is 2 bytes long. The Cisco default community format is NNAA. In the most recent RFC for BGP, a community takes the form AA:NN. In this format, the first part represents the AS number and the second part is a 2-byte number.

8. Use the **end** command to return to privileged EXEC mode.

```
Device(config)# end
```

9. Use the **show ip bgp community** command to verify the configuration.

```
Device# show ip bgp community
```

Configure BGP neighbors and peer groups

Configure BGP neighbor and peer group settings by applying router configuration commands to specific IP addresses or peer group names.

This task enables administrators to manage BGP peer connectivity and attributes without removing existing configuration information.

To assign configuration options to an individual neighbor, use the neighbor IP address with the router configuration command. To assign options to a peer group, use the peer group name with the command. To disable a BGP peer or peer group without removing all configuration information, use the **neighbor shutdown** router configuration command.

1. Use the **configure terminal** command to enter global configuration mode.

```
Device# configure terminal
```

2. Use the **router bgp** *autonomous-system* command to enter BGP router configuration mode.

```
Device(config)# router bgp 65000
```

3. Configure BGP neighbor and peer group settings:
   a) Use the **neighbor** *peer-group-name* **peer-group** command to create a BGP peer group.

```
Device(config-router)# neighbor internal-peers peer-group
```

   b) Use the **neighbor** *ip-address* **peer-group** *peer-group-name* command to make a BGP neighbor a member of the peer group.

```
Device(config-router)# neighbor 192.0.2.2 peer-group internal-peers
```

   c) Use the **neighbor** { *ip-address* | *peer-group-name* } **remote-as** *number* command to specify a BGP neighbor.

```
Device(config-router)# neighbor internal-peers remote-as 65001
```

   Specifies a BGP neighbor. If a peer group is not configured with a **remote-AS** *number*, use this command to create peer groups containing EBGP neighbors. The range is 1 to 65535.

   d) (Optional) Use the **neighbor** { *ip-address* | *peer-group-name* } **description** *text* command to associate a description with a neighbor.

```
Device(config-router)# neighbor internal-peers description Branch peer
group
```

4. (Optional) Configure BGP session and connection optimization:
   a) Use the **neighbor** { *ip-address* | *peer-group-name* } **update-source** *interface* command to allow internal BGP sessions to use any operational interface for TCP connections.

```
Device(config-router)# neighbor 192.0.2.2 update-source Loopback0
```

   b) Use the **neighbor** { *ip-address* | *peer-group-name* } **ebgp-multihop** command to allow BGP sessions, even when the neighbor is not on a directly connected segment.

```
Device(config-router)# neighbor 192.0.2.2 ebgp-multihop
```

   The multihop session is not established if the only route to the multihop peer's address is the default route.

c) Use the **neighbor** { *ip-address* | *peer-group-name* } **local-as** *number* command to specify an AS number to use as the local AS.

```
Device(config-router)# neighbor 192.0.2.2 local-as 65010
```

The range is 1 to 65535.

d) Use the **neighbor** { *ip-address* | *peer-group-name* } **password** *string* command to set MD5 authentication on a TCP connection to a BGP peer.

```
Device(config-router)# neighbor 192.0.2.2 password bgpSharedKey
```

The same password must be configured on both BGP peers, or the connection between them is not made.

e) Use the **neighbor** { *ip-address* | *peer-group-name* } **timers** *keepalive holdtime* command to set timers for the neighbor or peer group.

```
Device(config-router)# neighbor 192.0.2.2 timers 30 90
```

- The *keepalive* interval is the time within which keepalive messages are sent to peers. The range is 1 to 4294967295 seconds; the default is 60.
- The *holdtime* is the interval after which a peer is declared inactive after not receiving a keepalive message from it. The range is 1 to 4294967295 seconds; the default is 180.

f) Use the **neighbor** { *ip-address* | *peer-group-name* } **version** *value* command to specify the BGP version to use when communicating with a neighbor.

```
Device(config-router)# neighbor 192.0.2.2 version 4
```

5. (Optional) Configure routing policy and filtering.

a) Use the **neighbor** { *ip-address* | *peer-group-name* } **route-map** *map-name* { **in** | **out** } command to apply a route map to incoming or outgoing routes.

```
Device(config-router)# neighbor 192.0.2.2 route-map FILTER-IN in
```

b) Use the **neighbor** { *ip-address* | *peer-group-name* } **distribute-list** { *access-list-number* | *name* } { **in** | **out** } command to filter BGP routing updates to or from neighbors, as specified in an access list.

```
Device(config-router)# neighbor 192.0.2.2 distribute-list 10 in
```

c) Use the **neighbor** { *ip-address* | *peer-group-name* } **filter-list** *access-list-number* { **in** | **out** | **weight** *weight* } command to establish a BGP filter.

```
Device(config-router)# neighbor 192.0.2.2 filter-list 1 out
```

d) Use the **neighbor** { *ip-address* | *peer-group-name* } **maximum-prefix** *maximum* [*threshold*] command to control how many prefixes can be received from a neighbor.

```
Device(config-router)# neighbor 192.0.2.2 maximum-prefix 10000 75
```

(Optional) Controls how many prefixes can be received from a neighbor. The range is 1 to 4294967295. The *threshold* (optional) is the percentage of maximum at which a warning message is generated. The default is 75 percent.

6. (Optional) Configure prefix and attributes.

a) Use the **neighbor** { *ip-address* | *peer-group-name* } **default-originate** [**route-map** *map-name*] command to allow a BGP speaker (the local router) to send the default route to a neighbor for use as a default route.

```
Device(config-router)# neighbor 192.0.2.2 default-originate route-map
DEFAULT-ONLY
```

b) Use the **neighbor** { *ip-address* | *peer-group-name* } **send-community** command to specify that the communities attribute be sent to the neighbor at this IP address.

```
Device(config-router)# neighbor 192.0.2.2 send-community
```

c) Use the **neighbor** { *ip-address* | *peer-group-name* } **next-hop-self** command to disable next-hop processing on the BGP updates to a neighbor.

```
Device(config-router)# neighbor 192.0.2.2 next-hop-self
```

d) Use the **neighbor** { *ip-address* | *peer-group-name* } **weight** *weight* command to specify a weight for all routes from a neighbor.

```
Device(config-router)# neighbor 192.0.2.2 weight 50
```

e) Use the **neighbor** { *ip-address* | *peer-group-name* } **advertisement-interval** *seconds* command to set the minimum interval between sending BGP routing updates.

```
Device(config-router)# neighbor 192.0.2.2 advertisement-interval 30
```

f) Use the **neighbor** { *ip-address* | *peer-group-name* } **soft-reconfiguration inbound** command to configure the software to start storing received updates.

```
Device(config-router)# neighbor 192.0.2.2 soft-reconfiguration inbound
```

7. Use the **end** command to return to privileged EXEC mode.

```
Device(config-router)# end
```

8. Use the **show ip bgp neighbors** command to verify the configuration.

```
Device# show ip bgp neighbors
```

Configure aggregate addresses in a routing table

Configure aggregate addresses in BGP routing tables to summarize route advertisements and control path information.

This task creates aggregate entries in BGP routing tables to summarize multiple routes into single advertisements, reducing routing table size and controlling route propagation.

Use aggregate addresses to consolidate multiple BGP routes into summary advertisements. This configuration helps reduce routing table size and provides better control over route advertisements to BGP neighbors.

Follow these steps to configure aggregate addresses in a routing table:

1. Use the **router bgp** *autonomous-system* command to enter BGP router configuration mode.

```
Device(config)# router bgp 106
```

2. Use the **aggregate-address** *address mask* command to create an aggregate entry in the BGP routing table.

```
Device(config-router)# aggregate-address 192.0.2.0 255.255.255.0
```

he aggregate route is advertised as coming from the AS. The atomic aggregate attribute is set to indicate that some information might be missing.

3. Use the **aggregate-address** *address mask* **as-set** command to generate AS SET path information. This step is optional.

```
Device(config-router)# aggregate-address 192.0.2.0 255.255.255.0 as-set
```

This command creates an aggregate entry that uses the same rules as the previous command. However, the advertised path will use an AS_SET, which consists of all elements found in all paths. Do not use this keyword when aggregating many paths, as this route must be continually withdrawn and updated.

4. Use the **aggregate-address** *address-mask* **summary-only** command to advertise summary addresses only. This step is optional.

```
Device(config-router)# aggregate-address 192.0.2.0 255.255.255.0 summary-only
```

5. Use the **aggregate-address** *address mask* **suppress-map** *map-name* command to suppress selected, more specific routes. This step is optional.

```
Device(config-router)# aggregate-address 192.0.2.0 255.255.255.0 suppress-map
 map1
```

6. Use the **aggregate-address** *address mask* **advertise-map** *map-name* command to generate an aggregate based on conditions specified by the route map. This step is optional.

```
Device(config-router)# aggregate-address 192.0.2.0 255.255.255.0 advertise-map
 map2
```

7. Use the **aggregate-address** *address mask* **attribute-map** *map-name* command to generate an aggregate with attributes specified in the route map. This step is optional.

```
Device(config-router)# aggregate-address 192.0.2.0 255.255.255.0 attribute-map
 map3
```

8. Use the **end** command to return to privileged EXEC mode.

```
Device(config)# end
```

9. Use the **show ip bgp neighbors** [**advertised-routes**] command to verify the configuration.

```
Device# show ip bgp neighbors
```

Configure routing domain confederations

Configure BGP routing domain confederations by specifying a confederation identifier and autonomous system peers.

Configure BGP routing domain confederations to group autonomous systems and manage routing policies across multiple AS domains within a larger confederation structure.

You must specify a confederation identifier that acts as the autonomous system number for the group of autonomous systems.

1. Use the **configure terminal** command to enter global configuration mode.

```
Device# configure terminal
```

2. Use the **router bgp** *autonomous-system* command to enter BGP router configuration mode.

```
Device(config)# router bgp 100
```

3. Use the **bgp confederation identifier** *autonomous-system* command to configure a BGP confederation identifier.

```
Device(config)# bgp confederation identifier 50007
```

4. Use the **bgp confederation peers** *autonomous-system* [*autonomous-system*...] command to specify the autonomous systems that belong to the confederation and that will be treated as special EBGP peers.

```
Device(config)# bgp confederation peers 51000 51001 51002
```

5. Use the **end** command to return to privileged EXEC mode.

```
Device(config)# end
```

6. Use the **show ip bgp neighbor** command to verify the configuration.

```
Device# show ip bgp neighbor
```

7. Use the **show ip bgp network** command to verify the configuration.

```
Device# show ip bgp network
```

Configure BGP route reflectors

Configure BGP route reflectors to improve scalability by reducing the number of required iBGP peering sessions in your network.

Configure BGP route reflectors to reduce the number of iBGP peering sessions in your network. This will improve scalability.

BGP route reflectors allow you to reduce the full mesh requirement for iBGP peers. You can designate specific routers as route reflectors to forward routing information between clients.

1. Use the **configure terminal** command to enter global configuration mode.

```
Device# configure terminal
```

2. Use the **router bgp** *autonomous-system* command to enter BGP router configuration mode.

```
Device(config)# router bgp 101
```

3. Use the **neighbor** {*ip-address* | *peer-group-name*} **route-reflector-client** command to configure the local router as a BGP route reflector and the specified neighbor as a client.

```
Device(config-router)# neighbor 192.0.2.24 route-reflector-client
```

4. (Optional) Use the **bgp cluster-id** *cluster-ID* command to configure the cluster ID if the cluster has more than one route reflector.

```
Device(config-router)# bgp cluster-id 192.0.2.42
```

5. (Optional) Use the **no bgp client-to-client reflection** command to disable client-to-client route reflection.

```
Device(config-router)# no bgp client-to-client reflection
```

By default, the routes from a route reflector client are reflected to other clients. However, if the clients are fully meshed, the route reflector does not need to reflect routes to clients.

6. Use the **end** command to return to privileged EXEC mode.

```
Device(config)# end
```

7. Use the **show ip bgp** command to verify the configuration.

```
Device# show ip bgp
```

Displays the originator ID and the cluster-list attributes.

Configure route dampening

Configure BGP route dampening to reduce the impact of route flapping on network stability.

Route dampening helps minimize the impact of route flapping by suppressing unstable routes until they become stable again.

BGP route dampening is used to reduce the propagation of flapping routes in BGP networks. When a route flaps frequently, it can consume network resources and affect routing stability.

1. Use the **configure terminal** command to enter global configuration mode.

```
Device# configure terminal
```

2. Use the **router bgp** *autonomous-system* command to enter BGP router configuration mode.

```
Device(config)# router bgp 100
```

3. Use the **bgp dampening** command to enable BGP route dampening.

```
Device(config-router)# bgp dampening
```

4. (Optional) Use the **bgp dampening** *half-life reuse suppress max-suppress* [**route-map** *map*] command to modify the default values of route dampening factors.

```
Device(config-router)# bgp dampening 30 1500 10000 120
```

5. Use the **end** command to return to privileged EXEC mode.

```
Device(config)# end
```

6. Use the **show ip bgp flap-statistics** [{**regexp** *regexp*} | {**filter-list** *list*} | {*address mask* [**longer-prefix**]}] command to monitor the flaps of all paths that are flapping. This step is optional.

```
Device# show ip bgp flap-statistics
```

   The statistics are deleted when the route is not suppressed and is stable.

7. Use the **show ip bgp dampened-paths** command to display the dampened routes, including the time remaining before they are suppressed. This step is optional.

```
Device# show pi bgp dampened-paths
```

8. Use the **clear ip bgp flap-statistics** [{**regexp** *regexp*} | {**filter-list** *list*} | {*address mask* [**longer-prefix**]}] command to clear BGP flap statistics to make it less likely that a route will be dampened. This step is optional.

```
Device# clear ip bgp flap-statistics
```

9. Use the **clear ip bgp dampening** command to clear route dampening information, and unsuppress the suppressed routes. This step is optional.

```
Device# clear ip bgp dampening
```

Conditionally inject BGP routes

Configure conditional injection of more specific BGP routes into a routing table to provide finer granularity of traffic engineering or administrative control.

Use this task to inject more specific prefixes into a BGP routing table, replacing less specific prefixes selected through normal route aggregation. These specific prefixes enable more detailed traffic engineering or administrative control than aggregated routes allow.

This task allows you to override standard route aggregation. You can inject more specific routes to achieve greater control over traffic engineering and routing decisions.

This task assumes that the IGP is already configured for the BGP peers.

Follow these steps to conditionally inject BGP routes:

1. Use the **configure terminal** command to enter global configuration mode.

```
Device# configure terminal
```

2. Use the **router bgp** *autonomous-system-number* command to enter router configuration mode for the specified routing process.

```
Device(config)# router bgp 40000
```

3. Use the **bgp inject-map** *inject-map-name* **exist-map** *exist-map-name* [**copy-attributes**] command to specify the inject map and the exist map for conditional route injection.

```
Device(config-router)# bgp inject-map ORIGINATE exist-map LEARNED_PATH
```

   - Use the **copy-attributes** keyword to specify that the injected ROUTE inherit the attributes of the aggregate route.

4. Use the **exit** command to exit router configuration mode and enter global configuration mode.

```
Device(config-router)# exit
```

5. Configure route maps, aggregate route specifications, and matching criteria for route injection and redistribution.
   a) Use the **route-map** *map-tag* [**permit** |**deny**] [*sequence-number*] command to configure a route map and enter route map configuration mode.

```
Device(config)# route-map LEARNED_PATH permit 10
```

   b) Use the **match ip address** {*access-list-number* [*access-list-number...* | *access-list-name...*] | *access-list-name* [*access-list-number...* | *access-list-name*] | **prefix-list** *prefix-list-name* [*prefix-list-name...*]} command to specify the aggregate route to which a more specific route will be injected.

```
Device(config-route-map)# match ip address prefix-list SOURCE
```

   - In this example, the prefix list named source is used to redistribute the source of the route.

   c) Use the **match ip route-source** {*access-list-number* | *access-list-name*} [*access-list-number...*| *access-list-name...*] command to specify the match conditions for redistributing the source of the route.

```
Device(config-route-map)# match ip route-source prefix-list ROUTE_SOURCE
```

   - In this example, the prefix list named ROUTE_SOURCE is used to redistribute the source of the route.

> **Note**
>
> The route source is the neighbor address that is configured with the **neighbor remote-as** command. The tracked prefix must come from this neighbor in order for conditional route injection to occur.

   d) Use the **exit** command to exit route map configuration mode and enter global configuration mode.

```
Device(config-route-map)# exit
```

6. Configure a route map to specify injected routes and define their associated BGP community attributes.
   a) Use the **route-map** *map-tag* [**permit** |**deny**] [*sequence-number*] command to configure a route map and enter route map configuration mode.

```
Device(config)# route-map ORIGINATE permit 10
```

   b) Use the **set ip address** {*access-list-number* [*access-list-number...* | *access-list-name...*] | *access-list-name* [*access-list-number...* | *access-list-name*] | **prefix-list** *prefix-list-name* [*prefix-list-name...*]} command to specify the routes to be injected.

```
Device(config-route-map)# set ip address prefix-list ORIGINATED_ROUTES
```

   - In this example, the prefix list named originated_routes is used to redistribute the source of the route.

   c) Use the **set community** {*community-number* [**additive**] [*well-known-community*] | **none**} command to set the BGP community attribute of the injected route.

```
Device(config-route-map)# set community 14616:555 additive
```

d) Use the **exit** command to exit route map configuration mode and enter global configuration mode.

```
Device(config-route-map)# exit
```

7. Use the **ip prefix-list** *list-name* [**seq** *seq-value*] {**deny** *network/length* | **permit** *network/length*} [**ge** *ge-value*] [**le** *le-value*] command to configure a prefix list.

```
Device(config)# ip prefix-list SOURCE permit 192.0.2.0/24
```

- In this example, the prefix list named source is configured to permit routes from network 192.0.2.0/24.

> **Note**
>
> Repeat this step for every prefix list to be created.

8. Use the **exit** command to exit global configuration mode and return to privileged EXEC mode.

```
Device(config)# exit
```

9. Use the **show ip bgp injected-paths** command to display information about injected paths. This step is optional.

```
Device# show ip bgp injected-paths
```

Configuring peer session templates

Describes how to create and configure peer session templates using specific configuration tasks to streamline BGP neighbor configurations.

Use the following tasks to create and configure a peer session template:

Configure a basic peer session template

Configure a basic peer session template with general BGP routing session commands that can be applied to many neighbors.

This task creates a basic peer session template that contains general BGP routing session commands for application to multiple neighbors, streamlining BGP configuration management.

Perform this task to create a basic peer session template with general BGP routing session commands that can be applied to many neighbors using one of the next two tasks.

> **Note**
>
> The commands in Step 5 and 6 are optional and could be replaced with any supported general session commands.

> **Note**
>
> Restrictions apply to peer session templates:
>
> - A peer session template can directly inherit only one session template, and each inherited session template can also contain one indirectly inherited session template. So, a neighbor or neighbor group can be configured with only one directly applied peer session template and seven additional indirectly inherited peer session templates.
> - A BGP neighbor cannot be configured to work with both peer groups and peer templates. A BGP neighbor can be configured to belong only to a peer group or to inherit policies only from peer templates.

Follow these steps to configure a basic peer session template:

1. Use the **enable** command to enable privileged EXEC mode.

```
Device> enable
```

   Enter your password if prompted.

2. Use the **configure terminal** command to enter global configuration mode.

```
Device# configure terminal
```

3. Use the **router bgp** *autonomous-system-number* command to enter router configuration mode and create a BGP routing process.

```
Device(config)# router bgp 101
```

4. Use the **template peer-session** *session-template-name* command to enter session-template configuration mode and create a peer session template.

```
Device(config-router)# template peer-session INTERNAL-BGP
```

5. Use the **remote-as** *autonomous-system-number* command to configure peering with a remote neighbor in the specified autonomous system. This step is optional.

```
Device(config-router-stmp)# remote-as 202
```

> **Note**
>
> Any supported general session command can be used here. For a list of the supported commands, see the "Restrictions" section.

6. Use the **timers** *keepalive-interval hold-time* command to configure BGP keepalive and hold timers. This step is optional.

```
Device(config-router-stmp)# timers 30 300
```

The hold time must be at least twice the keepalive time.

> **Note**
>
> Any supported general session command can be used here. For a list of the supported commands, see the "Restrictions" section.

7. Use the **end** command to exit session-template configuration mode and return to privileged EXEC mode.

```
Device(config-router)# end
```

8. Use the **show ip bgp template peer-session** [*session-template-name*] command to display locally configured peer session templates.

```
Device# show ip bgp template peer-session
```

The output can be filtered to display a single peer policy template with the *session-template-name* argument. This command also supports all standard output modifiers.

You have successfully configured a basic peer session template that can be applied to multiple BGP neighbors. The template is now available for use in BGP neighbor configurations.

Configure peer session template inheritance with the inherit peer-session command
Configure peer session template inheritance with the inherit peer-session command.

This task creates and configures a peer session template. It also allows the peer session template to inherit a configuration from another peer session template by using the **inherit peer-session** command.

These steps help to configure peer session template inheritance with the inherit peer-session command:

1. Use the **enable** command to enable privileged EXEC mode.

```
Device> enable
```

Enter your password if prompted.

2. Use the **configure terminal** command to enter global configuration mode.

```
Device# configure terminal
```

3. Use the **router bgp** *autonomous-system-number* command to enter router configuration mode and create a BGP routing process.

```
Device(config)# router bgp 101
```

4. Use the **template peer-session** *session-template-name* command to enter session-template configuration mode and create a peer session template.

```
Device(config-router)# template peer-session CORE1
```

5. (Optional) Use the **description** *text-string* command to configure a description.

```
Device(config-router-stmp)# description CORE-123
```

The text string can be up to 80 characters.

> **Note**
>
> Any supported general session command can be used here.

6. (Optional) Use the **update-source** *interface-type interface-number* command to configure a router to select a specific source or interface to receive routing table updates.

```
Device(config-router-stmp)# update-source loopback 1
```

The example uses a loopback interface. The advantage to this configuration is that the loopback interface is not as susceptible to the effects of a flapping interface.

> **Note**
>
> Any supported general session command can be used here. For a list of the supported commands, see the "Restrictions" section.

7. Use the **inherit peer-session** *session-template-name* command to configure this peer session template to inherit the configuration of another peer session template.

```
Device(config-router-stmp)# inherit peer-session INTERNAL-BGP
```

This peer session template inherits the configuration from INTERNAL-BGP. You can apply this template to a neighbor to indirectly apply the INTERNAL-BGP configuration. You cannot directly apply additional peer session templates. However, the directly inherited template can include up to seven indirectly inherited peer session templates.

8. Use the **end** command to exit session-template configuration mode and enter privileged EXEC mode.

```
Device(config-router)# end
```

9. Use the **show ip bgp template peer-session** [*session-template-name*] command to display locally configured peer session templates.

```
Device# show ip bgp template peer-session
```

The output can be filtered to display a single peer policy template with the optional *session-template-name* argument. This command also supports all standard output modifiers.

Configure peer session template inheritance with the neighbor inherit peer-session command

Configure a device to send a peer session template to a neighbor to inherit the configuration from the specified peer session template with the neighbor inherit peer-session command.

Configure a device to send a peer session template to a neighbor to inherit the configuration from the specified peer session template with the **neighbor inherit peer-session** command.

This task helps you configure a device to send a peer session template to a neighbor. The neighbor inherits configuration from the specified peer session template by using the **neighbor inherit peer-session** command.

Follow these steps to configure peer session template inheritance:

1. Use the **configure terminal** command to enter global configuration mode.

```
Device# configure terminal
```

2. Use the **router bgp** *autonomous-system-number* command to enter router configuration mode and create a BGP routing process.

```
Device(config)# router bgp 101
```

3. Use the **neighbor** *ip-address* **remote-as** *autonomous-system-number* command to configure a peering session with the specified neighbor.

```
Device(config-router)# neighbor 192.0.2.31 remote-as 20
```

The explicit **remote-as** statement is required for the neighbor inherit statement to operate correctly. If peering is not configured, the specified neighbor will not accept the session template.

4. Use the **neighbor** *ip-address* **inherit peer-session** *session-template-name* command to send a peer session template to a neighbor so that the neighbor can inherit the configuration.

```
Device(config-router)# neighbor 192.0.2.31 inherit peer-session CORE1
```

The example configures a device to send the peer session template named CORE1 to the 192.0.2.31 neighbor to inherit. This template can be applied to a neighbor, and if another peer session template is indirectly inherited in CORE1, the indirectly inherited configuration will also be applied. No additional peer session templates can be directly applied. However, the directly inherited template can also inherit up to seven additional indirectly inherited peer session templates.

5. Use the **end** command to exit router configuration mode and enter privileged EXEC mode.

```
Device(config-router)# end
```

6. Use the **show ip bgp template peer-session** [*session-template-name*] command to display locally configured peer session templates.

```
Device# show ip bgp template peer-session
```

You can filter the output to display a single peer policy template by using the optional *session-template-name* argument. The command supports all standard output modifiers.

Configuring peer policy templates

Describes how to create and configure peer policy templates using specific configuration tasks for network policy management.

Use the following tasks to create and configure a peer policy template:

Configure basic peer policy templates

Configure basic peer policy templates with BGP policy configuration commands that can be applied to many neighbors.

Create a basic peer policy template with BGP policy configuration commands that can be applied to many neighbors.

Perform this task to create a basic peer policy template with BGP policy configuration commands. You can apply the template to many neighbors by using one of the next two tasks.

The following restrictions apply to the peer policy templates:

- A peer policy template can directly or indirectly inherit up to eight peer policy templates.
- A BGP neighbor cannot be configured to work with both peer groups and peer templates. Each neighbor can belong to a peer group or inherit policies from peer templates, but not both.

1. Use the **configure terminal** command to enter global configuration mode.

```
Device# configure terminal
```

2. Use the **router bgp** *autonomous-system-number* command to enter router configuration mode and create a BGP routing process.

```
Device(config)# router bgp 45000
```

3. Use the **template peer-policy** *policy-template-name* command to enter policy-template configuration mode and create a peer policy template.

```
Device(config-router)# template peer-policy GLOBAL
```

4. (Optional) Use the **maximum-prefix** *prefix-limit* [*threshold*] [**restart** *restart-interval* | **warning-only**] command to configure the maximum number of prefixes that a neighbor accept from this peer.

```
Device(config-router-ptmp)# maximum-prefix 10000
```

> **Note**
>
> Any supported BGP policy configuration command can be used here.

5. (Optional) Use the **weight** *weight-value* command to set the default weight for routes that are sent from this neighbor.

```
Device(config-router-ptmp)# weight 300
```

> **Note**
>
> Any supported BGP policy configuration command can be used here.

6. (Optional) Use the **prefix-list** *prefix-list-name*{**in** | **out**} command to filter prefixes that are received by the router or sent from the router.

```
Device(config-router-ptmp)# prefix-list NO-MARKETING in
```

The prefix list in the example filters inbound internal addresses.

> **Note**
>
> Any supported BGP policy configuration command can be used here.

7. Use the **end** command to exit policy-template configuration mode and return to privileged EXEC mode.

```
Device(config-router-ptmp)# end
```

Configure peer policy template inheritance with the inherit peer-policy command

Configure peer policy template inheritance using the inherit peer-policy command to create a peer policy template that inherits configuration from another template.

Configure peer policy template inheritance to create a peer policy template that inherits configuration from another peer policy template.

This task configures peer policy template inheritance using the **inheritpeer-policy** command. It creates and configures a peer policy template and allows it to inherit configuration from another peer policy template.

Follow these steps to configure peer policy template inheritance:

1. Use the **configure terminal** command to enter global configuration mode.

```
Device# configure terminal
```

2. Use the **router bgp** *autonomous-system-number* command to enter router configuration mode and create a BGP routing process.

```
Device(config)# router bgp 45000
```

3. Use the **template peer-policy** *policy-template-name* command to enter policy-template configuration mode and create a peer policy template.

```
Device(config-router)# template peer-policy NETWORK1
```

4. (Optional) Use the **route-map** *map-name* {**in**| **out**} command to apply the specified route map to inbound or outbound routes.

```
Device(config-router-ptmp)# route-map ROUTE in
```

> **Note**
>
> Any supported BGP policy configuration command can be used here.

5. Use the **inherit peer-policy** *policy-template-name sequence-number* command to configure the peer policy template to inherit the configuration of another peer policy template.

```
Device(config-router-ptmp)# inherit peer-policy GLOBAL 10
```

   - The *sequence-number* argument sets the order in which the peer policy template is evaluated. Like a route map sequence number, the lowest sequence number is evaluated first.
   - In this configuration, the peer policy template inherits the configuration from GLOBAL. Applying the template to a neighbor causes the GLOBAL configuration to be inherited and applied indirectly. Up to six additional peer policy templates can be indirectly inherited from GLOBAL, for a total of eight directly applied and indirectly inherited peer policy templates.
   - This template in the example will be evaluated first if no other templates are configured with a lower sequence number.

6. Use the **end** command to exit policy-template configuration mode and return to privileged EXEC mode.

```
Device(config-router-ptmp)# end
```

7. Use the **show ip bgp template peer-policy** [*policy-template-name*[**detail**]] command to display locally configured peer policy templates.

```
Device# show ip bgp template peer-policy NETWORK1 detail

Template:NETWORK1, index:2.
Local policies:0x1, Inherited polices:0x80840
This template inherits:
  GLOBAL, index:1, seq_no:10, flags:0x1
Locally configured policies:
  route-map ROUTE in
Inherited policies:
  prefix-list NO-MARKETING in
  weight 300
  maximum-prefix 10000
 Template:NETWORK1 <detail>
Locally configured policies:
  route-map ROUTE in
route-map ROUTE, permit, sequence 10
  Match clauses:
    ip address prefix-lists: DEFAULT
ip prefix-list DEFAULT: 1 entries
   seq 5 permit 192.0.2.0/25
  Set clauses:
  Policy routing matches: 0 packets, 0 bytes
Inherited policies:
  prefix-list NO-MARKETING in
ip prefix-list NO-MARKETING: 1 entries
   seq 5 deny 192.0.2.128/25
```

- The output can be filtered to display a single peer policy template with the *policy-template-name* argument. This command also supports all standard output modifiers.
- Use the **detail** keyword to display detailed policy information.

Configure peer policy template inheritance with the neighbor inherit peer-policy command

Configure peer policy template inheritance with the neighbor inherit peer-policy command.

This task configures a device to send a peer policy template to a neighbor to inherit using the **neighbor inherit peer-policy** command.

When BGP neighbors use multiple levels of peer templates, it can be difficult to determine which policies are applied to the neighbor. The **policy** and **detail** keywords of the **show ip BGP neighbors** command display both inherited policies and policies that are configured directly on the specified neighbor.

Follow these steps to send a peer policy template configuration to a neighbor to inherit:

1. Use the **configure terminal** command to enter global configuration mode.

```
Device# configure terminal
```

2. Use the **router bgp** *autonomous-system-number* command to enter router configuration mode and create a BGP routing process.

```
Device(config)# router bgp 45000
```

3. Use the **neighbor** *ip-address* **remote-as** *autonomous-system-number* command to configure a peering session with the specified neighbor.

```
Device(config-router)# neighbor 192.0.2.12 remote-as 40000
```

   The explicit **remote-as** statement is required for the **neighbor inherit** statement to work. If a peering is not configured, the specified neighbor will not accept the session template.

4. Use the **address-family ipv4** [**multicast** | **unicast** | **vrf** *vrf-name*] command to enter address family configuration mode to configure a neighbor to accept address family-specific command configurations.

```
Device(config-router)# address-family ipv4 unicast
```

5. Use the **neighbor** *ip-address* **inherit peer-policy** *policy-template-name* command to send a peer policy template to a neighbor so that the neighbor can inherit the configuration.

```
Device(config-router-af)# neighbor 192.0.2.12 inherit peer-policy GLOBAL
```

   The example configures a router to send the peer policy template that is named GLOBAL to the 192.0.2.12 neighbor to inherit. This template can be applied to a neighbor. If another peer policy template is indirectly inherited from GLOBAL, the configuration from that template will also be applied. Up to seven additional peer policy templates can be indirectly inherited from GLOBAL.

6. Use the **end** command to exit address family configuration mode and return to privileged EXEC mode.

```
Device(config-router-af)# end
```

7. Use the **show ip bgp neighbors**[*ip-address*[**policy** [**detail**]]] command to display locally configured peer policy templates.

```
Device# show ip bgp neighbors 192.0.2.12 policy
Neighbor: 192.0.2.12, Address-Family: IPv4 Unicast
Locally configured policies:
 route-map ROUTE in
Inherited polices:
 prefix-list NO-MARKETING in
 route-map ROUTE in
 weight 300
 maximum-prefix 10000
```

   - The output can be filtered to display a single peer policy template with the *policy-template-name* argument. This command also supports all standard output modifiers.
   - Use the **policy** keyword to display the policies that are applied to this neighbor per address family.
   - Use the **detail** keyword to display detailed policy information.

Configuration Examples for BGP

Example: configure conditional BGP route injection

Provides sample output for the show ip BGP injected-paths command used in conditional BGP route injection configuration.

This example shows the sample output displayed when the **showipbgpinjected-paths** command is entered for conditional BGP route injection.

```
Device# show ip bgp injected-paths

BGP table version is 11, local router ID is 192.0.2.1
Status codes:s suppressed, d damped, h history, * valid, > best, i -
internal
Origin codes:i - IGP, e - EGP, ? - incomplete
   Network          Next Hop            Metric LocPrf Weight Path
*> 192.0.2.0/25       192.0.2.2                              0 ?
*> 192.0.2.128/25    192.0.2.2                             0 ?
```

Example: configure peer session templates

Provides examples of configuring peer session templates in BGP router configuration.

This topic demonstrates how to create and configure peer session templates for BGP routing, including template inheritance and neighbor configuration.

This example creates a peer session template named INTERNAL-BGP in session-template configuration mode:

```
router bgp 45000
 template peer-session INTERNAL-BGP
 remote-as 50000
 timers 30 300
 exit-peer-session
```

This example creates a peer session template named CORE1. This example inherits the configuration of the peer session template named INTERNAL-BGP.

```
router bgp 45000
 template peer-session CORE1
 description CORE-123
 update-source loopback 1
 inherit peer-session INTERNAL-BGP
 exit-peer-session
```

This example configures the 192.0.2.32 neighbor to inherit the CORE1 peer session template. The 192.0.2.32 neighbor will also indirectly inherit the configuration from the peer session template named INTERNAL-BGP. The explicit **remote-as** statement is required for the neighbor inherit statement to function. If peering is not configured, the neighbor does not accept the session template.

```
router bgp 45000
 neighbor 192.0.2.32 remote-as 50000
 neighbor 192.0.2.32 inherit peer-session CORE1
```

Configure peer policy templates

Provides examples for configuring peer policy templates in BGP routing.

This topic explains how to configure peer policy templates for BGP routing. It covers creating templates with different configurations and inheritance patterns.

This example creates a peer policy template named GLOBAL and enters policy-template configuration mode:

```
router bgp 45000
 template peer-policy GLOBAL
  weight 1000
  maximum-prefix 5000
  prefix-list NO_SALES in
  exit-peer-policy
```

This example creates a peer policy template named PRIMARY-IN and enters policy-template configuration mode:

```
router bgp 45000
 template peer-policy PRIMARY-IN
  prefix-list ALLOW-PRIMARY-A in
  route-map SET-LOCAL in
  weight 2345
  default-originate
  exit-peer-policy
```

This example creates a peer policy template named CUSTOMER-A. It is configured to inherit the configuration from the peer policy templates named PRIMARY-IN and GLOBAL.

```
router bgp 45000
 template peer-policy CUSTOMER-A
  route-map SET-COMMUNITY in
  filter-list 20 in
  inherit peer-policy PRIMARY-IN 20
  inherit peer-policy GLOBAL 10
  exit-peer-policy
```

This example configures the 192.0.2.22 neighbor in address family mode to inherit the peer policy template named CUSTOMER-A. Because the peer policy template named CUSTOMER-A inherited the configuration from the templates named PRIMARY-IN and GLOBAL, the 192.0.2.22 neighbor will also indirectly inherit the peer policy templates named PRIMARY-IN and GLOBAL.

```
router bgp 45000
 neighbor 192.0.2.22 remote-as 50000
 address-family ipv4 unicast
  neighbor 192.0.2.22 inherit peer-policy CUSTOMER-A
  end
```

Monitor and maintain BGP

Lists privileged EXEC commands for clearing and displaying BGP routing tables, caches, and databases.

You can remove all contents of a particular cache, table, or database. This might be necessary when the contents of the particular structure have become or are suspected to be invalid.

You can display specific statistics, such as the contents of BGP routing tables, caches, and databases. This information can help you understand resource utilization and resolve network problems. You can also display information about node reachability and the routing path for your device's packets.

Table 8: IP BGP clear and show commands

Command	Description
clear IP BGP *address*	Resets a particular BGP connection.
clear IP BGP *	Resets all BGP connections.
clear IP BGP peer-group *tag*	Removes all members of a BGP peer group.
show IP BGP *prefix*	Displays peer groups and peers not in peer groups to which the prefix has been advertised. Also displays prefix attributes such AS the next hop and the local prefix.
show IP BGP cidr-only	Displays all BGP routes that contain subnet and supernet network masks.
show IP BGP community [*community-number*] [**exact**]	Displays routes that belong to the specified communities.
show IP BGP community-list *community-list-number* [**exact-match**]	Displays routes that are permitted by the community list.
show IP BGP filter-list *access-list-number*	Displays routes that are matched by the specified AS path access list.
show IP BGP inconsistent-AS	Displays the routes with inconsistent originating autonomous systems.
show IP BGP regexp *regular-expression*	Displays the routes that have an AS path that matches the specified regular expression entered on the command line.
show IP BGP	Displays the contents of the BGP routing table.
show IP BGP neighbors [*address*]	Displays detailed information on the BGP and TCP connections to individual neighbors.

Command	Description
show IP BGP neighbors [*address*] [**advertised-routes** | **dampened-routes** | **flap-statistics** | **paths** *regular-expression* | **received-routes** | **routes**]	Displays routes learned from a particular BGP neighbor.
show IP BGP paths	Displays all BGP paths in the database.
show IP BGP peer-group [*tag*] [**summary**]	Displays information about BGP peer groups.
show IP BGP summary	Displays the status of all BGP connections.

The **BGP log-neighbor changes** command is enabled by default. It allows the system to log messages generated when a BGP neighbor resets, comes up, or goes down.