IP Routing Configuration Guide, Cisco IOS XE Amsterdam 17.2.x (Catalyst 9300 Switches)

First Published: 2020-03-30

Americas Headquarters
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
USA
http://www.cisco.com
Tel: 408 526-4000
800 553-NETS (6387)
Fax: 408 527-0883

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.

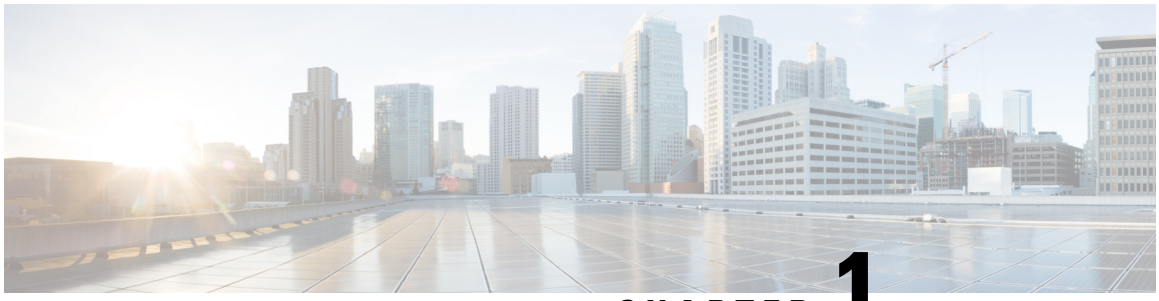
Any Internet Protocol (IP) addresses and phone numbers used in this document are not intended to be actual addresses and phone numbers. Any examples, command display output, network topology diagrams, and other figures included in the document are shown for illustrative purposes only. Any use of actual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

© 2020 Cisco Systems, Inc. All rights reserved.



CHAPTER 1

Configuring Bidirectional Forwarding Detection

This document describes how to enable the Bidirectional Forwarding Detection (BFD) protocol. BFD is a detection protocol that is designed to provide fast forwarding path failure detection times for all media types, encapsulations, topologies, and routing protocols.

BFD provides a consistent failure detection method for network administrators, in addition to fast forwarding path failure detection. Because the network administrator can use BFD to detect forwarding path failures at a uniform rate, rather than the variable rates for different routing protocol hello mechanisms, network profiling and planning will be easier, and reconvergence time will be consistent and predictable.

- Prerequisites for Bidirectional Forwarding Detection, on page 1
- Restrictions for Bidirectional Forwarding Detection, on page 1
- Information About Bidirectional Forwarding Detection, on page 2
- How to Configure Bidirectional Forwarding Detection, on page 5
- Feature History for Configuring Bidirectional Forwarding Detection , on page 22

Prerequisites for Bidirectional Forwarding Detection

- 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 Bidirectional Forwarding Detection

- BFD works only for directly connected neighbors. BFD neighbors must be no more than one IP hop away. BFD does not support Multihop configurations.
- 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.
- The QoS policy for self-generated packets does not match BFD packets.

- The **class class-default** command matches BFD packets. So, you must make sure of the availability of appropriate bandwidth to prevent dropping of BFD packets due to oversubscription.
- BFD HA is not supported.

Information About Bidirectional Forwarding Detection

The following sections provide information about bidirectional forwarding detection.

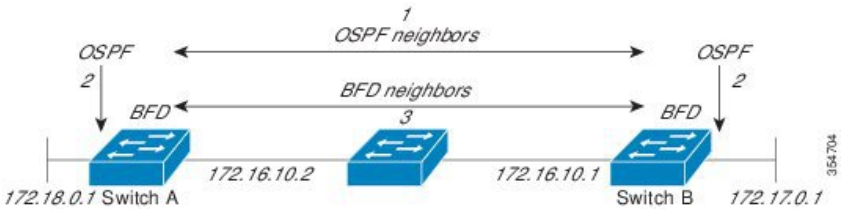
BFD Operation

BFD provides a low-overhead, short-duration method of detecting failures in the forwarding path between two adjacent devices. These devices include the interfaces, data links, and forwarding planes.

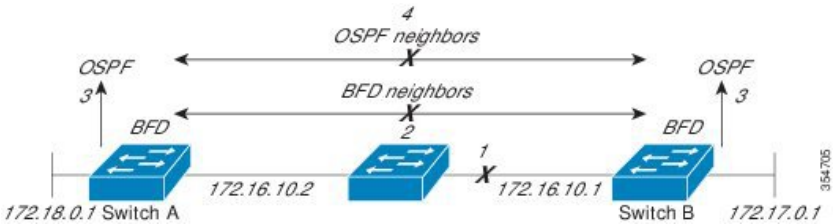
BFD is a detection protocol that you enable at the interface and routing protocol levels. Cisco supports BFD asynchronous mode. BFD asychronous mode depends on the sending of BFD control packets between two systems to activate and maintain BFD neighbor sessions between devices. Therefore, in order to 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

BFD provides fast BFD peer failure detection times independently. This is independent of all media types, encapsulations, topologies, and routing protocols such as BGP, EIGRP, IS-IS, and OSPF. BFD sends rapid failure detection notices to the routing protocols in the local router to initiate the routing table recalculation process. In this way, BFD contributes to greatly reduced overall network convergence time. The figure below shows a simple network with two routers 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 router (2). The BFD neighbor session with the OSPF neighbor router is established (3).



The figure below shows what happens when a failure occurs in the network (1). The BFD neighbor session with the OSPF neighbor router 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 routers immediately start converging on it.



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 routers in a DROTHER state.

BFD Detection of Failures

Once a BFD session is established and timer negations are complete, BFD peers send BFD control packets. The packets act in the same manner as an IGP hello protocol to detect liveliness, except at a more accelerated rate. The following information should be noted:

- 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

All BFD sessions come up as Version 1 by default and are interoperable with Version 0. The system automatically performs BFD version detection, and BFD sessions between neighbors run in the highest common BFD version between neighbors. 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.

See the "Example Configuring BFD in an EIGRP Network with Echo Mode Enabled by Default" for an example of BFD version detection.

BFD Session Limits

Starting with Cisco IOS XE Denali 16.3.1, the number of BFD sessions that can be created has been increased to 100.

BFD Support for Nonbroadcast Media Interfaces

Starting from Cisco IOS XE Denali 16.3.1, the BFD feature is supported on routed, SVI, and L3 port channels. The **bfd interval** command must be configured on the interface to initiate BFD monitoring.

BFD Support for Nonstop Forwarding with Stateful Switchover

Typically, when a networking device restarts, all routing peers of that device detect that the device went down and then came back up. This transition results in a routing flap, which could spread across multiple routing domains. Routing flaps that are caused by routing restarts create routing instabilities, which are detrimental to the overall network performance. Nonstop forwarding (NSF) helps to suppress routing flaps in devices enabled with stateful switchover (SSO), thus reducing network instability.

NSF allows for the forwarding of data packets to continue along known routes while the routing protocol information is restored after a switchover. With NSF, peer networking devices do not experience routing flaps. Data traffic is forwarded through intelligent line cards or dual forwarding processors while the standby RP assumes control from the failed active RP during a switchover. One key to NSF operation is the ability of line cards and forwarding processors to remain up through a switchover. They remain current with the Forwarding Information Base (FIB) on the active RP.

In devices that support dual RPs, SSO establishes one of the RPs as the active processor; the other RP is designated as the standby processor. SSO synchronizes information between the active and standby processor. A switchover from the active to the standby processor occurs when the active RP fails, it is removed from the networking device, or it is manually taken down for maintenance.

BFD Support for Stateful Switchover

The BFD protocol provides short-duration detection of failures in the path between adjacent forwarding engines. In network deployments that use dual RP routers or switches (to provide redundancy), the routers have a graceful restart mechanism. This mechanism protects the forwarding state during a switchover between the active RP and the standby RP.

The dual RPs have variable switchover times that depend on the ability of the hardware to detect a communication failure. When BFD is running on the RP, some platforms are not able to detect a switchover before the BFD protocol times out. These platforms are referred to as slow switchover platforms.

BFD Support for Static Routing

Unlike dynamic routing protocols, such as OSPF and BGP, static routing has no method of peer discovery. Therefore, when BFD is configured, the reachability of the gateway depends on the state of the BFD session to the specified neighbor. Unless the BFD session is up, the gateway for the static route is unreachable, and the affected routes are not installed in the appropriate Routing Information Base (RIB).

To successfully establish a BFD session, BFD must be configured on the interface on the peer. There must be a BFD client that is registered on the peer for the address of the BFD neighbor. When an interface is used by dynamic routing protocols, the latter requirement is met by configuring the routing protocol instances on each neighbor for BFD. When an interface is used exclusively for static routing, this requirement must be met by configuring static routes on the peers.

If a BFD configuration is removed from the remote peer while the BFD session is in the up state, the updated state of the BFD session is not signaled to IPv4 static. This causes the static route to remain in the RIB. The only workaround is to remove the IPv4 static BFD neighbor configuration so that the static route no longer tracks BFD session state. Also, if you change the encapsulation type on a serial interface to one that is unsupported by BFD, BFD will be in a down state on that interface. The workaround is to shut down the interface, change to a supported encapsulation type, and then reconfigure BFD.

A single BFD session can be used by an IPv4 static client to track the reachability of next hops through a specific interface. You can assign a BFD group for a set of BFD-tracked static routes. Each group must have one active static BFD configuration, one or more passive BFD configurations, and the corresponding BFD tracked static routes. Nongroup entries are BFD-tracked static routes for which a BFD group is not assigned. A BFD group must accommodate static BFD configurations that can be part of different VRFs. Effectively, the passive static BFD configurations need not be in the same VRF as that of the active configuration.

For each BFD group, there can be only one active static BFD session. You can configure the active BFD session by adding a static BFD configuration and a corresponding static route that uses the BFD configuration. The BFD session in a group is created only when there is an active static BFD configuration and the static route that uses the static BFD configuration. When the active static BFD configuration or the active static route is removed from a BFD group, all the passive static routes are withdrawn from the RIB. Effectively, all

the passive static routes are inactive until an active static BFD configuration and a static route to be tracked by the active BFD session are configured in the group.

Similarly, for each BFD group, there can be one or more passive static BFD configurations and their corresponding static routes to be BFD-tracked. Passive static session routes take effect only when the active BFD session state is reachable. Though the active BFD session state of the group is reachable, the passive static route is added to the RIB only if the corresponding interface state is up. When a passive BFD session is removed from a group, it will not affect the active BFD session if one existed, or the BFD group reachability status.

Benefits of Using BFD for Failure Detection

When you deploy any feature, it is important to consider all the alternatives and be aware of any trade-offs.

The closest alternative to BFD, in conventional deployments, is the use of modified failure detection mechanisms for EIGRP, IS-IS, and OSPF routing protocols.

If you set EIGRP hello and hold timers to their absolute minimums, the failure detection rate for EIGRP falls to within a one- to two-second range. If you use fast hellos for Interior Gateway Protocol (IGP) protocols such as IS-IS or OSPF, they reduce their failure detection mechanisms to a minimum of one second.

There are several advantages to implementing BFD over reduced timer mechanisms for routing protocols:

- Although reducing the EIGRP, IS-IS, and OSPF timers can result in minimum detection timer of one to two seconds, BFD can provide failure detection in less than one second.
- Because BFD is not tied to any particular routing protocol, it can be used as a generic and consistent failure detection mechanism for EIGRP, IS-IS, and OSPF.
- Because some parts of BFD can be distributed to the data plane, it can be less CPU-intensive than the reduced EIGRP, IS-IS, and OSPF timers, which exist wholly at the control plane.

How to Configure Bidirectional Forwarding Detection

The following sections provide configurational information about bidirectional forwarding detection.

Configuring BFD Session Parameters on the Interface

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.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device>enable`	Enables privileged EXEC mode. Enter your password if prompted.

	Command or Action	Purpose
Step 2	**configure terminal** **Example:** `Device#configure terminal`	Enters global configuration mode.

Step 3	Perform one of the following steps:
	• **ip address** *ipv4-address mask*
	• **ipv6 address** *ipv6-address/mask*
	Example:
	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`
		Configures an IP address for the interface.

Step 4	**bfd interval** *milliseconds* **min_rx** *milliseconds* **multiplier** *interval-multiplier* **Example:** `Device(config-if)#bfd interval 100 min_rx 100 multiplier 3`	Enables BFD on the interface. 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.
Step 5	**end** **Example:** `Device(config-if)#end`	Exits interface configuration mode and returns to privileged EXEC mode.

Configuring BFD Support for Dynamic Routing Protocols

The following sections provide configurational information about BFD support for dynamic routing protocols.

Configuring BFD Support for eBGP

This section describes the procedure for configuring BFD support for BGP. This ensures that BGP is a registered protocol with BFD and receives forwarding path detection failure messages from BFD.

Before you begin

eBGP must be running on all participating routers.

Configure the baseline parameters for BFD sessions on the interfaces over which you want to run BFD sessions to BFD neighbors. See the Configuring BFD Session Parameters on the Interface section for more information.

Note Output from the **show bfd neighbors details** command shows the configured intervals.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device>enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device#configure terminal`	Enters global configuration mode.
Step 3	**router bgp** *as-tag* **Example:** `Device(config)#router bgp tag1`	Specifies a BGP process and enters router configuration mode.
Step 4	**neighbor** *ip-address* **fall-over bfd** **Example:** `Device(config-router)#neighbor 172.16.10.2 fall-over bfd`	Enables BFD support for fallover.
Step 5	**end** **Example:** `Device(config-router)#end`	Exits router configuration mode and returns the router to privileged EXEC mode.
Step 6	**show bfd neighbors** [**details**] **Example:** `Device#show bfd neighbors detail`	(Optional) Verifies that the BFD neighbor is active and displays the routing protocols that BFD has registered.

	Command or Action	Purpose
Step 7	**show ip bgp neighbor** **Example:** `Device#show ip bgp neighbor`	(Optional) Displays information about BGP and TCP connections to neighbors.

Configuring BFD Support for EIGRP

This section describes the procedure for configuring BFD support for EIGRP. This ensures EIGRP is a registered protocol with BFD and receives forwarding path detection failure messages from BFD. There are two methods for enabling BFD support for EIGRP:

- You can enable BFD for all interfaces for which EIGRP is routing by using the **bfd all-interfaces** command in router configuration mode.
- You can enable BFD for a subset of the interfaces for which EIGRP is routing by using the **bfd interface** *type number* command in router configuration mode.

Before you begin

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

Note Output from the **show bfd neighbors details** command shows the configured intervals.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device>enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device#configure terminal`	Enters global configuration mode.
Step 3	**router eigrp** *as-number* **Example:** `Device(config)#router eigrp 123`	Configures the EIGRP routing process and enters router configuration mode.

	Command or Action	Purpose

Step 4	Do one of the following:
	• **bfd all-interfaces**
	• **bfd interface** *type number*
	Example:
	`Device(config-router)#bfd all-interfaces`
	Example:
	`Device(config-router)#bfd interface GigabitFastEthernet 1/0/1`
		Enables BFD globally on all interfaces that are associated with the EIGRP routing process.
		Or
		Enables BFD on a per-interface basis for one or more interfaces that are associated with the EIGRP routing process.

Step 5	**end** **Example:** `Device(config-router)#end`	Exits router configuration mode and returns the router to privileged EXEC mode.
Step 6	**show bfd neighbors** [**details**] **Example:** `Device#show bfd neighbors details`	(Optional) Verifies that the BFD neighbor is active and displays the routing protocols that BFD has registered.
Step 7	**show ip eigrp interfaces** [*type number*] [*as-number*] [**detail**] **Example:** `Device#show ip eigrp interfaces detail`	(Optional) Displays the interfaces for which BFD support for EIGRP is enabled.

Configuring BFD Support for IS-IS

This section describes the procedures for configuring BFD support for IS-IS so that IS-IS is a registered protocol with BFD and will receive forwarding path detection failure messages from BFD. There are two methods for enabling BFD support for IS-IS:

- You can enable BFD for all of the interfaces on which IS-IS is supporting IPv4 routing by using the **bfd all-interfaces** command in router configuration mode. You can then disable BFD for one or more of those interfaces using the **isis bfd disable** command in interface configuration mode.
- You can enable BFD for a subset of the interfaces for which IS-IS is routing by using the **isis bfd** command in interface configuration mode.

To configure BFD support for IS-IS, perform the steps in one of the following sections:

Prerequisites

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

Note Output from the **show bfd neighbors details** command shows the configured intervals. The output does not show intervals that were changed because hardware-offloaded BFD sessions were configured with Tx and Rx intervals that are not multiples of 50 ms.

Configuring BFD Support for IS-IS for All Interfaces

To configure BFD on all IS-IS interfaces that support IPv4 routing, perform the steps in this section.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **router isis** *area-tag*
4. **bfd all-interfaces**
5. **exit**
6. **interface** *type number*
7. **ip router isis** [*tag*]
8. **isis bfd** [**disable**]
9. **end**
10. **show bfd neighbors** [**details**]
11. **show clns interface**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device>enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device#configure terminal`	Enters global configuration mode.
Step 3	**router isis** *area-tag* **Example:** `Device(config)#router isis tag1`	Specifies an IS-IS process and enters router configuration mode.
Step 4	**bfd all-interfaces** **Example:** `Device(config-router)#bfd all-interfaces`	Enables BFD globally on all interfaces that are associated with the IS-IS routing process.

	Command or Action	Purpose
Step 5	**exit** **Example:** `Device(config-router)#exit`	(Optional) Returns the device to global configuration mode.
Step 6	**interface** *type number* **Example:** `Device(config)#interface fastethernet 6/0`	(Optional) Enters interface configuration mode.
Step 7	**ip router isis** [*tag*] **Example:** `Device(config-if)#ip router isis tag1`	(Optional) Enables support for IPv4 routing on the interface.
Step 8	**isis bfd** [**disable**] **Example:** `Device(config-if)#isis bfd`	(Optional) Enables or disables BFD on a per-interface basis for one or more interfaces that are associated with the IS-IS routing process. **Note** You should use the **disable** keyword only if you had earlier enabled BFD on all the interfaces that IS-IS is associated with, using the **bfd all-interfaces** command in configuration mode.
Step 9	**end** **Example:** `Device(config-if)#end`	Exits interface configuration mode and returns the device to privileged EXEC mode.
Step 10	**show bfd neighbors** [**details**] **Example:** `Device#show bfd neighbors details`	(Optional) Displays information that can be used to verify if the BFD neighbor is active and displays the routing protocols that BFD has registered.
Step 11	**show clns interface** **Example:** `Device#show clns interface`	(Optional) Displays information that can be used to verify if BFD for IS-IS has been enabled for a specific IS-IS interface that is associated.

Configuring BFD Support for IS-IS for One or More Interfaces

To configure BFD for only one or more IS-IS interfaces, perform the steps in this section.

SUMMARY STEPS

1. **enable**
2. **configure terminal**

3. **interface** *type number*
4. **ip router isis** [*tag*]
5. **isis bfd [disable]**
6. **end**
7. **show bfd neighbors** [**details**]
8. **show clns interface**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device>enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device#configure terminal`	Enters global configuration mode.
Step 3	**interface** *type number* **Example:** `Device(config)#interface fastethernet 6/0`	Enters interface configuration mode.
Step 4	**ip router isis** [*tag*] **Example:** `Device(config-if)#ip router isis tag1`	Enables support for IPv4 routing on the interface.
Step 5	**isis bfd [disable]** **Example:** `Device(config-if)#isis bfd`	Enables or disables BFD on a per-interface basis for one or more interfaces that are associated with the IS-IS routing process. **Note** You should use the **disable** keyword only if you enabled BFD on all the interfaces that IS-IS is associated with using the **bfd all-interfaces** command in router configuration mode.
Step 6	**end** **Example:** `Device(config-if)#end`	Exits interface configuration mode and returns the device to privileged EXEC mode.

	Command or Action	Purpose
Step 7	**show bfd neighbors** [**details**] **Example:** `Device#show bfd neighbors details`	(Optional) Displays information that can help verify if the BFD neighbor is active and displays the routing protocols that BFD has registered.
Step 8	**show clns interface** **Example:** `Device#show clns interface`	(Optional) Displays information that can help verify if BFD for IS-IS has been enabled for a specific IS-IS interface that is associated.

Configuring BFD Support for OSPF

This section 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. You can either configure BFD support for OSPF globally on all interfaces or configure it selectively on one or more interfaces.

There are two methods for enabling BFD support for OSPF:

- You can enable BFD for all the interfaces for which OSPF is routing by using the **bfd all-interfaces** command in router 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.

See the following sections for tasks for configuring BFD support for OSPF:

Configuring BFD Support for OSPF for All Interfaces

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.

Before you begin

- OSPF must be running on all participating devices.
- 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.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device>enable`	Enables privileged EXEC mode. Enter your password if prompted.

	Command or Action	Purpose
Step 2	**configure terminal** **Example:** `Device#configure terminal`	Enters global configuration mode.
Step 3	**router ospf** *process-id* **Example:** `Device(config)#router ospf 4`	Specifies an OSPF process and enters router configuration mode.
Step 4	**bfd all-interfaces** **Example:** `Device(config-router)#bfd all-interfaces`	Enables BFD globally on all interfaces that are associated with the OSPF routing process.
Step 5	**exit** **Example:** `Device(config-router)#exit`	(Optional) Returns the device to global configuration mode. Enter this command only if you want to perform Step 7 to disable BFD for one or more interfaces.
Step 6	**interface** *type number* **Example:** `Device(config)#interface fastethernet 6/0`	(Optional) Enters interface configuration mode. Enter this command only if you want to perform Step 7 to disable BFD for one or more interfaces.
Step 7	**ip ospf bfd** [**disable**] **Example:** `Device(config-if)#ip ospf bfd disable`	(Optional) Disables BFD on a per-interface basis for one or more interfaces that are associated with the OSPF routing process. **Note** You should use the **disable** keyword only if you enabled BFD on all the interfaces that OSPF is associated with using the **bfd all-interfaces** command in router configuration mode.
Step 8	**end** **Example:** `Device(config-if)#end`	Exits interface configuration mode and returns the router to privileged EXEC mode.
Step 9	**show bfd neighbors** [**details**] **Example:** `Device#show bfd neighbors detail`	(Optional) Displays information that can help verify if the BFD neighbor is active and displays the routing protocols that BFD has registered.
Step 10	**show ip ospf** **Example:**	(Optional) Displays information that can help verify if BFD for OSPF has been enabled.

	Command or Action	Purpose
	`Device#show ip ospf`	

Configuring OSPF Support for BFD over IPv4 for One or More Interfaces

To configure BFD on one or more OSPF interfaces, perform the steps in this section.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface** *type number*
4. **ip ospf bfd** [**disable**]
5. **end**
6. **show bfd neighbors** [**details**]
7. **show ip ospf**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device>enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device#configure terminal`	Enters global configuration mode.
Step 3	**interface** *type number* **Example:** `Device(config)#interface fastethernet 6/0`	Enters interface configuration mode.
Step 4	**ip ospf bfd** [**disable**] **Example:** `Device(config-if)#ip ospf bfd`	Enables or disables BFD on a per-interface basis for one or more interfaces that are associated with the OSPF routing process. **Note** Use the **disable** keyword only if you enable BFD on all the interfaces that OSPF is associated with using the **bfd all-interfaces** command in router configuration mode.
Step 5	**end** **Example:** `Device(config-if)#end`	Exits interface configuration mode and returns the device to privileged EXEC mode.

	Command or Action	Purpose
Step 6	**show bfd neighbors** [**details**] **Example:** `Device#show bfd neighbors details`	(Optional) Displays information that can help verify if the BFD neighbor is active and displays the routing protocols that BFD has registered. **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.
Step 7	**show ip ospf** **Example:** `Device#show ip ospf`	(Optional) Displays information that can help verify if BFD support for OSPF has been enabled.

Configuring BFD Support for HSRP

Perform this task to enable BFD support for Hot Standby Router Protocol (HSRP.) Repeat the steps in this procedure for each interface over which you want to run BFD sessions to HSRP peers.

HSRP supports BFD by default. If HSRP support for BFD has been manually disabled, you can reenable it at the device level to enable BFD support globally for all interfaces or on a per-interface basis at the interface level.

Before you begin

- HSRP must be running on all participating devices.
- Cisco Express Forwarding must be enabled.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device>enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device#configure terminal`	Enters global configuration mode.
Step 3	**ip cef** [**distributed**] **Example:**	Enables Cisco Express Forwarding or distributed Cisco Express Forwarding.

	Command or Action	Purpose
	`Device(config)#ip cef`	
Step 4	**interface** *type number* **Example:** `Device(config)#interface FastEthernet 6/0`	Enters interface configuration mode.
Step 5	**ip address** *ip-address mask* **Example:** `Device(config-if)#ip address 10.1.0.22 255.255.0.0`	Configures an IP address for the interface.
Step 6	**standby** [*group-number*] **ip** [*ip-address* [**secondary**]] **Example:** `Device(config-if)#standby 1 ip 10.0.0.11`	Activates HSRP.
Step 7	**standby bfd** **Example:** `Device(config-if)#standby bfd`	(Optional) Enables HSRP support for BFD on the interface.
Step 8	**exit** **Example:** `Device(config-if)#exit`	Exits interface configuration mode.
Step 9	**standby bfd all-interfaces** **Example:** `Device(config)#standby bfd all-interfaces`	(Optional) Enables HSRP support for BFD on all interfaces.
Step 10	**exit** **Example:** `Device(config)#exit`	Exits global configuration mode.
Step 11	**show standby neighbors** **Example:** `Device#show standby neighbors`	(Optional) Displays information about HSRP support for BFD.

Configuring BFD Support for Static Routing

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.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device>enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device#configure terminal`	Enters global configuration mode.
Step 3	**interface** *type number* **Example:** `Device(config)#interface serial 2/0`	Configures an interface and enters interface configuration mode.

Step 4	Perform one of the following steps:
	• **ip address** *ipv4-address mask*
	• **ipv6 address** *ipv6-address/mask*
	Example:
	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`
		Configures an IP address for the interface.

Step 5	**bfd interval** *milliseconds* **mix_rx** *milliseconds* **multiplier** *interval-multiplier* **Example:** `Device(config-if)#bfd interval 500 min_rx 500 multiplier 5`	Enables BFD on the interface. 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 IPv4 address is removed from an interface • an IPv6 address is removed from an interface • IPv6 is disabled from an interface. • an interface is shutdown • IPv4 CEF is disabled globally or locally on an interface. • IPv6 CEF is disabled globally or locally on an interface.

	Command or Action	Purpose
Step 6	**exit** **Example:** `Device(config-if)#exit`	Exits interface configuration mode and returns to global configuration mode.
Step 7	**ip route static bfd** *interface-type interface-number ip-address* [**group** *group-name* [**passive**]] **Example:** `Device(config)#ip route static bfd TenGigabitEthernet1/0/1 10.10.10.2 group group1 passive`	Specifies a static route BFD neighbor. • The *interface-type*, *interface-number*, and *ip-address* arguments are required because BFD support exists only for directly connected neighbors.
Step 8	**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*] **Example:** `Device(config)#ip route 10.0.0.0 255.0.0.0`	Specifies a static route BFD neighbor.
Step 9	**exit** **Example:** `Device(config)#exit`	Exits global configuration mode and returns to privileged EXEC mode.
Step 10	**show ip static route** **Example:** `Device#show ip static route`	(Optional) Displays static route database information.
Step 11	**show ip static route bfd** **Example:** `Device#show ip static route bfd`	(Optional) Displays information about the static BFD configuration from the configured BFD groups and nongroup entries.
Step 12	**exit** **Example:** `Device#exit`	Exits privileged EXEC mode and returns to user EXEC mode.

Configuring BFD Echo Mode

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. Echo packets are sent by the forwarding engine and forwarded back along the same path in order to perform detection--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; therefore, the number of BFD control packets that are sent out between two BFD neighbors is reduced. In addition, because the forwarding engine is testing the forwarding path on the remote (neighbor) system without involving the remote system, there is an opportunity to improve the interpacket delay variance, thereby achieving quicker failure detection times than when 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

- 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.

Restrictions

BFD echo mode does not work in conjunction with Unicast Reverse Path Forwarding (uRPF) configuration. If BFD echo mode and uRPF configurations are enabled, then the sessions will flap.

Disabling BFD Echo Mode Without Asymmetry

The steps in this procedure show how to disable BFD echo mode without asymmetry—no echo packets will be sent by the device, and the device will not forward BFD echo packets that are received from any neighbor devices.

Repeat the steps in this procedure for each BFD Device.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device>enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device#configure terminal`	Enters global configuration mode.
Step 3	**no bfd echo** **Example:** `Device(config)#no bfd echo`	Disables BFD echo mode. Use the **no** form to disable BFD echo mode.

	Command or Action	Purpose
Step 4	**end** **Example:** `Device(config)#end`	Exits global configuration mode and returns to privileged EXEC mode.

Creating and Configuring BFD Templates

You can configure a single-hop template to specify a set of BFD interval values. BFD interval values specified as part of the BFD template are not specific to a single interface.

Note Configuring BFD-template will disable echo mode.

Configuring a Single-Hop Template

Perform this task to create a BFD single-hop template and configure BFD interval timers.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device>enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device#configure terminal`	Enters global configuration mode.
Step 3	**bfd-template single-hop** *template-name* **Example:** `Device(config)#bfd-template single-hop bfdtemplate1`	Creates a single-hop BFD template and enters BFD configuration mode.
Step 4	**interval min-tx** *milliseconds* **min-rx** *milliseconds* **multiplier** *multiplier-value* **Example:** `Device(bfd-config)#interval min-tx 120 min-rx 100 multiplier 3`	Configures the transmit and receive intervals between BFD packets, and specifies the number of consecutive BFD control packets that must be missed before BFD declares that a peer is unavailable.
Step 5	**end** **Example:**	Exits BFD configuration mode and returns the device to privileged EXEC mode.

	Command or Action	Purpose
	`Device(bfd-config)#end`	

Monitoring and Troubleshooting BFD

This section describes how to retrieve BFD information for maintenance and troubleshooting. The commands in these tasks can be entered in any order as needed.

This section contains information for monitoring and troubleshooting BFD for the following Cisco platforms:

Monitoring and Troubleshooting BFD

To monitor or troubleshoot BFD, perform one or more of the steps in this section.

SUMMARY STEPS

1. **enable**
2. **show bfd neighbors** [**details**]
3. **debug bfd** [**packet** | **event**]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device>enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**show bfd neighbors** [**details**] **Example:** `Device#show bfd neighbors details`	(Optional) Displays the BFD adjacency database. The **details** keyword shows all BFD protocol parameters and timers per neighbor.
Step 3	**debug bfd** [**packet**	**event**] **Example:** `Device#debug bfd packet`	(Optional) Displays debugging information about BFD packets.

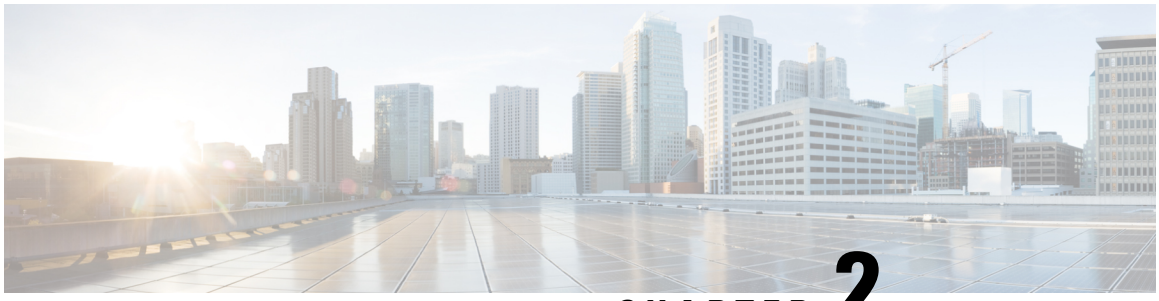
Feature History for Configuring Bidirectional Forwarding Detection

This table provides release and related information for the features explained in this module.

These features are available in all the releases subsequent to the one they were introduced in, unless noted otherwise.

Release	Feature	Feature Information
Cisco IOS XE Everest 16.5.1a	Bidirectional Forwarding Detection	BFD is a detection protocol that is designed to provide fast forwarding path failure detection times for all media types, encapsulations, topologies, and routing protocols.

Use the Cisco Feature Navigator to find information about platform and software image support. To access Cisco Feature Navigator, go to http://www.cisco.com/go/cfn.



CHAPTER 2

Configuring BFD Support for EIGRP IPv6

- Prerequisites for BFD Support for EIGRP IPv6, on page 25
- Restrictions for BFD Support for EIGRP IPv6, on page 25
- Information About BFD Support for EIGRP IPv6, on page 25
- How to Configure BFD Support for EIGRP IPv6, on page 26
- Configuration Examples for BFD Support for EIGRP IPv6, on page 29
- Additional References, on page 30
- Feature History for Configuring BFD Support for EIGRP IPv6, on page 31

Prerequisites for BFD Support for EIGRP IPv6

EIGRP IPv6 sessions have a shutdown option in router, address family, and address-family interface configuration modes. To enable BFD support on EIGRP IPv6 sessions, the routing process should be in no shut mode in the abovementioned modes.

Restrictions for BFD Support for EIGRP IPv6

- The BFD Support for EIGRP IPv6 feature is supported only in EIGRP named mode.
- EIGRP supports only single-hop Bidirectional Forwarding Detection (BFD).
- The BFD Support for EIGRP IPv6 feature is not supported on passive interfaces.

Information About BFD Support for EIGRP IPv6

The BFD Support for EIGRP IPv6 feature provides Bidirectional Forwarding Detection (BFD) support for Enhanced Interior Gateway Routing Protocol (EIGRP) IPv6 sessions. It facilitates rapid fault detection and alternate-path selection in EIGRP IPv6 topologies. BFD is a detection protocol that provides a consistent failure-detection method for network administrators. Network administrators use BFD to detect forwarding path failures at a uniform rate and not at variable rates for 'Hello' mechanisms of different routing protocols. This failure-detection methodology ensures easy network profiling and planning and consistent and predictable reconvergence time. This document provides information about BFD support for EIGRP IPv6 networks and explains how to configure BFD support in EIGRP IPv6 networks.

How to Configure BFD Support for EIGRP IPv6

The following sections provide information on configuring BFD support for EIGRP IPv6 for an interface and all interfaces.

Configuring BFD Support on All Interfaces

The following steps show how to configure BFD support on all interfaces:

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**ipv6 unicast-routing** **Example:** `Device(config)# ipv6 unicast-routing`	Enables the forwarding of IPv6 unicast datagrams.
Step 4	**interface** *type number* **Example:** `Device(config)# interface ethernet0/0`	Specifies the interface type and number, and enters the interface configuration mode.
Step 5	**ipv6 address** *ipv6-address/prefix-length* **Example:** `Device(config-if)# ipv6 address 2001:DB8:A:B::1/64`	Configures an IPv6 address.
Step 6	**bfd interval** *milliseconds* **min_rx** *milliseconds* **multiplier** *interval-multiplier* **Example:** `Device(config-if)# bfd interval 50 min_rx 50 multiplier 3`	Sets the baseline BFD session parameters on an interface.
Step 7	**exit** **Example:** `Device(config-if)# exit`	Exits interface configuration mode and returns to global configuration mode.
Step 8	**router eigrp** *virtual-name* **Example:** `Device(config)# router eigrp name`	Specifies an EIGRP routing process and enters router configuration mode.

	Command or Action	Purpose
Step 9	**address-family ipv6 autonomous-system** *as-number* **Example:** `Device(config-router)# address-family ipv6 autonomous-system 3`	Enters address family configuration mode for IPv6 and configures an EIGRP routing instance.
Step 10	**eigrp router-id** *ip-address* **Example:** `Device(config-router-af)# eigrp router-id 172.16.1.3`	Sets the device ID used by EIGRP for this address family when EIGRP peers communicate with their neighbors.
Step 11	**af-interface default** **Example:** `Device(config-router-af)# af-interface default`	Configures interface-specific commands on all interfaces that belong to an address family in EIGRP named mode configurations. Enters address-family interface configuration mode.
Step 12	**bfd** **Example:** `Device(config-router-af-interface)# bfd`	Enables BFD on all interfaces.
Step 13	**End** **Example:** `Device(config-router-af-interface)# end`	Exits address-family interface configuration mode and returns to privileged EXEC mode.
Step 14	**show eigrp address-family ipv6 neighbors detail** **Example:** `Device# show eigrp address-family ipv6 neighbors detail`	(Optional) Displays detailed information about the neighbors that are discovered by EIGRP with BFD enabled on an interface.
Step 15	**show bfd neighbors** **Example:** `Device# show bfd neighbors`	(Optional) Displays BFD information to neighbors.

Configuring BFD Support on an Interface

The following steps show how to configure BFD support on an interface:

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **ipv6 unicast-routing**
4. **interface** *type number*
5. **ipv6 address** *ipv6-address /prefix-length*
6. **bfd interval** *milliseconds* **min_rx** *milliseconds* **multiplier** *interval-multiplier*
7. **exit**

8. **router eigrp** *virtual-name*
9. **address-family ipv6 autonomous-system** *as-number*
10. **eigrp router-id** *ip-address*
11. **af-interface** *interface-type interface-number*
12. **bfd**
13. **end**
14. **show eigrp address-family ipv6 neighbors**
15. **show bfd neighbors**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**ipv6 unicast-routing** **Example:** `Device(config)# ipv6 unicast-routing`	Enables the forwarding of IPv6 unicast datagrams.
Step 4	**interface** *type number* **Example:** `Device(config)# interface ethernet0/0`	Specifies the interface type and number, and enters the interface configuration mode.
Step 5	**ipv6 address** *ipv6-address /prefix-length* **Example:** `Device(config-if)# ipv6 address 2001:DB8:A:B::1/64`	Configures an IPv6 address.
Step 6	**bfd interval** *milliseconds* **min_rx** *milliseconds* **multiplier** *interval-multiplier* **Example:** `Device(config-if)# bfd interval 50 min_rx 50 multiplier 3`	Sets the baseline BFD session parameters on an interface.
Step 7	**exit** **Example:** `Device(config-if)# exit`	Exits interface configuration mode and returns to global configuration mode.
Step 8	**router eigrp** *virtual-name* **Example:** `Device(config)# router eigrp name`	Specifies an EIGRP routing process and enters router configuration mode.

	Command or Action	Purpose
Step 9	**address-family ipv6 autonomous-system** *as-number* **Example:** `Device(config-router)# address-family ipv6 autonomous-system 3`	Enters address family configuration mode for IPv6 and configures an EIGRP routing instance.
Step 10	**eigrp router-id** *ip-address* **Example:** `Device(config-router-af)# eigrp router-id 172.16.1.3`	Sets the device ID used by EIGRP for this address family when EIGRP peers communicate with their neighbors.
Step 11	**af-interface** *interface-type interface-number* **Example:** `Device(config-router-af)# af-interface ethernet0/0`	Configures interface-specific commands on an interface that belongs to an address family in an EIGRP named mode configuration. Enters address-family interface configuration mode.
Step 12	**bfd** **Example:** `Device(config-router-af-interface)# bfd`	Enables BFD on the specified interface.
Step 13	**end** **Example:** `Device(config-router-af-interface)# end`	Exits address-family interface configuration mode and returns to privileged EXEC mode.
Step 14	**show eigrp address-family ipv6 neighbors** **Example:** `Device# show eigrp address-family ipv6 neighbors`	(Optional) Displays neighbors for which have BFD enabled.
Step 15	**show bfd neighbors** **Example:** `Device# show bfd neighbors`	(Optional) Displays BFD information to neighbors.

Configuration Examples for BFD Support for EIGRP IPv6

The following sections provide configuration examples for BFD support for EIGRP:

Example: Configuring BFD Support on All Interfaces

```
Device> enable
Device# configure terminal
Device(config)# ipv6 unicast-routing
Device(config)# interface Ethernet0/0
Device(config-if)# ipv6 address 2001:0DB8:1::12/64
Device(config-if)# bfd interval 50 min_rx 50 multiplier 3
Device(config-if)# exit
Device(config)# router eigrp name
Device(config-router)# address-family ipv6 unicast autonomous-system 1
```

```
Device(config-router-af)# eigrp router-id 172.16.0.1
Device(config-router-af)# af-interface default
Device(config-router-af-interface)# bfd
Device(config-router-af-interface)# end
```

The following example displays the output for the **show eigrp address-family ipv6 neighbors detail** command.

```
Device# show eigrp address-family ipv6 neighbors detail
EIGRP-IPv6 VR(test) Address-Family Neighbors for AS(5)
H   Address                 Interface              Hold Uptime   SRTT   RTO  Q  Seq
                                                   (sec)         (ms)       Cnt Num
0   Link-local address:     Et0/0                    14 00:02:04    1  4500  0  4
    FE80::10:2
   Version 23.0/2.0, Retrans: 2, Retries: 0, Prefixes: 1
   Topology-ids from peer - 0
   Topologies advertised to peer:   base

Max Nbrs: 0, Current Nbrs: 0

BFD sessions
NeighAddr         Interface
FE80::10:2            Ethernet0/0
```

The following example displays the output for the **show bfd neighbor** command.

```
Device# show bfd neighbors

IPv6 Sessions
NeighAddr                              LD/RD         RH/RS     State     Int
FE80::10:2                              2/0          Down      Down      Et0/0
```

Example: Configuring BFD Support on an Interface

The following example shows how to configure BFD Support on an interface:

```
Device> enable
Device# configure terminal
Device(config)# ipv6 unicast-routing
Device(config)# Ethernet0/0
Device(config-if)# ipv6 address 2001:DB8:A:B::1/64
Device(config-if)# bfd interval 50 min_rx 50 multiplier 3
Device(config-if)# exit
Device(config)# router eigrp name
Device(config-router)# address-family ipv6 autonomous-system 3
Device(config-router-af)# af-interface Ethernet0/0
Device(config-router-af-interface)# bfd
Device(config-router-af-interface)# end
```

Additional References

Related Documents

Related Topic	Document Title
BFD commands: complete command syntax, command mode, command history, defaults, usage guidelines, and examples.	See the *IP Routing* section of the *Command Reference (Catalyst 9300 Series Switches)*

Related Topic	Document Title
EIGRP commands: complete command syntax, command mode, command history, defaults, usage guidelines, and examples.	See the *IP Routing* section of the *Command Reference (Catalyst 9300 Series Switches)*
Configuring EIGRP	See the *Routing* section of the *Software Configuration Guide (Catalyst 9300 Switches)*

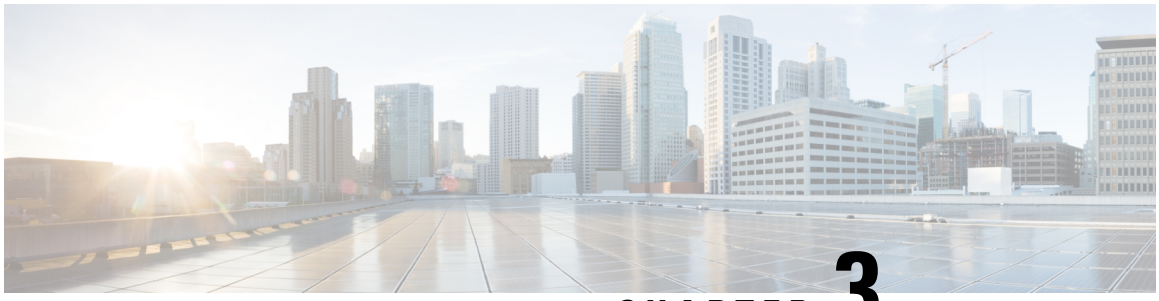
Feature History for Configuring BFD Support for EIGRP IPv6

This table provides release and related information for the features explained in this module.

These features are available in all the releases subsequent to the one they were introduced in, unless noted otherwise.

Release	Feature	Feature Information
Cisco IOS XE Gibraltar 16.11.1	BFD Support for EIGRP IPv6	The BFD Support for EIGRP IPv6 feature provides BFD support for EIGRP IPv6 sessions.

Use the Cisco Feature Navigator to find information about platform and software image support. To access Cisco Feature Navigator, go to http://www.cisco.com/go/cfn.



CHAPTER 3

Configuring MSDP

- Information About Configuring MSDP, on page 33
- How to Configure MSDP, on page 36
- Monitoring and Maintaining MSDP, on page 54
- Configuration Examples for Configuring MSDP, on page 55
- Feature History for Multicast Source Discovery Protocol, on page 56

Information About Configuring MSDP

This section describes how to configure the Multicast Source Discovery Protocol (MSDP on the switch. The MSDP connects multiple Protocol-Independent Multicast sparse-mode (PIM-SM) domains.

MSDP is not fully supported in this software release because of a lack of support for Multicast Border Gateway Protocol (MBGP), which works closely with MSDP. However, it is possible to create default peers that MSDP can operate with if MBGP is not running.

Note To use this feature, the active switch must be running the Network Advantage feature set.

MSDP Overview

MSDP allows multicast sources for a group to be known to all rendezvous points (RPs) in different domains. Each PIM-SM domain uses its own RPs and does not depend on RPs in other domains. An RP runs MSDP over the Transmission Control Protocol (TCP) to discover multicast sources in other domains.

An RP in a PIM-SM domain has an MSDP peering relationship with MSDP-enabled devices in another domain. The peering relationship occurs over a TCP connection, primarily exchanging a list of sources sending to multicast groups. The TCP connections between RPs are achieved by the underlying routing system. The receiving RP uses the source lists to establish a source path.

The purpose of this topology is to have domains discover multicast sources in other domains. If the multicast sources are of interest to a domain that has receivers, multicast data is delivered over the normal, source-tree building mechanism in PIM-SM. MSDP is also used to announce sources sending to a group. These announcements must originate at the domain's RP.

MSDP depends heavily on the Border Gateway Protocol (BGP) or MBGP for interdomain operation. We recommend that you run MSDP in RPs in your domain that are RPs for sources sending to global groups to be announced to the Internet.

MSDP Operation

When a source sends its first multicast packet, the first-hop router (*designated router* or RP) directly connected to the source sends a PIM register message to the RP. The RP uses the register message to register the active source and to forward the multicast packet down the shared tree in the local domain. With MSDP configured, the RP also forwards a source-active (SA) message to all MSDP peers. The SA message identifies the source, the group the source is sending to, and the address of the RP or the originator ID (the IP address of the interface used as the RP address), if configured.

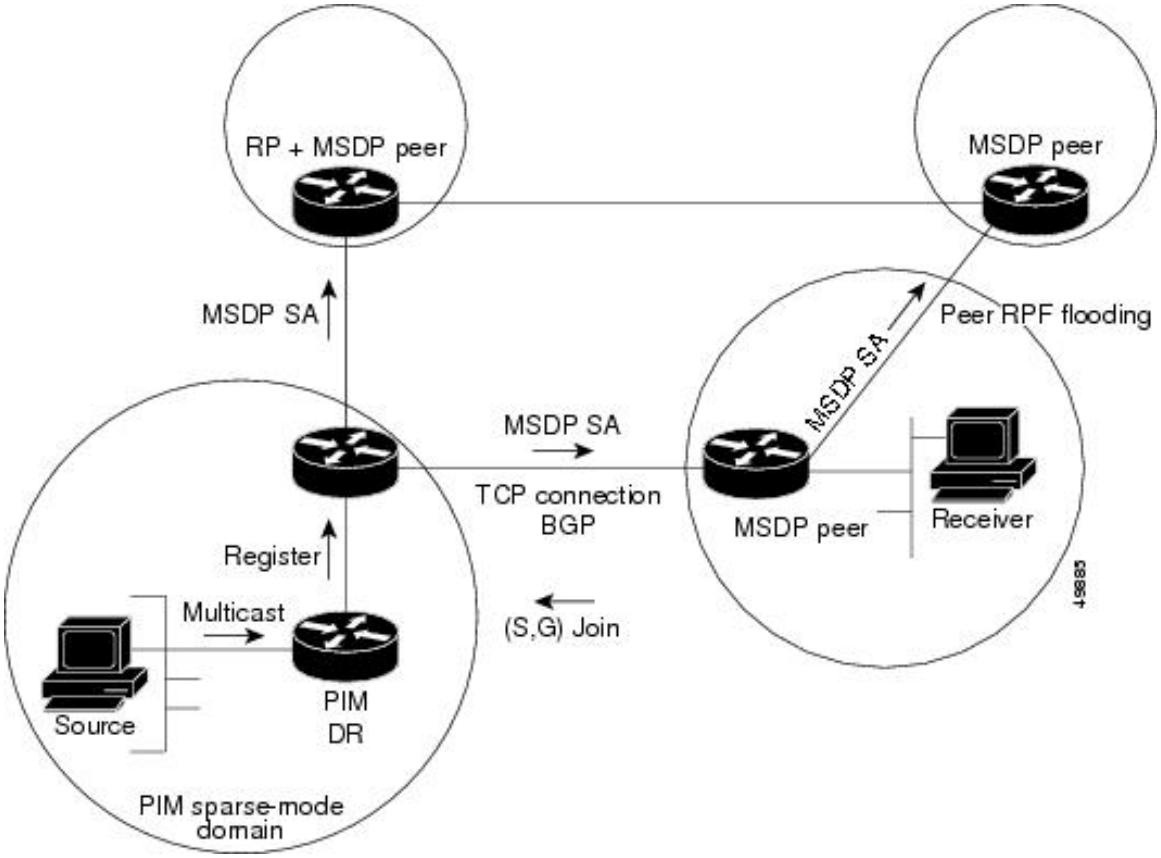
Each MSDP peer receives and forwards the SA message away from the originating RP to achieve peer reverse-path flooding (RPF). The MSDP device examines the BGP or MBGP routing table to discover which peer is the next hop toward the originating RP of the SA message. Such a peer is called an *RPF peer* (reverse-path forwarding peer). The MSDP device forwards the message to all MSDP peers other than the RPF peer. For information on how to configure an MSDP peer when BGP and MBGP are not supported, see the Configuring a Default MSDP Peer, on page 36.

If the MSDP peer receives the same SA message from a non-RPF peer toward the originating RP, it drops the message. Otherwise, it forwards the message to all its MSDP peers.

The RP for a domain receives the SA message from an MSDP peer. If the RP has any join requests for the group the SA message describes and if the (*,G) entry exists with a nonempty outgoing interface list, the domain is interested in the group, and the RP triggers an (S,G) join toward the source. After the (S,G) join reaches the source's DR, a branch of the source tree has been built from the source to the RP in the remote domain. Multicast traffic can now flow from the source across the source tree to the RP and then down the shared tree in the remote domain to the receiver.

Figure 1: MSDP Running Between RP Peers

This figure shows MSDP operating between two MSDP peers. PIM uses MSDP as the standard mechanism to register a source with the RP of a domain. When MSDP is configured, this sequence occurs.



By default, the switch does not cache source or group pairs from received SA messages. When the switch forwards the MSDP SA information, it does not store it in memory. Therefore, if a member joins a group soon after an SA message is received by the local RP, that member needs to wait until the next SA message to hear about the source. This delay is known as join latency.

Local RPs can send SA requests and get immediate responses for all active sources for a given group. By default, the switch does not send any SA request messages to its MSDP peers when a new member joins a group and wants to receive multicast traffic. The new member waits to receive the next periodic SA message.

If you want a new member of a group to learn the active multicast sources in a connected PIM sparse-mode domain that are sending to a group, configure the switch to send SA request messages to the specified MSDP peer when a new member joins a group.

MSDP Benefits

MSDP has these benefits:

- It breaks up the shared multicast distribution tree. You can make the shared tree local to your domain. Your local members join the local tree, and join messages for the shared tree never need to leave your domain.

- PIM sparse-mode domains can rely only on their own RPs, decreasing reliance on RPs in another domain. This increases security because you can prevent your sources from being known outside your domain.
- Domains with only receivers can receive data without globally advertising group membership.
- Global source multicast routing table state is not required, saving memory.

How to Configure MSDP

Default MSDP Configuration

MSDP is not enabled, and no default MSDP peer exists.

Configuring a Default MSDP Peer

Before you begin

Configure an MSDP peer.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device>enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device#configure terminal`	Enters global configuration mode.
Step 3	**ip msdp default-peer** *ip-address* | *name* [**prefix-list** *list*] **Example:** `Device(config)#ip msdp default-peer 10.1.1.1 prefix-list site-a`	Defines a default peer from which to accept all MSDP SA messages. • For *ip-address* | *name*, enter the IP address or Domain Name System (DNS) server name of the MSDP default peer. • (Optional) For **prefix-list** *list*, enter the list name that specifies the peer to be the default peer only for the listed prefixes. You can have multiple active default peers when you have a prefix list associated with each. When you enter multiple **ip msdp default-peer** commands with the **prefix-list** keyword, you use all the default peers at the same time for different RP

	Command or Action	Purpose
		prefixes. This syntax is typically used in a service provider cloud that connects stub site clouds. When you enter multiple **ip msdp default-peer** commands without the **prefix-list** keyword, a single active peer accepts all SA messages. If that peer fails, the next configured default peer accepts all SA messages. This syntax is typically used at a stub site.
Step 4	**ip prefix-list** *name* [**description** *string*] | **seq** *number* {**permit** | **deny**} *network length* **Example:** `Device(config)#prefix-list site-a seq 3 permit 12 network length 128`	(Optional) Creates a prefix list using the name specified in Step 2. • (Optional) For **description** *string*, enter a description of up to 80 characters to describe this prefix list. • For **seq** *number*, enter the sequence number of the entry. The range is 1 to 4294967294. • The **deny** keyword denies access to matching conditions. • The **permit** keyword permits access to matching conditions. • For *network length*, specify the network number and length (in bits) of the network mask that is permitted or denied.
Step 5	**ip msdp description** {*peer-name* | *peer-address*} *text* **Example:** `Device(config)#ip msdp description peer-name site-b`	(Optional) Configures a description for the specified peer to make it easier to identify in a configuration or in **show** command output. By default, no description is associated with an MSDP peer.
Step 6	**end** **Example:** `Device(config)#end`	Returns to privileged EXEC mode.
Step 7	**show running-config** **Example:** `Device#show running-config`	Verifies your entries.
Step 8	**copy running-config startup-config** **Example:**	(Optional) Saves your entries in the configuration file.

	Command or Action	Purpose
	`Device#copy running-config startup-config`	

Caching Source-Active State

If you want to sacrifice some memory in exchange for reducing the latency of the source information, you can configure the device to cache SA messages. Perform the following steps to enable the caching of source/group pairs:

Follow these steps to enable the caching of source/group pairs:

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device>enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device#configure terminal`	Enters global configuration mode.
Step 3	**ip msdp cache-sa-state** [**list** *access-list-number*] **Example:** `Device(config)#ip msdp cache-sa-state 100`	Enables the caching of source/group pairs (create an SA state). Those pairs that pass the access list are cached. For **list** *access-list-number*, the range is 100 to 199. **Note** An alternative to this command is the **ip msdp sa-reques** global configuration command, which causes the device to send an SA request message to the MSDP peer when a new member for a group becomes active.
Step 4	**access-list** *access-list-number* {**deny** | **permit**} *protocol source source-wildcard destination destination-wildcard* **Example:** `Device(config)#access-list 100 permit ip 171.69.0.0 0.0.255.255 224.2.0.0 0.0.255.255`	Creates an IP extended access list, repeating the command as many times as necessary. • For *access-list-number*, the range is 100 to 199. Enter the same number created in Step 2. • The **deny** keyword denies access if the conditions are matched. The **permit** keyword permits access if the conditions are matched. • For *protocol*, enter **ip** as the protocol name.

	Command or Action	Purpose
		• For *source*, enter the number of the network or host from which the packet is being sent. • For *source-wildcard*, enter the wildcard bits in dotted decimal notation to be applied to the source. Place ones in the bit positions that you want to ignore. • For *destination*, enter the number of the network or host to which the packet is being sent. • For *destination-wildcard*, enter the wildcard bits in dotted decimal notation to be applied to the destination. Place ones in the bit positions that you want to ignore. Recall that the access list is always terminated by an implicit deny statement for everything.
Step 5	**end** **Example:** `Device(config)#end`	Returns to privileged EXEC mode.
Step 6	**show running-config** **Example:** `Device#show running-config`	Verifies your entries.
Step 7	**copy running-config startup-config** **Example:** `Device#copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

Requesting Source Information from an MSDP Peer

If you want a new member of a group to learn the active multicast sources in a connected PIM sparse-mode domain that are sending to a group, perform this task for the device to send SA request messages to the specified MSDP peer when a new member joins a group. The peer replies with the information in its SA cache. If the peer does not have a cache configured, this command has no result. Configuring this feature reduces join latency but sacrifices memory.

Follow these steps to configure the device to send SA request messages to the MSDP peer when a new member joins a group and wants to receive multicast traffic:

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device>enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device#configure terminal`	Enters global configuration mode.
Step 3	**ip msdp sa-request** {*ip-address* | *name*} **Example:** `Device(config)#ip msdp sa-request 171.69.1.1`	Configure the device to send SA request messages to the specified MSDP peer. For *ip-address* | *name*, enter the IP address or name of the MSDP peer from which the local device requests SA messages when a new member for a group becomes active. Repeat the command for each MSDP peer that you want to supply with SA messages.
Step 4	**end** **Example:** `Device(config)#end`	Returns to privileged EXEC mode.
Step 5	**show running-config** **Example:** `Device#show running-config`	Verifies your entries.
Step 6	**copy running-config startup-config** **Example:** `Device#copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

Controlling Source Information that Your Switch Originates

You can control the multicast source information that originates with your device:

- Sources you advertise (based on your sources)
- Receivers of source information (based on knowing the requestor)

For more information, see the Redistributing Sources, on page 41 and the Filtering Source-Active Request Messages, on page 43.

Redistributing Sources

SA messages originate on RPs to which sources have registered. By default, any source that registers with an RP is advertised. The *A flag* is set in the RP when a source is registered, which means the source is advertised in an SA unless it is filtered.

Follow these steps to further restrict which registered sources are advertised:

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device>enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device#configure terminal`	Enters global configuration mode.
Step 3	**ip msdp redistribute** [**list** *access-list-name*] [**asn** *aspath-access-list-number*] [**route-map** *map*] **Example:** `Device(config)#ip msdp redistribute list 21`	Configures which (S,G) entries from the multicast routing table are advertised in SA messages. By default, only sources within the local domain are advertised. • (Optional) **list** *access-list-name*— Enters the name or number of an IP standard or extended access list. The range is 1 to 99 for standard access lists and 100 to 199 for extended lists. The access list controls which local sources are advertised and to which groups they send. • (Optional) **asn** *aspath-access-list-number*—Enters the IP standard or extended access list number in the range 1 to 199. This access list number must also be configured in the **ip as-path access-list** command. • (Optional) **route-map** *map*—Enters the IP standard or extended access list number in the range 1 to 199. This access list number must also be configured in the **ip as-path access-list** command. The device advertises (S,G) pairs according to the access list or autonomous system path access list.

	Command or Action	Purpose
Step 4	Use one of the following: • `access-list` *access-list-number* { `deny` | `permit` } *source* [*source-wildcard*] • `access-list` *access-list-number* { `deny` | `permit` } *protocol source source-wildcard destination destination-wildcard* **Example:** `Device(config)#`**`access list 21 permit 194.1.22.0`** or `Device(config)#`**`access list 21 permit ip 194.1.22.0 1.1.1.1 194.3.44.0 1.1.1.1`**	Creates an IP standard access list, repeating the command as many times as necessary. or Creates an IP extended access list, repeating the command as many times as necessary. • *access-list-number*—Enters the same number created in Step 2. The range is 1 to 99 for standard access lists and 100 to 199 for extended lists. • **deny**—Denies access if the conditions are matched. The **permit** keyword permits access if the conditions are matched. • *protocol*—Enters **ip** as the protocol name. • *source*—Enters the number of the network or host from which the packet is being sent. • *source-wildcard*—Enters the wildcard bits in dotted decimal notation to be applied to the source. Place ones in the bit positions that you want to ignore. • *destination*—Enters the number of the network or host to which the packet is being sent. • *destination-wildcard*—Enters the wildcard bits in dotted decimal notation to be applied to the destination. Place ones in the bit positions that you want to ignore. Recall that the access list is always terminated by an implicit deny statement for everything.
Step 5	**end** **Example:** `Device(config)#`**`end`**	Returns to privileged EXEC mode.
Step 6	**show running-config** **Example:** `Device#`**`show running-config`**	Verifies your entries.
Step 7	**copy running-config startup-config** **Example:** `Device#`**`copy running-config startup-config`**	(Optional) Saves your entries in the configuration file.

Filtering Source-Active Request Messages

By default, only device that are caching SA information can respond to SA requests. By default, such a device honors all SA request messages from its MSDP peers and supplies the IP addresses of the active sources.

However, you can configure the device to ignore all SA requests from an MSDP peer. You can also honor only those SA request messages from a peer for groups described by a standard access list. If the groups in the access list pass, SA request messages are accepted. All other such messages from the peer for other groups are ignored.

To return to the default setting, use the **no ip msdp filter-sa-request** {*ip-address*| *name*} global configuration command.

Follow these steps to configure one of these options:

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device>enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device#configure terminal`	Enters global configuration mode.
Step 3	Use one of the following: • `ip msdp filter-sa-request` { *ip-address name* } • `ip msdp filter-sa-request` { *ip-address name* } `list` *access-list-number* **Example:** `Device(config)#ip msdp filter sa-request 171.69.2.2`	Filters all SA request messages from the specified MSDP peer. or Filters SA request messages from the specified MSDP peer for groups that pass the standard access list. The access list describes a multicast group address. The range for the access-list-number is 1 to 99.
Step 4	**access-list** *access-list-number* {**deny** | **permit**} *source* [*source-wildcard*] **Example:** `Device(config)#access-list 1 permit 192.4.22.0 0.0.0.255`	Creates an IP standard access list, repeating the command as many times as necessary. • For *access-list-number*, the range is 1 to 99. • The **deny** keyword denies access if the conditions are matched. The **permit** keyword permits access if the conditions are matched. • For *source*, enter the number of the network or host from which the packet is being sent.

	Command or Action	Purpose
		• (Optional) For *source-wildcard*, enter the wildcard bits in dotted decimal notation to be applied to the source. Place ones in the bit positions that you want to ignore. Recall that the access list is always terminated by an implicit deny statement for everything.
Step 5	**end** **Example:** `Device(config)#end`	Returns to privileged EXEC mode.
Step 6	**show running-config** **Example:** `Device#show running-config`	Verifies your entries.
Step 7	**copy running-config startup-config** **Example:** `Device#copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

Controlling Source Information that Your Switch Forwards

By default, the device forwards all SA messages it receives to all its MSDP peers. However, you can prevent outgoing messages from being forwarded to a peer by using a filter or by setting a time-to-live (TTL) value.

Using a Filter

By creating a filter, you can perform one of these actions:

- Filter all source/group pairs
- Specify an IP extended access list to pass only certain source/group pairs
- Filter based on match criteria in a route map

Follow these steps to apply a filter:

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:**	Enables privileged EXEC mode. • Enter your password if prompted.

	Command or Action	Purpose
	`Device>enable`	
Step 2	**configure terminal** **Example:** `Device#configure terminal`	Enters global configuration mode.

Step 3	Use one of the following:
	• **ip msdp sa-filter out** { *ip-address name* }
	• **ip msdp sa-filter out** { *ip-address name* } `list` *access-list-number*
	• **ip msdp sa-filter out** { *ip-address name* } `route-map` *map-tag*
	Example:
	`Device(config)#ip msdp sa-filter out switch.cisco.com`
	or
	`Device(config)#ip msdp sa-filter out list 100`
	or
	`Device(config)#ip msdp sa-filter out switch.cisco.com route-map 22`
		• Filters all SA messages to the specified MSDP peer.
		• Passes only those SA messages that pass the IP extended access list to the specified peer. The range for the extended *access-list-number* is 100 to 199.
		If both the **list** and the **route-map** keywords are used, all conditions must be true to pass any (S,G) pair in outgoing SA messages.
		• Passes only those SA messages that meet the match criteria in the route map *map-tag* to the specified MSDP peer.
		If all match criteria are true, a **permit** from the route map passes routes through the filter. A **deny** filters routes.

Step 4	**access-list** *access-list-number* {**deny** | **permit**} *protocol source source-wildcard destination destination-wildcard* **Example:** `Device(config)#access list 100 permit ip 194.1.22.0 1.1.1.1 194.3.44.0 1.1.1.1`	(Optional) Creates an IP extended access list, repeating the command as many times as necessary. • For *access-list-number*, enter the number specified in Step 2. • The **deny** keyword denies access if the conditions are matched. The **permit** keyword permits access if the conditions are matched. • For *protocol*, enter **ip** as the protocol name. • For *source*, enter the number of the network or host from which the packet is being sent.

	Command or Action	Purpose
		• For *source-wildcard*, enter the wildcard bits in dotted decimal notation to be applied to the source. Place ones in the bit positions that you want to ignore. • For *destination*, enter the number of the network or host to which the packet is being sent. • For *destination-wildcard*, enter the wildcard bits in dotted decimal notation to be applied to the destination. Place ones in the bit positions that you want to ignore. Recall that the access list is always terminated by an implicit deny statement for everything.
Step 5	**end** **Example:** `Device(config)#end`	Returns to privileged EXEC mode.
Step 6	**show running-config** **Example:** `Device#show running-config`	Verifies your entries.
Step 7	**copy running-config startup-config** **Example:** `Device#copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

Using TTL to Limit the Multicast Data Sent in SA Messages

You can use a TTL value to control what data is encapsulated in the first SA message for every source. Only multicast packets with an IP-header TTL greater than or equal to the *ttl* argument are sent to the specified MSDP peer. For example, you can limit internal traffic to a TTL of 8. If you want other groups to go to external locations, you must send those packets with a TTL greater than 8.

Follow these steps to establish a TTL threshold:

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device>enable`	Enables privileged EXEC mode. • Enter your password if prompted.

	Command or Action	Purpose
Step 2	**configure terminal** **Example:** `Device#configure terminal`	Enters global configuration mode.
Step 3	**ip msdp ttl-threshold** {*ip-address* | *name*} *ttl* **Example:** `Device(config)#ip msdp ttl-threshold switch.cisco.com 0`	Limits which multicast data is encapsulated in the first SA message to the specified MSDP peer. • For *ip-address* | *name*, enter the IP address or name of the MSDP peer to which the TTL limitation applies. • For *ttl*, enter the TTL value. The default is 0, which means all multicast data packets are forwarded to the peer until the TTL is exhausted. The range is 0 to 255.
Step 4	**end** **Example:** `Device(config)#end`	Returns to privileged EXEC mode.
Step 5	**show running-config** **Example:** `Device#show running-config`	Verifies your entries.
Step 6	**copy running-config startup-config** **Example:** `Device#copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

Controlling Source Information that Your Switch Receives

By default, the device receives all SA messages that its MSDP RPF peers send to it. However, you can control the source information that you receive from MSDP peers by filtering incoming SA messages. In other words, you can configure the device to not accept them.

You can perform one of these actions:

- Filter all incoming SA messages from an MSDP peer
- Specify an IP extended access list to pass certain source/group pairs
- Filter based on match criteria in a route map

Follow these steps to apply a filter:

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device>enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device#configure terminal`	Enters global configuration mode.

Step 3	Use one of the following:
	• **ip msdp sa-filter in** { *ip-address name* }
	• **ip msdp sa-filter in** { *ip-address name* } `list` *access-list-number*
	• **ip msdp sa-filter in** { *ip-address name* } `route-map` *map-tag*
	Example:
	`Device(config)#ip msdp sa-filter in switch.cisco.com`
	or
	`Device(config)#ip msdp sa-filter in list 100`
	or
	`Device(config)#ip msdp sa-filter in switch.cisco.com route-map 22`
		• Filters all SA messages to the specified MSDP peer.
		• Passes only those SA messages from the specified peer that pass the IP extended access list. The range for the extended *access-list-number* is 100 to 199.
		If both the **list** and the **route-map** keywords are used, all conditions must be true to pass any (S,G) pair in outgoing SA messages.
		• Passes only those SA messages from the specified MSDP peer that meet the match criteria in the route map *map-tag*.
		If all match criteria are true, a **permit** from the route map passes routes through the filter. A **deny** filters routes.

Step 4	**access-list** *access-list-number* {**deny** | **permit**} *protocol source source-wildcard destination destination-wildcard* **Example:** `Device(config)#access list 100 permit ip 194.1.22.0 1.1.1.1 194.3.44.0 1.1.1.1`	(Optional) Creates an IP extended access list, repeating the command as many times as necessary. • *access-list-number*, enter the number specified in Step 2. • The **deny** keyword denies access if the conditions are matched. The **permit** keyword permits access if the conditions are matched. • For *protocol*, enter **ip** as the protocol name.

	Command or Action	Purpose
		• For *source*, enter the number of the network or host from which the packet is being sent. • For *source-wildcard*, enter the wildcard bits in dotted decimal notation to be applied to the source. Place ones in the bit positions that you want to ignore. • For *destination*, enter the number of the network or host to which the packet is being sent. • For *destination-wildcard*, enter the wildcard bits in dotted decimal notation to be applied to the destination. Place ones in the bit positions that you want to ignore. Recall that the access list is always terminated by an implicit deny statement for everything.
Step 5	**end** **Example:** `Device(config)#end`	Returns to privileged EXEC mode.
Step 6	**show running-config** **Example:** `Device#show running-config`	Verifies your entries.
Step 7	**copy running-config startup-config** **Example:** `Device#copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

Configuring an MSDP Mesh Group

An MSDP mesh group is a group of MSDP speakers that have fully meshed MSDP connectivity among one another. Any SA messages received from a peer in a mesh group are not forwarded to other peers in the same mesh group. Thus, you reduce SA message flooding and simplify peer-RPF flooding. Use the **ip msdp mesh-group** global configuration command when there are multiple RPs within a domain. It is especially used to send SA messages across a domain. You can configure multiple mesh groups (with different names) in a single device.

Follow these steps to create a mesh group:

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device>enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device#configure terminal`	Enters global configuration mode.
Step 3	**ip msdp mesh-group** *name* {*ip-address* | *name*} **Example:** `Devic(config)#ip msdp mesh-group 2 switch.cisco.com`	Configures an MSDP mesh group, and specifies the MSDP peer belonging to that mesh group. By default, the MSDP peers do not belong to a mesh group. • For *name*, enter the name of the mesh group. • For *ip-address* | *name*, enter the IP address or name of the MSDP peer to be a member of the mesh group. Repeat this procedure on each MSDP peer in the group.
Step 4	**end** **Example:** `Device(config)#end`	Returns to privileged EXEC mode.
Step 5	**show running-config** **Example:** `Device#show running-config`	Verifies your entries.
Step 6	**copy running-config startup-config** **Example:** `Device#copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

Shutting Down an MSDP Peer

If you want to configure many MSDP commands for the same peer and you do not want the peer to become active, you can shut down the peer, configure it, and later bring it up. When a peer is shut down, the TCP connection is terminated and is not restarted. You can also shut down an MSDP session without losing configuration information for the peer.

Follow these steps to shut down a peer:

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device>enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device#configure terminal`	Enters global configuration mode.
Step 3	**ip msdp shutdown** {*peer-name* | *peer address*} **Example:** `Device(config)#ip msdp shutdown switch.cisco.com`	Shuts down the specified MSDP peer without losing configuration information. For *peer-name* | *peer address*, enter the IP address or name of the MSDP peer to shut down.
Step 4	**end** **Example:** `Device(config)#end`	Returns to privileged EXEC mode.
Step 5	**show running-config** **Example:** `Device#show running-config`	Verifies your entries.
Step 6	**copy running-config startup-config** **Example:** `Device#copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

Including a Bordering PIM Dense-Mode Region in MSDP

You can configure MSDP on a device that borders a PIM sparse-mode region with a dense-mode region. By default, active sources in the dense-mode region do not participate in MSDP.

Note We do not recommend using the **ip msdp border sa-address** global configuration command. It is better to configure the border router in the sparse-mode domain to proxy-register sources in the dense-mode domain to the RP of the sparse-mode domain and have the sparse-mode domain use standard MSDP procedures to advertise these sources.

The **ip msdp originator-id** global configuration command also identifies an interface to be used as the RP address. If both the **ip msdp border sa-address** and the **ip msdp originator-id** global configuration commands are configured, the address derived from the **ip msdp originator-id** command specifies the RP address.

Follow these steps to configure the border router to send SA messages for sources active in the dense-mode region to the MSDP peers:

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device>enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device#configure terminal`	Enters global configuration mode.
Step 3	**ip msdp border sa-address** *interface-id* **Example:** `Device(config)#ip msdp border sa-address 0/1`	Configures the switch on the border between a dense-mode and sparse-mode region to send SA messages about active sources in the dense-mode region. For *interface-id*, specifies the interface from which the IP address is derived and used as the RP address in SA messages. The IP address of the interface is used as the Originator-ID, which is the RP field in the SA message.
Step 4	**ip msdp redistribute** [**list** *access-list-name*] [**asn** *aspath-access-list-number*] [**route-map** *map*] **Example:** `Device(config)#ip msdp redistribute list 100`	Configures which (S,G) entries from the multicast routing table are advertised in SA messages. For more information, see the #unique_71.
Step 5	**end** **Example:** `Device(config)#end`	Returns to privileged EXEC mode.

	Command or Action	Purpose
Step 6	**show running-config** **Example:** `Device#show running-config`	Verifies your entries.
Step 7	**copy running-config startup-config** **Example:** `Device#copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

Configuring an Originating Address other than the RP Address

You can allow an MSDP speaker that originates a SA message to use the IP address of the interface as the RP address in the SA message by changing the Originator ID. You might change the Originator ID in one of these cases:

- If you configure a logical RP on multiple device in an MSDP mesh group.
- If you have a device that borders a PIM sparse-mode domain and a dense-mode domain. If a device borders a dense-mode domain for a site, and sparse-mode is being used externally, you might want dense-mode sources to be known to the outside world. Because this device is not an RP, it would not have an RP address to use in an SA message. Therefore, this command provides the RP address by specifying the address of the interface.

If both the **ip msdp border sa-address** and the **ip msdp originator-id** global configuration commands are configured, the address derived from the **ip msdp originator-id** command specifies the address of the RP.

Follow these steps to allow an MSDP speaker that originates an SA message to use the IP address on the interface as the RP address in the SA message:

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device>enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device#configure terminal`	Enters global configuration mode.

	Command or Action	Purpose
Step 3	**ip msdp originator-id** *interface-id* **Example:** `Device(config)#ip msdp originator-id 0/1`	Configures the RP address in SA messages to be the address of the originating device interface. For *interface-id*, specify the interface on the local device.
Step 4	**end** **Example:** `Device(config)#end`	Returns to privileged EXEC mode.
Step 5	**show running-config** **Example:** `Device#show running-config`	Verifies your entries.
Step 6	**copy running-config startup-config** **Example:** `Device#copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

Monitoring and Maintaining MSDP

Commands that monitor MSDP SA messages, peers, state, and peer status:

Table 1: Commands for Monitoring and Maintaining MSDP

Command	Purpose
debug ip msdp [*peer-address* | *name*] [**detail**] [**routes**]	Debugs an MSDP activity.
debug ip msdp resets	Debugs MSDP peer reset reasons.
show ip msdp count [*autonomous-system-number*]	Displays the number of sources and groups originated in SA messages from each autonomous system. The **ip msdp cache-sa-state** command must be configured for this command to produce any output.
show ip msdp peer [*peer-address* | *name*]	Displays detailed information about an MSDP peer.
show ip msdp sa-cache [*group-address* | *source-address* | *group-name* | *source-name*] [*autonomous-system-number*]	Displays (S,G) state learned from MSDP peers.

Command	Purpose
show ip msdp summary	Displays MSDP peer status and SA message counts.

Commands that clear MSDP connections, statistics, and SA cache entries:

Table 2: Commands for Clearing MSDP Connections, Statistics, or SA Cache Entries

Command	Purpose
clear ip msdp peer *peer-address* | *name*	Clears the TCP connection to the specified MSDP peer, resetting all MSDP message counters.
clear ip msdp statistics [*peer-address* | *name*]	Clears statistics counters for one or all the MSDP peers without resetting the sessions.
clear ip msdp sa-cache [*group-address* | *name*]	Clears the SA cache entries for all entries, all sources for a specific group, or all entries for a specific source/group pair.

Configuration Examples for Configuring MSDP

This section provides examples for configuring MSP:

Configuring a Default MSDP Peer: Example

This example shows a partial configuration of Router A and Router C in . Each of these ISPs have more than one customer (like the customer in) who use default peering (no BGP or MBGP). In that case, they might have similar configurations. That is, they accept SAs only from a default peer if the SA is permitted by the corresponding prefix list.

Router A

```
Device(config)#ip msdp default-peer 10.1.1.1
Device(config)#ip msdp default-peer 10.1.1.1 prefix-list site-a
Device(config)#ip prefix-list site-b permit 10.0.0.0/1
```

Router C

```
Device(config)#ip msdp default-peer 10.1.1.1 prefix-list site-a
Device(config)#ip prefix-list site-b permit 10.0.0.0/1
```

Caching Source-Active State: Example

This example shows how to enable the cache state for all sources in 171.69.0.0/16 sending to groups 224.2.0.0/16:

```
Device(config)#ip msdp cache-sa-state 100
```

```
Device(config)#access-list 100 permit ip 171.69.0.0 0.0.255.255 224.2.0.0 0.0.255.255
```

Requesting Source Information from an MSDP Peer: Example

This example shows how to configure the switch to send SA request messages to the MSDP peer at 171.69.1.1:

```
Device(config)#ip msdp sa-request 171.69.1.1
```

Controlling Source Information that Your Switch Originates: Example

This example shows how to configure the switch to filter SA request messages from the MSDP peer at 171.69.2.2. SA request messages from sources on network 192.4.22.0 pass access list 1 and are accepted; all others are ignored.

```
Device(config)#ip msdp filter sa-request 171.69.2.2 list 1
Device(config)#access-list 1 permit 192.4.22.0 0.0.0.255
```

Controlling Source Information that Your Switch Forwards: Example

This example shows how to allow only (S,G) pairs that pass access list 100 to be forwarded in an SA message to the peer named *switch.cisco.com*:

```
Device(config)#ip msdp peer switch.cisco.com connect-source gigabitethernet1/0/1
Device(config)# ip msdp sa-filter out switch.cisco.com list 100
Device(config)#access-list 100 permit ip 171.69.0.0 0.0.255.255 224.20 0 0.0.255.255
```

Controlling Source Information that Your Switch Receives: Example

This example shows how to filter all SA messages from the peer named *switch.cisco.com*:

```
Device(config)#ip msdp peer switch.cisco.com connect-source gigabitethernet1/0/1
Device(config)#ip msdp sa-filter in switch.cisco.com
```

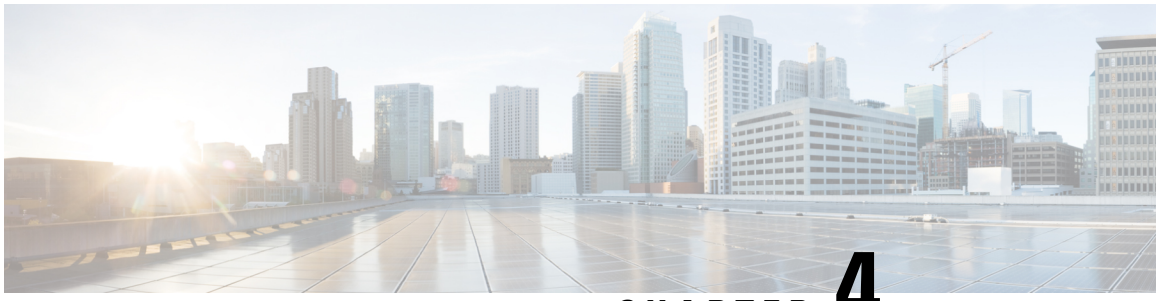
Feature History for Multicast Source Discovery Protocol

This table provides release and related information for the features explained in this module.

These features are available in all the releases subsequent to the one they were introduced in, unless noted otherwise.

Release	Feature	Feature Information
Cisco IOS XE Everest 16.5.1a	MSDP	MSDP allows multicast sources for a group to be known to all rendezvous points (RPs) in different domains.

Use the Cisco Feature Navigator to find information about platform and software image support. To access Cisco Feature Navigator, go to http://www.cisco.com/go/cfn.



CHAPTER 4

Configuring IP Unicast Routing

- Restrictions for IP Unicast Routing, on page 59
- Information About Configuring IP Unicast Routing, on page 59
- Information About IP Routing, on page 60
- Configuration Guidelines for IP Routing, on page 67
- How to Configure IP Addressing, on page 67
- How to Configure IP Unicast Routing, on page 84
- Configuration Example: Enabling IP Routing, on page 85
- Monitoring and Maintaining IP Addressing, on page 86
- Monitoring and Maintaining the IP Network, on page 86
- Feature Information for IP Unicast Routing, on page 87

Restrictions for IP Unicast Routing

- On enabling IP routing, the VLAN configured as SVI will also learn broadcast ARP requests which are not self destined.
- The switch does not support tunnel interfaces for unicast routed traffic.
- The number of routed ports and SVIs that you can configure is 2000. Exceeding the recommended number and volume of features being implemented might impact CPU utilization because of hardware limitations.
- Subnetwork Access Protocol (SNAP) address resolution is not supported on this device.

Information About Configuring IP Unicast Routing

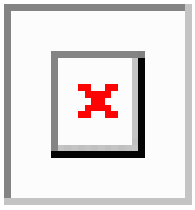
This module describes how to configure IP Version 4 (IPv4) unicast routing on the switch.

Note In addition to IPv4 traffic, you can also enable IP Version 6 (IPv6) unicast routing and configure interfaces to forward IPv6 traffic if the switch or switch stack is running the Network Essentials or Network Advantage license.

Information About IP Routing

In some network environments, VLANs are associated with individual networks or subnetworks. In an IP network, each subnetwork is mapped to an individual VLAN. Configuring VLANs helps control the size of the broadcast domain and keeps local traffic local. However, network devices in different VLANs cannot communicate with one another without a Layer 3 device (router) to route traffic between the VLAN, referred to as inter-VLAN routing. You configure one or more routers to route traffic to the appropriate destination VLAN.

Figure 2: Routing Topology Example

This figure shows a basic routing topology. Switch A is in VLAN 10, and Switch B is in VLAN 20. The router has an interface in each VLAN. 

When Host A in VLAN 10 needs to communicate with Host B in VLAN 10, it sends a packet addressed to that host. Switch A forwards the packet directly to Host B, without sending it to the router.

When Host A sends a packet to Host C in VLAN 20, Switch A forwards the packet to the router, which receives the traffic on the VLAN 10 interface. The router checks the routing table, finds the correct outgoing interface, and forwards the packet on the VLAN 20 interface to Switch B. Switch B receives the packet and forwards it to Host C.

Types of Routing

Routers and Layer 3 switches can route packets in these ways:

- By using default routing
- By using preprogrammed static routes for the traffic

Default routing refers to sending traffic with a destination unknown to the router to a default outlet or destination.

Static unicast routing forwards packets from predetermined ports through a single path into and out of a network. Static routing is secure and uses little bandwidth, but does not automatically respond to changes in the network, such as link failures, and therefore, might result in unreachable destinations. As networks grow, static routing becomes a labor-intensive liability.

Dynamic routing protocols are used by routers to dynamically calculate the best route for forwarding traffic. There are two types of dynamic routing protocols:

- Routers using distance-vector protocols maintain routing tables with distance values of networked resources, and periodically pass these tables to their neighbors. Distance-vector protocols use one or a series of metrics for calculating the best routes. These protocols are easy to configure and use.
- Routers using link-state protocols maintain a complex database of network topology, based on the exchange of link-state advertisements (LSAs) between routers. LSAs are triggered by an event in the network, which speeds up the convergence time or time required to respond to these changes. Link-state

protocols respond quickly to topology changes, but require greater bandwidth and more resources than distance-vector protocols.

Distance-vector protocols supported by the switch are Routing Information Protocol (RIP), which uses a single distance metric (cost) to determine the best path and Border Gateway Protocol (BGP), which adds a path vector mechanism. The switch also supports the Open Shortest Path First (OSPF) link-state protocol and Enhanced IGRP (EIGRP), which adds some link-state routing features to traditional Interior Gateway Routing Protocol (IGRP) to improve efficiency.

Note On a switch or switch stack, the supported protocols are determined by the software running on the active switch. If the active switch is running the Network Essentials license, only default routing, static routing and RIP are supported. All other routing protocols require the Network Advantage license.

IP Routing and Switch Stacks

A switch stack appears to the network as a single switch, regardless of which switch in the stack is connected to a routing peer.

The active switch performs these functions:

- It generates, maintains, and distributes the distributed Cisco Express Forwarding (dCEF) database to all stack members. The routes are programmed on all switches in the stack bases on this database.
- The MAC address of the active switch is used as the router MAC address for the whole stack, and all outside devices use this address to send IP packets to the stack.
- All IP packets that require software forwarding or processing go through the CPU of the active switch.

Stack members perform these functions:

- They act as routing standby switches, ready to take over in case they are elected as the new active switch if the active switch fails.
- They program the routes into hardware.

If a active switch fails, the stack detects that the active switch is down and elects one of the stack members to be the new active switch. During this period, except for a momentary interruption, the hardware continues to forward packets with no active protocols.

However, even though the switch stack maintains the hardware identification after a failure, the routing protocols on the router neighbors might flap during the brief interruption before the active switch restarts. Routing protocols such as OSPF and EIGRP need to recognize neighbor transitions. The router uses two levels of nonstop forwarding (NSF) to detect a switchover, to continue forwarding network traffic, and to recover route information from peer devices:

- NSF-aware routers tolerate neighboring router failures. After the neighbor router restarts, an NSF-aware router supplies information about its state and route adjacencies on request.
- NSF-capable routers support NSF. When they detect a active switch change, they rebuild routing information from NSF-aware or NSF-capable neighbors and do not wait for a restart.

The switch stack supports NSF-capable routing for OSPF and EIGRP.

Upon election, the new active switch performs these functions:

- It starts generating, receiving, and processing routing updates.
- It builds routing tables, generates the CEF database, and distributes it to stack members.
- It uses its MAC address as the router MAC address. To notify its network peers of the new MAC address, it periodically (every few seconds for 5 minutes) sends a gratuitous ARP reply with the new router MAC address.

Note If you configure the persistent MAC address feature on the stack and the active switch changes, the stack MAC address does not change for the configured time period. If the previous active switch rejoins the stack as a member switch during that time period, the stack MAC address remains the MAC address of the previous active switch.

- It attempts to determine the reachability of every proxy ARP entry by sending an ARP request to the proxy ARP IP address and receiving an ARP reply. For each reachable proxy ARP IP address, it generates a gratuitous ARP reply with the new router MAC address. This process is repeated for 5 minutes after a new active switch election.

Note When a active switch is running the Network Advantage license, the stack can run all supported protocols, including Enhanced IGRP (EIGRP), and Border Gateway Protocol (BGP). If the active switch fails and the new elected active switch is running the Network Essentials license, these protocols will no longer run in the stack.

Caution Partitioning of the switch stack into two or more stacks might lead to undesirable behavior in the network.

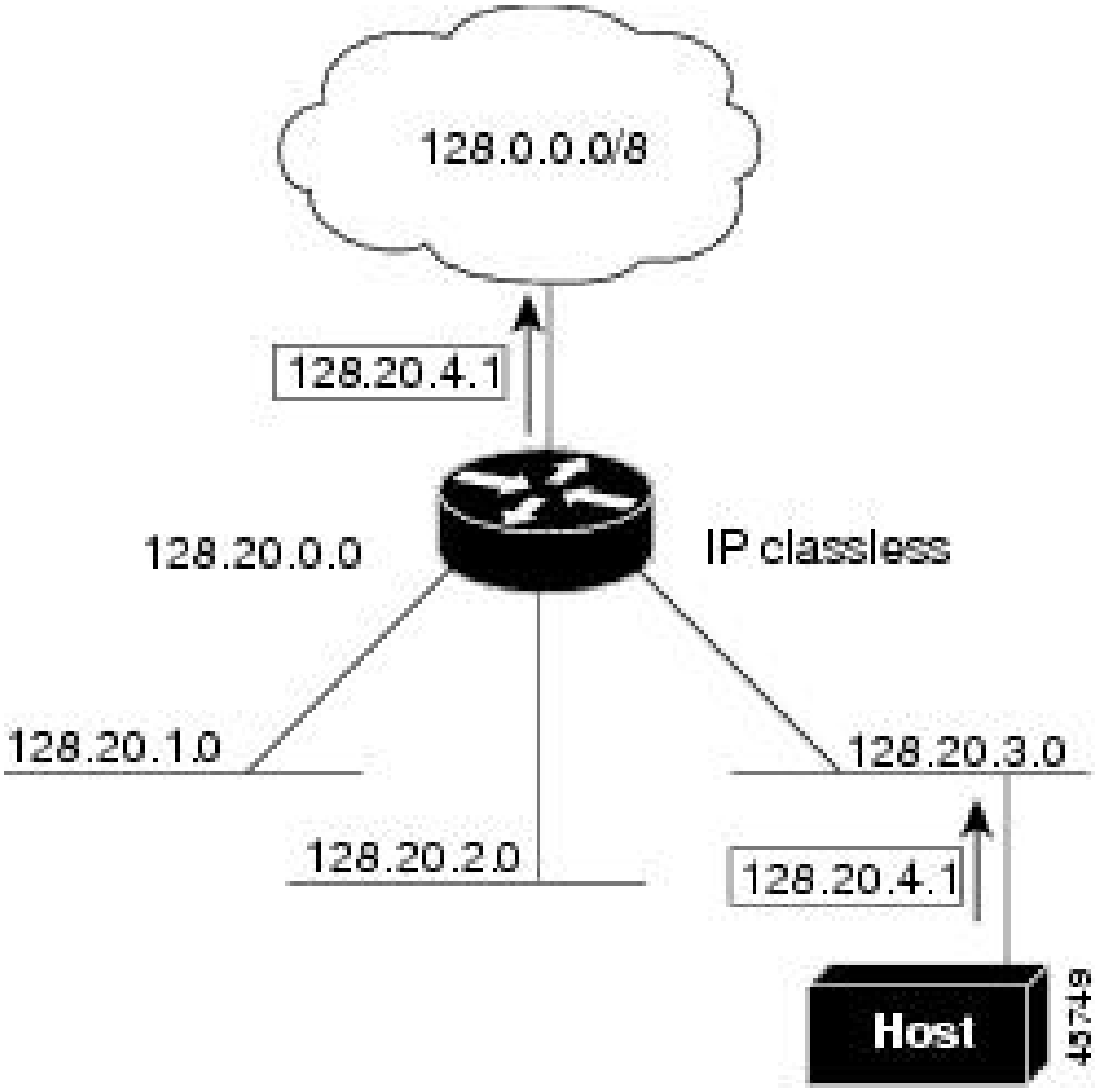
If the switch is reloaded, then all the ports on that switch go down and there is a loss of traffic for the interfaces involved in routing, despite NSF/SSO capability.

Classless Routing

By default, classless routing behavior is enabled on the device when it is configured to route. With classless routing, if a router receives packets for a subnet of a network with no default route, the router forwards the packet to the best supernet route. A supernet consists of contiguous blocks of Class C address spaces used to simulate a single, larger address space and is designed to relieve the pressure on the rapidly depleting Class B address space.

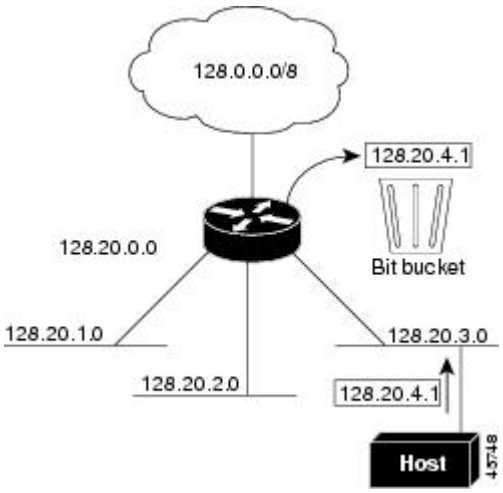
In the figure, classless routing is enabled. When the host sends a packet to 120.20.4.1, instead of discarding the packet, the router forwards it to the best supernet route. If you disable classless routing and a router receives packets destined for a subnet of a network with no network default route, the router discards the packet.

Figure 3: IP Classless Routing



In the figure , the router in network 128.20.0.0 is connected to subnets 128.20.1.0, 128.20.2.0, and 128.20.3.0. If the host sends a packet to 120.20.4.1, because there is no network default route, the router discards the packet.

Figure 4: No IP Classless Routing



To prevent the device from forwarding packets destined for unrecognized subnets to the best supernet route possible, you can disable classless routing behavior.

Address Resolution

You can control interface-specific handling of IP by using address resolution. A device using IP can have both a local address or MAC address, which uniquely defines the device on its local segment or LAN, and a network address, which identifies the network to which the device belongs.

Note In a switch stack, network communication uses a single MAC address and the IP address of the stack.

The local address or MAC address is known as a data link address because it is contained in the data link layer (Layer 2) section of the packet header and is read by data link (Layer 2) devices. To communicate with a device on Ethernet, the software must learn the MAC address of the device. The process of learning the MAC address from an IP address is called *address resolution*. The process of learning the IP address from the MAC address is called *reverse address resolution*.

The device can use these forms of address resolution:

- Address Resolution Protocol (ARP) is used to associate IP address with MAC addresses. Taking an IP address as input, ARP learns the associated MAC address and then stores the IP address/MAC address association in an ARP cache for rapid retrieval. Then the IP datagram is encapsulated in a link-layer frame and sent over the network.
- Proxy ARP helps hosts with no routing tables learn the MAC addresses of hosts on other networks or subnets. If the device (router) receives an ARP request for a host that is not on the same interface as the ARP request sender, and if the router has all of its routes to the host through other interfaces, it generates a proxy ARP packet giving its own local data link address. The host that sent the ARP request then sends its packets to the router, which forwards them to the intended host.

The device also uses the Reverse Address Resolution Protocol (RARP), which functions the same as ARP does, except that the RARP packets request an IP address instead of a local MAC address. Using RARP

requires a RARP server on the same network segment as the router interface. Use the **ip rarp-server** *address* interface configuration command to identify the server.

Proxy ARP

Proxy ARP, the most common method for learning about other routes, enables an Ethernet host with no routing information to communicate with hosts on other networks or subnets. The host assumes that all hosts are on the same local Ethernet and that they can use ARP to learn their MAC addresses. If a device receives an ARP request for a host that is not on the same network as the sender, the device evaluates whether it has the best route to that host. If it does, it sends an ARP reply packet with its own Ethernet MAC address, and the host that sent the request sends the packet to the device, which forwards it to the intended host. Proxy ARP treats all networks as if they are local, and performs ARP requests for every IP address.

ICMP Router Discovery Protocol

Router discovery allows the device to dynamically learn about routes to other networks using ICMP router discovery protocol (IRDP). IRDP allows hosts to locate routers. When operating as a client, the device generates router discovery packets. When operating as a host, the device receives router discovery packets. The device can also listen to Routing Information Protocol (RIP) routing updates and use this information to infer locations of routers. The device does not actually store the routing tables sent by routing devices; it merely keeps track of which systems are sending the data. The advantage of using IRDP is that it allows each router to specify both a priority and the time after which a device is assumed to be down if no further packets are received.

Each device discovered becomes a candidate for the default router, and a new highest-priority router is selected when a higher priority router is discovered, when the current default router is declared down, or when a TCP connection is about to time out because of excessive retransmissions.

IRDP packets are not sent while enabling or disabling IP routing. When interface is shutting down, the last IRDP message do not have a lifetime; it is 0 for all routers.

UDP Broadcast Packets and Protocols

User Datagram Protocol (UDP) is an IP host-to-host layer protocol, as is TCP. UDP provides a low-overhead, connectionless session between two end systems and does not provide for acknowledgment of received datagrams. Network hosts occasionally use UDP broadcasts to find address, configuration, and name information. If such a host is on a network segment that does not include a server, UDP broadcasts are normally not forwarded. You can remedy this situation by configuring an interface on a router to forward certain classes of broadcasts to a helper address. You can use more than one helper address per interface.

You can specify a UDP destination port to control which UDP services are forwarded. You can specify multiple UDP protocols. You can also specify the Network Disk (ND) protocol, which is used by older diskless Sun workstations and the network security protocol SDNS.

By default, both UDP and ND forwarding are enabled if a helper address has been defined for an interface.

Broadcast Packet Handling

After configuring an IP interface address, you can enable routing and configure one or more routing protocols, or you can configure the way the device responds to network broadcasts. A broadcast is a data packet destined for all hosts on a physical network. The device supports two kinds of broadcasting:

- A directed broadcast packet is sent to a specific network or series of networks. A directed broadcast address includes the network or subnet fields.
- A flooded broadcast packet is sent to every network.

Note You can also limit broadcast, unicast, and multicast traffic on Layer 2 interfaces by using the **storm-control** interface configuration command to set traffic suppression levels.

Routers provide some protection from broadcast storms by limiting their extent to the local cable. Bridges (including intelligent bridges), because they are Layer 2 devices, forward broadcasts to all network segments, thus propagating broadcast storms. The best solution to the broadcast storm problem is to use a single broadcast address scheme on a network. In most modern IP implementations, you can set the address to be used as the broadcast address. Many implementations, including the one in the device, support several addressing schemes for forwarding broadcast messages.

IP Broadcast Flooding

You can allow IP broadcasts to be flooded throughout your internetwork in a controlled fashion by using the database created by the bridging STP. Using this feature also prevents loops. To support this capability, bridging must be configured on each interface that is to participate in the flooding. If bridging is not configured on an interface, it still can receive broadcasts. However, the interface never forwards broadcasts it receives, and the router never uses that interface to send broadcasts received on a different interface.

Packets that are forwarded to a single network address using the IP helper-address mechanism can be flooded. Only one copy of the packet is sent on each network segment.

To be considered for flooding, packets must meet these criteria. (Note that these are the same conditions used to consider packet forwarding using IP helper addresses.)

- The packet must be a MAC-level broadcast.
- The packet must be an IP-level broadcast.
- The packet must be a TFTP, DNS, Time, NetBIOS, ND, or BOOTP packet, or a UDP specified by the **ip forward-protocol udp** global configuration command.
- The time-to-live (TTL) value of the packet must be at least two.

A flooded UDP datagram is given the destination address specified with the **ip broadcast-address** interface configuration command on the output interface. The destination address can be set to any address. Thus, the destination address might change as the datagram propagates through the network. The source address is never changed. The TTL value is decremented.

When a flooded UDP datagram is sent out an interface (and the destination address possibly changed), the datagram is handed to the normal IP output routines and is, therefore, subject to access lists, if they are present on the output interface.

In the switch, the majority of packets are forwarded in hardware; most packets do not go through the switch CPU. For those packets that do go to the CPU, you can speed up spanning tree-based UDP flooding by a factor of about four to five times by using turbo-flooding. This feature is supported over Ethernet interfaces configured for ARP encapsulation.

Configuration Guidelines for IP Routing

By default, IP routing is disabled on the device, and you must enable it before routing can take place.

In the following procedures, the specified interface must be one of these Layer 3 interfaces:

- A routed port: a physical port configured as a Layer 3 port by using the **no switchport** interface configuration command.
- A switch virtual interface (SVI): a VLAN interface created by using the **interface vlan** *vlan_id* global configuration command and by default a Layer 3 interface.
- An EtherChannel port channel in Layer 3 mode: a port-channel logical interface created by using the **interface port-channel** *port-channel-number* global configuration command and binding the Ethernet interface into the channel group.

All Layer 3 interfaces on which routing will occur must have IP addresses assigned to them.

Note A Layer 3 switch can have an IP address assigned to each routed port and SVI.

Configuring routing consists of several main procedures:

- To support VLAN interfaces, create and configure VLANs on the switch or switch stack, and assign VLAN membership to Layer 2 interfaces. For more information, see the "Configuring VLANs" chapter.
- Configure Layer 3 interfaces.
- Enable IP routing on the switch.
- Assign IP addresses to the Layer 3 interfaces.
- Enable selected routing protocols on the switch.
- Configure routing protocol parameters (optional).

How to Configure IP Addressing

A required task for configuring IP routing is to assign IP addresses to Layer 3 network interfaces to enable the interfaces and allow communication with the hosts on those interfaces that use IP. The following sections describe how to configure various IP addressing features. Assigning IP addresses to the interface is required; the other procedures are optional.

Default IP Addressing Configuration

Table 3: Default Addressing Configuration

Feature	Default Setting
IP address	None defined.

Feature	Default Setting
ARP	No permanent entries in the Address Resolution Protocol (ARP) cache. Encapsulation: Standard Ethernet-style ARP. Timeout: 14400 seconds (4 hours).
IP broadcast address	255.255.255.255 (all ones).
IP classless routing	Enabled.
IP default gateway	Disabled.
IP directed broadcast	Disabled (all IP directed broadcasts are dropped).
IP domain	Domain list: No domain names defined. Domain lookup: Enabled. Domain name: Enabled.
IP forward-protocol	If a helper address is defined or User Datagram Protocol (UDP) flooding is configured, UDP f is enabled on default ports. Any-local-broadcast: Disabled. Spanning Tree Protocol (STP): Disabled. Turbo-flood: Disabled.
IP helper address	Disabled.
IP host	Disabled.
IRDP	Disabled. Defaults when enabled: • Broadcast IRDP advertisements. • Maximum interval between advertisements: 600 seconds. • Minimum interval between advertisements: 0.75 times max interval • Preference: 0.
IP proxy ARP	Enabled.
IP routing	Disabled.
IP subnet-zero	Disabled.

Assigning IP Addresses to Network Interfaces

An IP address identifies a location to which IP packets can be sent. Some IP addresses are reserved for special uses and cannot be used for host, subnet, or network addresses. RFC 1166, “Internet Numbers,” contains the official description of IP addresses.

An interface can have one primary IP address. A mask identifies the bits that denote the network number in an IP address. When you use the mask to subnet a network, the mask is referred to as a subnet mask. To receive an assigned network number, contact your Internet service provider.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device>enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device#configure terminal`	Enters global configuration mode.
Step 3	**interface** *interface-id* **Example:** `Device(config)#interface gigabitethernet 1/0/1`	Enters interface configuration mode, and specifies the Layer 3 interface to configure.
Step 4	**no switchport** **Example:** `Device(config-if)#no switchport`	Removes the interface from Layer 2 configuration mode (if it is a physical interface).
Step 5	**ip address** *ip-address subnet-mask* **Example:** `Device(config-if)#ip address 10.1.5.1 255.255.255.0`	Configures the IP address and IP subnet mask.
Step 6	**no shutdown** **Example:** `Device(config-if)#no shutdown`	Enables the physical interface.
Step 7	**end** **Example:** `Device(config)#end`	Returns to privileged EXEC mode.
Step 8	**show ip route** **Example:**	Verifies your entries.

	Command or Action	Purpose
	`Device#show ip route`	
Step 9	**show ip interface** [*interface-id*] **Example:** `Device#show ip interface gigabitethernet 1/0/1`	Verifies your entries.
Step 10	**show running-config** **Example:** `Device#show running-config`	Verifies your entries.
Step 11	**copy running-config startup-config** **Example:** `Device#copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

Using Subnet Zero

Subnetting with a subnet address of zero is strongly discouraged because of the problems that can arise if a network and a subnet have the same addresses. For example, if network 131.108.0.0 is subnetted as 255.255.255.0, subnet zero would be written as 131.108.0.0, which is the same as the network address.

You can use the all ones subnet (131.108.255.0) and even though it is discouraged, you can enable the use of subnet zero if you need the entire subnet space for your IP address.

Use the **no ip subnet-zero** global configuration command to restore the default and disable the use of subnet zero.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device>enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device#configure terminal`	Enters global configuration mode.
Step 3	**ip subnet-zero** **Example:**	Enables the use of subnet zero for interface addresses and routing updates.

	Command or Action	Purpose
	`Device(config)#ip subnet-zero`	
Step 4	**end** **Example:** `Device(config)#end`	Returns to privileged EXEC mode.
Step 5	**show running-config** **Example:** `Device#show running-config`	Verifies your entries.
Step 6	**copy running-config startup-config** **Example:** `Device#copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

Disabling Classless Routing

To prevent the device from forwarding packets destined for unrecognized subnets to the best supernet route possible, you can disable classless routing behavior.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device>enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device#configure terminal`	Enters global configuration mode.
Step 3	**no ip classless** **Example:** `Device(config)#no ip classless`	Disables classless routing behavior.
Step 4	**end** **Example:**	Returns to privileged EXEC mode.

	Command or Action	Purpose
	`Device(config)#end`	
Step 5	**show running-config** **Example:** `Device#show running-config`	Verifies your entries.
Step 6	**copy running-config startup-config** **Example:** `Device#copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

Configuring Address Resolution Methods

You can perform the following tasks to configure address resolution.

Defining a Static ARP Cache

ARP and other address resolution protocols provide dynamic mapping between IP addresses and MAC addresses. Because most hosts support dynamic address resolution, you usually do not need to specify static ARP cache entries. If you must define a static ARP cache entry, you can do so globally, which installs a permanent entry in the ARP cache that the device uses to translate IP addresses into MAC addresses. Optionally, you can also specify that the device responds to ARP requests as if it were the owner of the specified IP address. If you do not want the ARP entry to be permanent, you can specify a timeout period for the ARP entry.

To define a static arp cache, perform this procedure:

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device>enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device#configure terminal`	Enters global configuration mode.

	Command or Action	Purpose
Step 3	**arp** *ip-address hardware-address type* **Example:** `Device(config)#ip 10.1.5.1 c2f3.220a.12f4 arpa`	Associates an IP address with a MAC (hardware) address in the ARP cache, and specifies encapsulation type as one of these: • **arpa**—ARP encapsulation for Ethernet interfaces • **sap**—HP's ARP type
Step 4	**arp** *ip-address hardware-address type* [**alias**] **Example:** `Device(config)#ip 10.1.5.3 d7f3.220d.12f5 arpa alias`	(Optional) Specifies that the switch responds to ARP requests as if it were the owner of the specified IP address.
Step 5	**interface** *interface-id* **Example:** `Device(config)#interface gigabitethernet 1/0/1`	Enters interface configuration mode, and specifies the interface to configure.
Step 6	**arp timeout** *seconds* **Example:** `Device(config-if)#arp timeout 20000`	(Optional) Sets the length of time an ARP cache entry stays in the cache. The recommended value of ARP timeout is 4 hours which is also the default setting. However, if your network experiences regular updates to ARP cache entries, consider changing the timeout. Note that decreasing the ARP timeout can result in increased network traffic. It is not recommended to set the ARP timeout to 60 seconds or less.
Step 7	**end** **Example:** `Device(config)#end`	Returns to privileged EXEC mode.
Step 8	**show interfaces** [*interface-id*] **Example:** `Device#show interfaces gigabitethernet 1/0/1`	Verifies the type of ARP and the timeout value that is used on all interfaces or a specific interface.
Step 9	**show arp** **Example:** `Device#show arp`	Views the contents of the ARP cache.
Step 10	**show ip arp** **Example:** `Device#show ip arp`	Views the contents of the ARP cache.

	Command or Action	Purpose
Step 11	**copy running-config startup-config** **Example:** `Device#copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

Setting ARP Encapsulation

By default, Ethernet ARP encapsulation (represented by the **arpa** keyword) is enabled on an IP interface.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device>enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device#configure terminal`	Enters global configuration mode.
Step 3	**interface** *interface-id* **Example:** `Device(config)#interface gigabitethernet 1/0/2`	Enters interface configuration mode, and specifies the Layer 3 interface to configure.
Step 4	**arp arpa** **Example:** `Device(config-if)#arp arpa`	Specifies the ARP encapsulation method. Use the **no arp arpa** command to disable ARP encapsulation method.
Step 5	**end** **Example:** `Device(config)#end`	Returns to privileged EXEC mode.
Step 6	**show interfaces** [*interface-id*] **Example:** `Device#show interfaces`	Verifies ARP encapsulation configuration on all interfaces or the specified interface.

	Command or Action	Purpose
Step 7	**copy running-config startup-config** **Example:** `Device#copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

Enabling Proxy ARP

By default, the device uses proxy ARP to help hosts learn MAC addresses of hosts on other networks or subnets.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device>enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device#configure terminal`	Enters global configuration mode.
Step 3	**interface** *interface-id* **Example:** `Device(config)#interface gigabitethernet 1/0/2`	Enters interface configuration mode, and specifies the Layer 3 interface to configure.
Step 4	**ip proxy-arp** **Example:** `Device(config-if)#ip proxy-arp`	Enables proxy ARP on the interface.
Step 5	**end** **Example:** `Device(config)#end`	Returns to privileged EXEC mode.
Step 6	**show ip interface** [*interface-id*] **Example:** `Device#show ip interface gigabitethernet 1/0/2`	Verifies the configuration on the interface or all interfaces.

	Command or Action	Purpose
Step 7	**copy running-config startup-config** **Example:** `Device#copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

Routing Assistance When IP Routing is Disabled

These mechanisms allow the device to learn about routes to other networks when it does not have IP routing enabled:

- Proxy ARP
- Default Gateway
- ICMP Router Discovery Protocol (IRDP)

Proxy ARP

Proxy ARP is enabled by default. To enable it after it has been disabled, see the "Enabling Proxy ARP" section. Proxy ARP works as long as other routers support it.

Default Gateway

Another method for locating routes is to define a default router or default gateway. All non-local packets are sent to this router, which either routes them appropriately or sends an IP Control Message Protocol (ICMP) redirect message back, defining which local router the host should use. The device caches the redirect messages and forwards each packet as efficiently as possible. A limitation of this method is that there is no means of detecting when the default router has gone down or is unavailable.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device>enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device#configure terminal`	Enters global configuration mode.
Step 3	**ip default-gateway** *ip-address* **Example:**	Sets up a default gateway (router).

	Command or Action	Purpose
	`Device(config)#ip default gateway 10.1.5.1`	
Step 4	**end** **Example:** `Device(config)#end`	Returns to privileged EXEC mode.
Step 5	**show ip redirects** **Example:** `Device#show ip redirects`	Displays the address of the default gateway router to verify the setting.
Step 6	**copy running-config startup-config** **Example:** `Device#copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

ICMP Router Discovery Protocol (IRDP)

The only required task for IRDP routing on an interface is to enable IRDP processing on that interface. When enabled, the default parameters apply.

You can optionally change any of these parameters. If you change the **maxadvertinterval** value, the **holdtime** and **minadvertinterval** values also change, so it is important to first change the **maxadvertinterval** value, before manually changing either the **holdtime** or **minadvertinterval** values.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device>enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device#configure terminal`	Enters global configuration mode.
Step 3	**interface** *interface-id* **Example:** `Device(config)#interface gigabitethernet 1/0/1`	Enters interface configuration mode, and specifies the Layer 3 interface to configure.

	Command or Action	Purpose
Step 4	**ip irdp** **Example:** `Device(config-if)#ip irdp`	Enables IRDP processing on the interface.
Step 5	**ip irdp multicast** **Example:** `Device(config-if)#ip irdp multicast`	(Optional) Sends IRDP advertisements to the multicast address (224.0.0.1) instead of IP broadcasts. **Note** This command allows for compatibility with Sun Microsystems Solaris, which requires IRDP packets to be sent out as multicasts. Many implementations cannot receive these multicasts; ensure end-host ability before using this command.
Step 6	**ip irdp holdtime** *seconds* **Example:** `Device(config-if)#ip irdp holdtime 1000`	(Optional) Sets the IRDP period for which advertisements are valid. The default is three times the **maxadvertinterval** value. It must be greater than **maxadvertinterval** and cannot be greater than 9000 seconds. If you change the **maxadvertinterval** value, this value also changes.
Step 7	**ip irdp maxadvertinterval** *seconds* **Example:** `Device(config-if)#ip irdp maxadvertinterval 650`	(Optional) Sets the IRDP maximum interval between advertisements. The default is 600 seconds.
Step 8	**ip irdp minadvertinterval** *seconds* **Example:** `Device(config-if)#ip irdp minadvertinterval 500`	(Optional) Sets the IRDP minimum interval between advertisements. The default is 0.75 times the **maxadvertinterval**. If you change the **maxadvertinterval**, this value changes to the new default (0.75 of **maxadvertinterval**).
Step 9	**ip irdp preference** *number* **Example:** `Device(config-if)#ip irdp preference 2`	(Optional) Sets a device IRDP preference level. The allowed range is –231 to 231. The default is 0. A higher value increases the router preference level.
Step 10	**ip irdp address** *address* [*number*] **Example:** `Device(config-if)#ip irdp address 10.1.10.10`	(Optional) Specifies an IRDP address and preference to proxy-advertise.
Step 11	**end** **Example:** `Device(config)#end`	Returns to privileged EXEC mode.

	Command or Action	Purpose
Step 12	**show ip irdp** **Example:** `Device#show ip irdp`	Verifies settings by displaying IRDP values.
Step 13	**copy running-config startup-config** **Example:** `Device#copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

Configuring Broadcast Packet Handling

Perform the tasks in these sections to enable these schemes:

- Enabling Directed Broadcast-to-Physical Broadcast Translation
- Forwarding UDP Broadcast Packets and Protocols
- Establishing an IP Broadcast Address
- Flooding IP Broadcasts

Enabling Directed Broadcast-to-Physical Broadcast Translation

By default, IP directed broadcasts are dropped; they are not forwarded. Dropping IP-directed broadcasts makes routers less susceptible to denial-of-service attacks.

You can enable forwarding of IP-directed broadcasts on an interface where the broadcast becomes a physical (MAC-layer) broadcast. Only those protocols configured by using the **ip forward-protocol** global configuration command are forwarded.

You can specify an access list to control which broadcasts are forwarded. When an access list is specified, only those IP packets permitted by the access list are eligible to be translated from directed broadcasts to physical broadcasts. For more information on access lists, see the “Configuring ACLs" chapter in the *Security Configuration Guide*.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device>enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**configure terminal** **Example:**	Enters global configuration mode.

	Command or Action	Purpose
	`Device#configure terminal`	
Step 3	**interface** *interface-id* **Example:** `Device(config)#interface gigabitethernet 1/0/2`	Enters interface configuration mode, and specifies the interface to configure.
Step 4	**ip directed-broadcast** [*access-list-number*] **Example:** `Device(config-if)#ip directed-broadcast 103`	Enables directed broadcast-to-physical broadcast translation on the interface. You can include an access list to control which broadcasts are forwarded. When an access list, only IP packets permitted by the access list can be translated.
Step 5	**exit** **Example:** `Device(config-if)#exit`	Returns to global configuration mode.
Step 6	**ip forward-protocol** {**udp** [*port*] | **nd** | **sdns**} **Example:** `Device(config)#ip forward-protocol nd`	Specifies which protocols and ports the router forwards when forwarding broadcast packets. • **udp**—Forward UPD datagrams. port: (Optional) Destination port that controls which UDP services are forwarded. • **nd**—Forward ND datagrams. • **sdns**—Forward SDNS datagrams
Step 7	**end** **Example:** `Device(config)#end`	Returns to privileged EXEC mode.
Step 8	**show ip interface** [*interface-id*] **Example:** `Device#show ip interface`	Verifies the configuration on the interface or all interfaces
Step 9	**show running-config** **Example:** `Device#show running-config`	Verifies your entries.
Step 10	**copy running-config startup-config** **Example:**	(Optional) Saves your entries in the configuration file.

	Command or Action	Purpose
	`Device#copy running-config startup-config`	

Forwarding UDP Broadcast Packets and Protocols

If you do not specify any UDP ports when you configure the forwarding of UDP broadcasts, you are configuring the router to act as a BOOTP forwarding agent. BOOTP packets carry DHCP information.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device>enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device#configure terminal`	Enters global configuration mode.
Step 3	**interface** *interface-id* **Example:** `Device(config)#interface gigabitethernet 1/0/1`	Enters interface configuration mode, and specifies the Layer 3 interface to configure.
Step 4	**ip helper-address** *address* **Example:** `Device(config-if)#ip helper address 10.1.10.1`	Enables forwarding and specifies the destination address for forwarding UDP broadcast packets, including BOOTP.
Step 5	**exit** **Example:** `Device(config-if)#exit`	Returns to global configuration mode.
Step 6	**ip forward-protocol** {**udp** [*port*] | **nd** | **sdns**} **Example:** `Device(config)#ip forward-protocol sdns`	Specifies which protocols the router forwards when forwarding broadcast packets.
Step 7	**end** **Example:**	Returns to privileged EXEC mode.

	Command or Action	Purpose
	`Device(config)#end`	
Step 8	**show ip interface** [*interface-id*] **Example:** `Device#show ip interface gigabitethernet 1/0/1`	Verifies the configuration on the interface or all interfaces.
Step 9	**show running-config** **Example:** `Device#show running-config`	Verifies your entries.
Step 10	**copy running-config startup-config** **Example:** `Device#copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

Establishing an IP Broadcast Address

The most popular IP broadcast address (and the default) is an address consisting of all ones (255.255.255.255). However, the switch can be configured to generate any form of IP broadcast address.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device>enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device#configure terminal`	Enters global configuration mode.
Step 3	**interface** *interface-id* **Example:** `Device(config)#interface gigabitethernet 1/0/1`	Enters interface configuration mode, and specifies the interface to configure.
Step 4	**ip broadcast-address** *ip-address* **Example:**	Enters a broadcast address different from the default, for example 128.1.255.255.

	Command or Action	Purpose
	`Device(config-if)#ip broadcast-address 128.1.255.255`	
Step 5	**end** **Example:** `Device(config)#end`	Returns to privileged EXEC mode.
Step 6	**show ip interface** [*interface-id*] **Example:** `Device#show ip interface`	Verifies the broadcast address on the interface or all interfaces.
Step 7	**copy running-config startup-config** **Example:** `Device#copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

Flooding IP Broadcasts

To configure IP broadcasts flooding, perform this procedure.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device>enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device#configure terminal`	Enters global configuration mode.
Step 3	**ip forward-protocol spanning-tree** **Example:** `Device(config)#ip forward-protocol spanning-tree`	Uses the bridging spanning-tree database to flood UDP datagrams.
Step 4	**ip forward-protocol turbo-flood** **Example:**	Uses the spanning-tree database to speed up flooding of UDP datagrams.

	Command or Action	Purpose
	`Device(config)#ip forward-protocol turbo-flood`	
Step 5	**end** **Example:** `Device(config)#end`	Returns to privileged EXEC mode.
Step 6	**show running-config** **Example:** `Device#show running-config`	Verifies your entries.
Step 7	**copy running-config startup-config** **Example:** `Device#copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

How to Configure IP Unicast Routing

The following sections provide configuration infornation about IP unicast routing.

Enabling IP Unicast Routing

By default, the device is in Layer 2 switching mode and IP routing is disabled. To use the Layer 3 capabilities of the device, you must enable IP routing.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device>enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device#configure terminal`	Enters global configuration mode.

	Command or Action	Purpose
Step 3	**ip routing** **Example:** `Device(config)#ip routing`	Enables IP routing.
Step 4	**end** **Example:** `Device(config)#end`	Returns to privileged EXEC mode.
Step 5	**show running-config** **Example:** `Device#show running-config`	Verifies your entries.
Step 6	**copy running-config startup-config** **Example:** `Device#copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

What to Do Next

You can now set up parameters for the selected routing protocols as described in these sections:

- RIP
- OSPF,
- EIGRP
- BGP
- Unicast Reverse Path Forwarding
- Protocol-Independent Features (optional)

Configuration Example: Enabling IP Routing

This example shows how to enable IP routingusing RIP as the routing protocol:

```
Device#configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
Device(config)#ip routing
Device(config-router)#end
```

Monitoring and Maintaining IP Addressing

When the contents of a particular cache, table, or database have become or are suspected to be invalid, you can remove all its contents by using the **clear** privileged EXEC commands. The Table lists the commands for clearing contents.

Table 4: Commands to Clear Caches, Tables, and Databases

Command	Purpose
clear arp-cache	Clears the IP ARP cache and the fast-switching cache.
clear host {*name* | *****}	Removes one or all entries from the hostname and the address cac
clear ip route {**network** [*mask*] | *****}	Removes one or more routes from the IP routing table.

You can display specific statistics, such as the contents of IP routing tables, caches, and databases; the reachability of nodes; and the routing path that packets are taking through the network. The Table lists the privileged EXEC commands for displaying IP statistics.

Table 5: Commands to Display Caches, Tables, and Databases

Command	Purpose
show arp	Displays the entries in the ARP table.
show hosts	Displays the default domain name, style of lookup service, name server the cached list of hostnames and addresses.
show ip aliases	Displays IP addresses mapped to TCP ports (aliases).
show ip arp	Displays the IP ARP cache.
show ip interface [*interface-id*]	Displays the IP status of interfaces.
show ip irdp	Displays IRDP values.
show ip masks *address*	Displays the masks used for network addresses and the number of subn each mask.
show ip redirects	Displays the address of a default gateway.
show ip route [*address* [*mask*]] | [*protocol*]	Displays the current state of the routing table.
show ip route summary	Displays the current state of the routing table in summary form.

Monitoring and Maintaining the IP Network

You can remove all contents of a particular cache, table, or database. You can also display specific statistics.

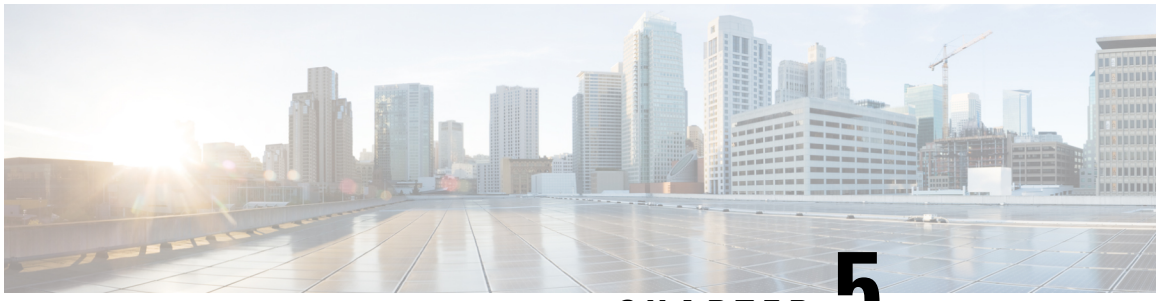
Table 6: Command to Clear IP Routes or Display Route Status

Command	Purpose
show ip route summary	Displays the current state of the routing table in summary

Feature Information for IP Unicast Routing

Table 7: Feature Information for IP Unicast Routing

Release	Feature Information
Cisco IOS XE Everest 16.5.1a	This feature was introduced



CHAPTER 5

Configuring IPv6 Unicast Routing

- Information About Configuring IPv6 Unicast Routing, on page 89
- How to Configure IPv6 Unicast Routing, on page 93
- Configuration Examples for IPv6 Unicast Routing, on page 106
- Additional References, on page 108
- Feature History for IPv6 Unicast Routing, on page 109

Information About Configuring IPv6 Unicast Routing

This chapter describes how to configure IPv6 unicast routing on the switch.

Note To use all IPv6 features in this chapter, the switch or active switch must be running the Network Advantage license. Switches running the Network Essentials license support IPv6 static routing and RIP for IPv6. Switches running the Network Advantage license support OSPF, EIGRP and BGP for IPv6.

Understanding IPv6

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

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.

Static Routes for IPv6

Static routes are manually configured and define an explicit route between two networking devices. Static routes are useful for smaller networks with only one path to an outside network or to provide security for certain types of traffic in a larger network.

Configuring Static Routing for IPv6 (CLI)

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.

Path MTU Discovery for IPv6 Unicast

The switch supports advertising the system maximum transmission unit (MTU) to IPv6 nodes and path MTU discovery. Path MTU discovery allows a host to dynamically discover and adjust to differences in the MTU size of every link along a given data path. In IPv6, if a link along the path is not large enough to accommodate the packet size, the source of the packet handles the fragmentation.

ICMPv6

The Internet Control Message Protocol (ICMP) in IPv6 generates error messages, such as ICMP destination unreachable messages, to report errors during processing and other diagnostic functions. In IPv6, ICMP packets are also used in the neighbor discovery protocol and path MTU discovery.

Neighbor Discovery

The switch supports NDP for IPv6, a protocol running on top of ICMPv6, and static neighbor entries for IPv6 stations that do not support NDP. The IPv6 neighbor discovery process uses ICMP messages and solicited-node multicast addresses to determine the link-layer address of a neighbor on the same network (local link), to verify the reachability of the neighbor, and to keep track of neighboring routers.

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.

IPv6 Router Advertisement Options for DNS Configuration

Most of the internet services are identified by a Domain Name Server (DNS) name. IPv6 Router Advertisement (RA) provides the following two options to allow IPv6 hosts to perform automatic DNS configuration:

- Recursive DNS Server (RDNSS)
- DNS Search List (DNSSL)

RDNSS contains the address of recursive DNS servers that help in DNS name resolution in IPv6 hosts. DNS Search List is a list of DNS suffix domain names used by IPv6 hosts when they perform DNS query searches.

For more information on RA options for DNS configuration, refer IETF RFC 6106.

For configuring DNSSL, see the *Configuring DNS Search List Using IPv6 Router Advertisement Options* section of the *IP Addressing Services Configuration Guide*.

Default Router Preference

The switch supports IPv6 default router preference (DRP), an extension in router advertisement messages. DRP improves the ability of a host to select an appropriate router, especially when the host is multihomed and the routers are on different links. The switch does not support the Route Information Option in RFC 4191.

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. By using DRP, you can configure an IPv6 host to prefer one router over another, provided both are reachable or probably reachable.

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.

Policy-Based Routing for IPv6

Policy-based routing (PBR) gives you a flexible means of routing packets by allowing you to configure a defined policy for traffic flows, which lessens reliance on routes derived from routing protocols. Therefore, PBR gives you more control over routing by extending and complementing the existing mechanisms provided by routing protocols. PBR allows you to set the IPv6 precedence. For a simple policy, you can use any one of these tasks; for a complex policy, you can use all of them. It also allows you to specify a path for certain traffic, such as priority traffic over a high-cost link.

PBR for IPv6 may be applied to both forwarded and originated IPv6 packets. For forwarded packets, PBR for IPv6 will be implemented as an IPv6 input interface feature, supported in the following 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 the following 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; you might need to route them to allow a specific quality of service (QoS) through 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.

Unsupported IPv6 Unicast Routing Features

The switch does not support these IPv6 features:

- IPv6 packets 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

Because IPv6 is implemented in switch hardware, some limitations occure due to the IPv6 compressed addresses in the hardware memory. This hardware limitation result in some loss of functionality and limits some features. For example, the switch cannot apply QoS classification on source-routed IPv6 packets in hardware.

IPv6 and Switch Stacks

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. They 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 and distributes them to the member switches. While the new active switch is being elected and is resetting, the switch stack does not forward IPv6 packets. The stack MAC address changes, which also changes 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.

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 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

Table 8: Default IPv6 Configuration

Feature	Default Setting
SDM template	Default is advance template
IPv6 routing	Disabled globally and on all interfaces
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

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

Configuring IPv6 Addressing and Enabling IPv6 Routing

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

Note IPv6 routing is not enabled by default and needs to be enabled using the **ipv6 unicast-routing** command.

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. The configured interface automatically joins these required multicast groups for that link:

- solicited-node multicast group FF02:0:0:0:0:1:ff00::/104 for each unicast address assigned to the interface (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.

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:

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**sdm prefer access** **Example:** `Device(config)# sdm prefer access`	Sets the switch to the access template.
Step 4	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode.

	Command or Action	Purpose
Step 5	**reload** **Example:** `Device# reload`	Reloads the operating system.
Step 6	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode after the switch reloads.
Step 7	**interface** *interface-id* **Example:** `Device(config)# interface gigabitethernet 1/0/1`	Enters interface configuration mode, and specifies the Layer 3 interface to configure. The interface can be a physical interface, a switch virtual interface (SVI), or a Layer 3 EtherChannel.
Step 8	**no switchport** **Example:** `Device(config-if)# no switchport`	Removes the interface from Layer 2 configuration mode (if it is a physical interface).

Step 9	Use one of the following:
	• **ipv6 address** *ipv6-prefix/prefix length* **eui-64**
	• **ipv6 address** *ipv6-address/prefix length*
	• **ipv6 address** *ipv6-address* **link-local**
	• **ipv6 enable**
	• **ipv6 address** *WORD*
	• **ipv6 address** *autoconfig*
	• **ipv6 address** *dhcp*
	Example:
	`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`
		• 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 last 64 bits are automatically computed from the switch MAC address. This enables IPv6 processing on the interface.
		• Manually configures an IPv6 address on the interface.
		• 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.
		• 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.

Step 10	**exit** **Example:**	Returns to global configuration mode.

	Command or Action	Purpose
	`Device(config-if)# exit`	
Step 11	**ip routing** **Example:** `Device(config)# ip routing`	Enables IP routing on the switch.
Step 12	**ipv6 unicast-routing** **Example:** `Device(config)# ipv6 unicast-routing`	Enables forwarding of IPv6 unicast data packets.
Step 13	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode.
Step 14	**show ipv6 interface** *interface-id* **Example:** `Device# show ipv6 interface gigabitethernet 1/0/1`	Verifies your entries.
Step 15	**copy running-config startup-config** **Example:** `Device# copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

Configuring IPv4 and IPv6 Protocol Stacks

Beginning in privileged EXEC mode, follow these steps to configure a Layer 3 interface to support both IPv4 and IPv6 and to enable IPv6 routing.

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

Procedure

	Command or Action	Purpose
Step 1	**enable**	Enables privileged EXEC mode.

	Command or Action	Purpose
	Example: `Device> enable`	Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**ip routing** **Example:** `Device(config)# ip routing`	Enables routing on the switch.
Step 4	**ipv6 unicast-routing** **Example:** `Device(config)# ipv6 unicast-routing`	Enables forwarding of IPv6 data packets on the switch.
Step 5	**interface** *interface-id* **Example:** `Device(config)# interface gigabitethernet 1/0/1`	Enters interface configuration mode, and specifies the Layer 3 interface to configure.
Step 6	**no switchport** **Example:** `Device(config-if)# no switchport`	Removes the interface from Layer 2 configuration mode (if it is a physical interface).
Step 7	**ip address** *ip-address mask* [**secondary**] **Example:** `Device(config-if)# ip address 10.1.2.3 255.255.255`	Specifies a primary or secondary IPv4 address for the interface.
Step 8	Use one of the following: • **ipv6 address** *ipv6-prefix/prefix length* **eui-64** • **ipv6 address** *ipv6-address/prefix length* • **ipv6 address** *ipv6-address* **link-local** • **ipv6 enable** • **ipv6 address***WORD* • **ipv6 address***autoconfig* • **ipv6 address***dhcp*	• 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.
Step 9	**end** **Example:**	Returns to privileged EXEC mode.

	Command or Action	Purpose
	`Device(config)# end`	
Step 10	Use one of the following: • **show interface** *interface-id* • **show ip interface** *interface-id* • **show ipv6 interface** *interface-id*	Verifies your entries.
Step 11	**copy running-config startup-config** **Example:** `Device# copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

Configuring Recursive DNS Server (RDNSS)

You can configure up to eight DNS servers to advertise with Router Advertisement. You can also remove one or more DNS servers from the advertising list by using the **no** form of the command.

Before you begin

Ensure that you are in the correct VDC (or use the **switchto vdc** command).

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter the password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface ethernet** *number* **Example:** `Device(config)# interface ethernet 3/3`	Enters interface configuration mode.
Step 4	**ipv6 nd ra dns server** *ipv6-addr* [*rdnss-life* | **infinite**] **sequence** *sequence-num* **Example:** `Device(config-if)# ipv6 nd ra dns server 1::1 1000 sequence 0`	Configures the recursive DNS server. You can specify the life time and the sequence of the server.
Step 5	**show ipv6 nd ra dns server** [**interface** *interface*] **Example:** `Device(config-if)# show ipv6 nd ra dns server`	(Optional) Displays the configured RDNSS list.

	Command or Action	Purpose
Step 6	**ipv6 nd ra dns server suppress** **Example:** `Device(config-if)# ipv6 nd ra dns server suppress`	(Optional) Disables the configured server list.

Configuring Default Router Preference

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.

Beginning in privileged EXEC mode, follow these steps to configure a DRP for a router on an interface.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface** *interface-id* **Example:** `Device(config)# interface gigabitethernet 1/0/1`	Enters interface configuration mode and identifies the Layer 3 interface on which you want to specify the DRP.
Step 4	**ipv6 nd router-preference** {**high** | **medium** | **low**} **Example:** `Device(config-if)# ipv6 nd router-preference medium`	Specifies a DRP for the router on the switch interface.
Step 5	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode.
Step 6	**show ipv6 interface** **Example:** `Device# show ipv6 interface`	Verifies the configuration.

	Command or Action	Purpose
Step 7	**copy running-config startup-config** **Example:** `Device# copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

Configuring IPv6 ICMP Rate Limiting

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

To change the ICMP rate-limiting parameters, perform this procedure:

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**ipv6 icmp error-interval** *interval* [*bucketsize*] **Example:** `Device(config)# ipv6 icmp error-interval 50 20`	Configures the interval and bucket size for IPv6 ICMP error messages: • *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.
Step 4	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode.
Step 5	**show ipv6 interface** [*interface-id*] **Example:** `Device# show ipv6 interface gigabitethernet0/1`	Verifies your entries.
Step 6	**copy running-config startup-config** **Example:** `Device# copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

Configuring Cisco Express Forwarding and distributed Cisco Express Forwarding for IPv6

Cisco Express Forwarding is a Layer 3 IP switching technology to improve network performance. Cisco Express Forwarding implements an advanced IP look-up and forwarding algorithm to deliver maximum Layer 3 switching performance. It is less CPU-intensive than fast-switching route-caching, allowing more CPU processing power to be dedicated to packet forwarding. In a switch stack, the hardware uses distributed Cisco Express Forwarding in the stack. 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. IPv6 Cisco Express Forwarding and distributed Cisco Express Forwarding cannot disabled through configuration. You can verify the IPv6 state by entering the **show ipv6 cef** command in privileged EXEC mode.

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** command in interface configuration mode.

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

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.

To configure static IPv6 routing, perform this procedure:

Before you begin

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

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**ipv6 route** *ipv6-prefix/prefix length* {*ipv6-address* | *interface-id* [*ipv6-address*]} [*administrative distance*]	Configures a static IPv6 route.

	Command or Action	Purpose
	Example: `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.
Step 4	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode.
Step 5	Use one of the following: • **show ipv6 static** [*ipv6-address* | *ipv6-prefix/prefix length*] [**interface** *interface-id*] [**detail**]][**recursive**] [**detail**] • **show ipv6 route static** [*updated*]	Verifies your entries by displaying the contents of the IPv6 routing table. • **interface** *interface-id*—(Optional) Displays only those static routes with the specified interface as an egress interface.

	Command or Action	Purpose

	Example:
	`Device# show ipv6 static 2001:0DB8::/32 interface gigabitethernet2/0/1`
	or
	`Device# show ipv6 route static`
		• **recursive**—(Optional) 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**—(Optional) 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.

Step 6	**copy running-config startup-config** **Example:** `Device# copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

Enabling IPv6 PBR on an Interface

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. Then you associate the route map on 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, selects a routing table, looks up the destination address, and then routes the packet.

To enable PBR for IPv6, perform this procedure:

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**route-map** *map-tag* [**permit** | **deny**] [*sequence-number*] **Example:** `Device(config)# route-map rip-to-ospf permit`	Defines the conditions for redistributing routes from one routing protocol into another, or enables policy routing, and enters route-map configuration mode.

	Command or Action	Purpose

Step 4	Do one of the following:
	• **match length** *minimum-length maximum-length*
	• **match ipv6 address** {**prefix-list** *prefix-list-name* | *access-list-name*}
	Example:
	`Device(config-route-map)# match length 3 200`
	Example:
	`Device(config-route-map)# match ipv6 address marketing`
		Specifies the match criteria.
		• You can specify any or all of the following:
		• Matches the Level 3 length of the packet.
		• Matches a specified IPv6 access list.
		• If you do not specify a **match** command, the route map applies to all packets.

Step 5	Do one of the following:
	• **set ipv6 next-hop** *global-ipv6-address* [*global-ipv6-address...*]
	• **set interface** *type number* [*...type number*]
	• **set ipv6 default next-hop** *global-ipv6-address* [*global-ipv6-address...*]
	• **set vrf** *vrf-name*
	Example:
	`Device(config-route-map)# set ipv6 next-hop 2001:DB8:2003:1::95`
	Example:
	`Device(config-route-map)# set ipv6 default next-hop 2001:DB8:2003:1::95`
	Example:
	`Device(config-route-map)# set vrf vrfname`
		Specifies the action or actions to take on the packets that match the criteria.
		• You can specify any or all of the following:
		• 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.

Step 6	**exit** **Example:** `Device(config-route-map)# exit`	Exits route-map configuration mode and returns to global configuration mode.
Step 7	**interface** *type number* **Example:** `Device(config)# interface FastEthernet 1/0`	Specifies an interface type and number, and places the router in interface configuration mode.
Step 8	**ipv6 policy route-map** *route-map-name* **Example:** `Device(config-if)# ipv6 policy-route-map interactive`	Identifies a route map to use for IPv6 PBR on an interface.
Step 9	**end** **Example:** `Device(config-if)# end`	Exits interface configuration mode and returns to privileged EXEC mode.

Enabling Local PBR for IPv6

Packets that are generated by the device are not normally policy routed. Perform this task to enable local IPv6 policy-based routing (PBR) for such packets, indicating which route map the device should use.

To enable Local PBR for IPv6, perform this procedure:

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**ipv6 local policy route-map** *route-map-name* **Example:** `Device(config)# ipv6 local policy route-map pbr-src-90`	Configures IPv6 PBR for packets generated by the device.
Step 4	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode.

Displaying IPv6

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

Table 9: 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.

Command	Purpose
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

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

Example: Configuring IPv4 and IPv6 Protocol Stacks

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

```
Device> enable
Device# configure terminal
Device(config)# ip routing
Device(config)# ipv6 unicast-routing
Devoce(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
```

Example: Configuring RDNSS

The following example shows how to configure Recursive DNS Server list on Ethernet 3/3 and verify the same.

```
Device> enable
Device# configure terminal
Device(config)# interface ethernet 3/3
Device(config-if)# ipv6 nd ra dns server 1::1 1000 sequence 0
Device(config-if)# ipv6 nd ra dns server 2::1 infinite sequence 1
Device(config-if)# exit

Device(config)# show ipv6 nd ra dns server

Recursive DNS Server List on: mgmt0
Suppress DNS Server List: No
Recursive DNS Server List on: Ethernet3/3
  Suppress DNS Server List: No
  DNS Server 1: 1::1 Lifetime:1000 seconds Sequence:0
  DNS Server 2: 2::1 Infinite Sequence:1
```

Example: Configuring DNSSL

The following example shows how to configure DNS Search list on Ethernet 3/3 and verify the same.

```
Device> enable
Device# configure terminal
Device(config)# interface ethernet 3/3
Device(config-if)# ipv6 nd ra dns search-list cisco.com 100 sequence 1
Device(config-if)# ipv6 nd ra dns search-list ind.cisco.com 100 sequence 2
Device(config-if)# exit

Device(config)# show ipv6 nd ra dns search-list

DNS Search List on: mgmt0
Suppress DNS Search List: No
 DNS Search List on: Ethernet3/3
  Suppress DNS Search List: No
  DNS Server 1:cisco.com 100 Sequence:1
  DNS Server 2:ind.cisco.com 100 Sequence:2
```

Example: Configuring Default Router Preference

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

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

Example: Configuring IPv6 ICMP Rate Limiting

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

```
Device> enable
Device# configure terminal
Device(config)#ipv6 icmp error-interval 50 20
```

Example: Configuring Static Routing for IPv6

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

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

Example: Enabling PBR on an Interface

In the following example, a route map named pbr-dest-1 is created and configured, specifying packet match criteria and desired policy-route action. PBR is then enabled on GigabitEthernet interface 0/0/1.

```
Device> enable
Device# configure terminal
Device(config)# ipv6 access-list match-dest-1
Device(config)# permit ipv6 any 2001:DB8:2001:1760::/32
Device(config)# route-map pbr-dest-1 permit 10
Device(config)# match ipv6 address match-dest-1
```

```
Device(config)# set interface GigabitEthernet 0/0/0
Device(config)# interface GigabitEthernet0/0/1
Device(config-if)# ipv6 policy-route-map interactive
```

Example: Enabling Local PBR for IPv6

In the following 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:

```
Device> enable
Device# configure terminal
Device(config)# ipv6 access-list src-90
Device(config)# permit ipv6 host 2001:DB8:2003::90 2001:DB8:2001:1000::/64
Device(config)# route-map pbr-src-90 permit 10
Device(config)# match ipv6 address src-90
Device(config)# set ipv6 next-hop 2001:DB8:2003:1::95
Device(config)# ipv6 local policy route-map pbr-src-90
```

Example: Displaying IPv6

This is an example of the output from the **show ipv6 interface** command:

```
Device> enable
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

Standards and RFCs

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

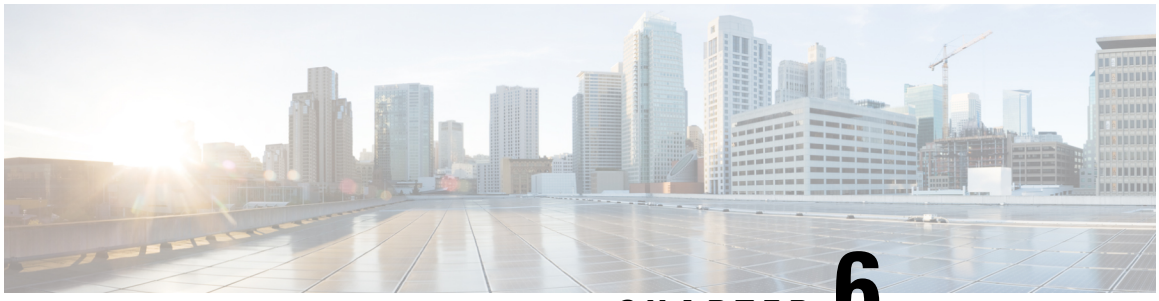
Feature History for IPv6 Unicast Routing

This table provides release and related information for the features explained in this module.

These features are available in all the releases subsequent to the one they were introduced in, unless noted otherwise.

Release	Feature	Feature Information
Cisco IOS XE Everest 16.5.1a	IPv6 Unicast Routing	IPv4 users can move to IPv6 and receive services such as end-to-end security, quality of service (QoS), and globally unique addresses.
Cisco IOS XE Gibraltar 16.11.1	RFC 5453	Support for RFC 5453 was introduced.
	IPv6 Router Advertisement Options for DNS Configuration	IPv6 Router Advertisement provides options to allow IPv6 hosts to perform automatic DNS configuration.

Use the Cisco Feature Navigator to find information about platform and software image support. To access Cisco Feature Navigator, go to http://www.cisco.com/go/cfn.



CHAPTER 6

Configuring RIP

- Information About RIP, on page 111
- How to Configure Routing Information Protocol, on page 112
- Configuration Examples for Routing Information Protocol, on page 120
- Feature History for Routing Information Protocol, on page 121

Information About RIP

The Routing Information Protocol (RIP) is an interior gateway protocol (IGP) created for use in small, homogeneous networks. It is a distance-vector routing protocol that uses broadcast User Datagram Protocol (UDP) data packets to exchange routing information. The protocol is documented in RFC 1058. You can find detailed information about RIP in *IP Routing Fundamentals,* published by Cisco Press.

Note RIP is supported in the Network Essentials feature set.

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

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

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

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

How to Configure Routing Information Protocol

The following sections provide configurational information about RIP.

Default RIP Configuration

Table 10: 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.

Feature	Default Setting
Validate-update-source	Enabled.
Version	Receives RIP Version 1 and 2 packets; sends Version 1 packets.

Configuring Basic RIP Parameters

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.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password, if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**ip routing** **Example:** `Device(config)# ip routing`	Enables IP routing. (Required only if IP routing is disabled.)
Step 4	**router rip** **Example:** `Device(config)# router rip`	Enables a RIP routing process, and enter router configuration mode.
Step 5	**network** *network number* **Example:** `Device(config-router)# network 12.0.0.0`	Associates a network with a RIP routing process. You can specify multiple **network** commands. RIP routing updates are sent and received through interfaces only on these networks. **Note** You must configure a network number for the RIP commands to take effect.
Step 6	**neighbor** *ip-address* **Example:** `Device(config-router)# neighbor 10.2.5.1`	(Optional) Defines a neighboring router with which to exchange routing information. This step allows routing updates from RIP (normally a broadcast protocol) to reach nonbroadcast networks.

	Command or Action	Purpose
Step 7	**offset-list** [*access-list number* | *name*] {**in** | **out**} *offset* [*type number*] **Example:** `Device(config-router)# offset-list 103 in 10`	(Optional) Applies an offset list to routing metrics to increase incoming and outgoing metrics to routes learned through RIP. You can limit the offset list with an access list or an interface.
Step 8	**timers basic** *update invalid holddown flush* **Example:** `Device(config-router)# timers basic 45 360 400 300`	(Optional) Adjusts routing protocol timers. Valid ranges for all timers are 0 to 4294967295 seconds. • *update*—The time between sending routing updates. The default is 30 seconds. • *invalid*—The timer after which a route is declared invalid. The default is 180 seconds. • *holddown*—The time before a route is removed from the routing table. The default is 180 seconds. • *flush*—The amount of time for which routing updates are postponed. The default is 240 seconds.
Step 9	**version** {**1** | **2**} **Example:** `Device(config-router)# version 2`	(Optional) Configures the switch to receive and send only RIP Version 1 or RIP Version 2 packets. By default, the switch receives Version 1 and 2 but sends only Version 1. You can also use the interface commands **ip rip** {**send** | **receive**} **version 1** | **2** | **1 2**} to control what versions are used for sending and receiving on interfaces.
Step 10	**no auto summary** **Example:** `Device(config-router)# no auto summary`	(Optional) Disables automatic summarization. By default, the switch summarizes subprefixes when crossing classful network boundaries. Disable summarization (RIP Version 2 only) to advertise subnet and host routing information to classful network boundaries.
Step 11	**output-delay** *delay* **Example:** `Device(config-router)# output-delay 8`	(Optional) Adds interpacket delay for RIP updates sent. By default, packets in a multiple-packet RIP update have no delay added between packets. If you are sending packets to a lower-speed device, you can add an interpacket delay in the range of 8 to 50 milliseconds.
Step 12	**end** **Example:** `Device(config-router)# end`	Returns to privileged EXEC mode.
Step 13	**show ip protocols** **Example:** `Device# show ip protocols`	Verifies your entries.

	Command or Action	Purpose
Step 14	**copy running-config startup-config** **Example:** `Device# copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

Configuring RIP Authentication

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, no authentication is performed, not even the default.

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

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface** *interface-id* **Example:** `Device(config)# interface gigabitethernet 1/0/1`	Enters interface configuration mode, and specifies the interface to configure.
Step 4	**ip rip authentication key-chain** *name-of-chain* **Example:** `Device(config-if)# ip rip authentication key-chain trees`	Enables RIP authentication.
Step 5	**ip rip authentication mode** {**text** | **md5**} **Example:** `Device(config-if)# ip rip authentication mode md5`	Configures the interface to use plain text authentication (the default) or MD5 digest authentication.

	Command or Action	Purpose
Step 6	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode.
Step 7	**show running-config** **Example:** `Device# show running-config`	Verifies your entries.
Step 8	**copy running-config startup-config** **Example:** `Device# copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

Configuring RIP for IPv6

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,

To configure RIP routing for IPv6, perform this procedure:

Before you begin

Before configuring the switch to run IPv6 RIP, you must 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, and enable IPv6 on any Layer 3 interfaces on which IPv6 RIP is to be enabled.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**ipv6 router rip** *name* **Example:** `Device(config)# ipv6 router rip cisco`	Configures an IPv6 RIP routing process, and enters router configuration mode for the process.

	Command or Action	Purpose
Step 4	**maximum-paths** *number-paths* **Example:** `Device(config-router)# maximum-paths 6`	(Optional) Define the maximum number of equal-cost routes that IPv6 RIP can support. The range is from 1 to 32, and the default is 16 routes.
Step 5	**exit** **Example:** `Device(config-router)# exit`	Returns to global configuration mode.
Step 6	**interface** *interface-id* **Example:** `Device(config)# interface gigabitethernet 1/0/1`	Enters interface configuration mode, and specifies the Layer 3 interface to configure.
Step 7	**ipv6 rip** *name* **enable** **Example:** `Device(config-if)# ipv6 rip cisco enable`	Enables the specified IPv6 RIP routing process on the interface.
Step 8	**ipv6 rip** *name* **default-information** {**only** | **originate**} **Example:** `Device(config-if)# ipv6 rip cisco default-information only`	(Optional) Originates the IPv6 default route (::/0) into the RIP routing process updates sent from the specified interface. **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.
Step 9	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode.

Step 10	Use one of the following:
	• **show ipv6 rip** [*name*] [**interface** *interface-id*] [**database**] [**next-hops**]
	• **show ipv6 rip**
	Example:
	`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.

	Command or Action	Purpose
Step 11	**copy running-config startup-config** **Example:** `Device# copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

Configuring Summary Addresses and Split Horizon

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

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

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface** *interface-id* **Example:** `Device(config)# interface gigabitethernet 1/0/1`	Enters interface configuration mode, and specifies the Layer 3 interface to configure.
Step 4	**ip address** *ip-address subnet-mask* **Example:** `Device(config-if)# ip address 10.1.1.10 255.255.255.0`	Configures the IP address and IP subnet.

	Command or Action	Purpose
Step 5	**ip summary-address rip ip address** *ip-network mask* **Example:** `Device(config-if)# ip summary-address rip ip address 10.1.1.30 255.255.255.0`	Configures the IP address to be summarized and the IP network mask.
Step 6	**no ip split horizon** **Example:** `Device(config-if)# no ip split horizon`	Disables split horizon on the interface.
Step 7	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode.
Step 8	**show ip interface** *interface-id* **Example:** `Device# show ip interface gigabitethernet 1/0/1`	Verifies your entries.
Step 9	**copy running-config startup-config** **Example:** `Device# copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

Configuring Split Horizon

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. 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.

Note In general, we do not recommend disabling split horizon unless you are certain that your application requires it to properly advertise routes.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:**	Enables privileged EXEC mode. Enter your password if prompted.

	Command or Action	Purpose
	`Device> enable`	
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface** *interface-id* **Example:** `Device(config)# interface gigabitethernet 1/0/1`	Enters interface configuration mode, and specifies the interface to configure.
Step 4	**ip address** *ip-address subnet-mask* **Example:** `Device(config-if)# ip address 10.1.1.10 255.255.255.0`	Configures the IP address and IP subnet.
Step 5	**no ip split-horizon** **Example:** `Device(config-if)# no ip split-horizon`	Disables split horizon on the interface.
Step 6	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode.
Step 7	**show ip interface** *interface-id* **Example:** `Device# show ip interface gigabitethernet 1/0/1`	Verifies your entries.
Step 8	**copy running-config startup-config** **Example:** `Device# copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

Configuration Examples for Routing Information Protocol

The following sections provide configuration examples for RIP.

Example: Configuring RIP 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:

```
Device> enable
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
```

Configuration Example for Summary Addresses and Split Horizon

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
```

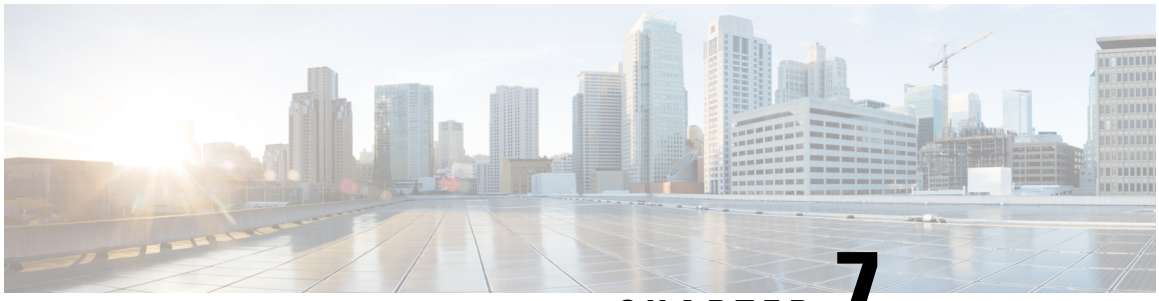
Feature History for Routing Information Protocol

This table provides release and related information for the features explained in this module.

These features are available in all the releases subsequent to the one they were introduced in, unless noted otherwise.

Release	Feature	Feature Information
Cisco IOS XE Everest 16.5.1a	Routing Information Protocol	The Routing Information Protocol is an interior gateway protocol (IGP) created for use in small and homogeneous networks.

Use the Cisco Feature Navigator to find information about platform and software image support. To access Cisco Feature Navigator, go to http://www.cisco.com/go/cfn.



CHAPTER 7

Configuring OSPF

- Information About OSPF, on page 123
- How to Configure OSPF, on page 127
- Monitoring OSPF, on page 140
- Configuration Examples for OSPF, on page 141
- Configuration Examples for OSPF, on page 141
- Example: Configuring Basic OSPF Parameters, on page 141
- Feature History for Open Shortest Path First, on page 141

Information About OSPF

OSPF is an Interior Gateway Protocol (IGP) designed expressly for IP networks, supporting IP subnetting and tagging of externally derived routing information. OSPF also allows packet authentication and uses IP multicast when sending and receiving packets. The Cisco implementation supports RFC 1253, OSPF management information base (MIB).

The Cisco implementation conforms to the OSPF Version 2 specifications with these key features:

- Definition of stub areas is supported.
- Routes learned through any IP routing protocol can be redistributed into another IP routing protocol. At the intradomain level, this means that OSPF can import routes learned through EIGRP and RIP. OSPF routes can also be exported into RIP.
- Plain text and MD5 authentication among neighboring routers within an area is supported.
- Configurable routing interface parameters include interface output cost, retransmission interval, interface transmit delay, router priority, router dead and hello intervals, and authentication key.
- Virtual links are supported.
- Not-so-stubby-areas (NSSAs) per RFC 1587are supported.

OSPF typically requires coordination among many internal routers, area border routers (ABRs) connected to multiple areas, and autonomous system boundary routers (ASBRs). The minimum configuration would use all default parameter values, no authentication, and interfaces assigned to areas. If you customize your environment, you must ensure coordinated configuration of all routers.

OSPF for IPv6

The switch supports Open Shortest Path First (OSPF) for IPv6, a link-state protocol for IP.

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

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

OSPF Nonstop Forwarding

The switch or switch stack supports two levels of nonstop forwarding (NSF):

- OSPF NSF Awareness, on page 124
- OSPF NSF Capability, on page 124

OSPF NSF Awareness

The Network Advantage license supports OSPF NSF Awareness for IPv4. When the neighboring router is NSF-capable, the Layer 3 device continues to forward packets from the neighboring router during the interval between the primary Route Processor (RP) in a router crashing and the backup RP taking over, or while the primary RP is manually reloaded for a non-disruptive software upgrade.

This feature cannot be disabled.

OSPF NSF Capability

The Network Advantage license supports the OSPFv2 NSF IETF format in addition to the OSPFv2 NSF Cisco format that is supported in earlier releases. For information about this feature, see : *NSF—OSPF (RFC 3623 OSPF Graceful Restart)*.

The Network Advantage license also supports OSPF NSF-capable routing for IPv4 for better convergence and lower traffic loss following a stack's active switch change.

When an active switch change occurs in an OSPF NSF-capable stack, the new active switch must do two things to resynchronize its link-state database with its OSPF neighbors:

- Release the available OSPF neighbors on the network without resetting the neighbor relationship.
- Reacquire the contents of the link-state database for the network.

After an active switch change, the new active switch sends an OSPF NSF signal to neighboring NSF-aware devices. A device recognizes this signal to mean that it should not reset the neighbor relationship with the stack. As the NSF-capable active switch receives signals from other routes on the network, it begins to rebuild its neighbor list.

When the neighbor relationships are reestablished, the NSF-capable active switch resynchronizes its database with its NSF-aware neighbors, and routing information is exchanged between the OSPF neighbors. The new active switch uses this routing information to remove stale routes, to update the routing information database (RIB), and to update the forwarding information base (FIB) with the new information. The OSPF protocols then fully converge.

Note OSPF NSF requires that all neighbor networking devices be NSF-aware. If an NSF-capable router discovers non-NSF aware neighbors on a network segment, it disables NSF capabilities for that segment. Other network segments where all devices are NSF-aware or NSF-capable continue to provide NSF capabilities.

Use the **nsf** OSPF routing configuration command to enable OSPF NSF routing. Use the **show ip ospf** privileged EXEC command to verify that it is enabled.

OSPF Area Parameters

You can optionally configure several OSPF area parameters. These parameters include authentication for password-based protection against unauthorized access to an area, stub areas, and not-so-stubby-areas (NSSAs). Stub areas are areas into which information on external routes is not sent. Instead, the area border router (ABR) generates a default external route into the stub area for destinations outside the autonomous system (AS). An NSSA does not flood all LSAs from the core into the area, but can import AS external routes within the area by redistribution.

Route summarization is the consolidation of advertised addresses into a single summary route to be advertised by other areas. If network numbers are contiguous, you can use the **area range** router configuration command to configure the ABR to advertise a summary route that covers all networks in the range.

Other OSPF Parameters

You can optionally configure other OSPF parameters in router configuration mode.

- Route summarization: When redistributing routes from other protocols. Each route is advertised individually in an external LSA. To help decrease the size of the OSPF link state database, you can use the **summary-address** router configuration command to advertise a single router for all the redistributed routes included in a specified network address and mask.
- Virtual links: In OSPF, all areas must be connected to a backbone area. You can establish a virtual link in case of a backbone-continuity break by configuring two Area Border Routers as endpoints of a virtual link. Configuration information includes the identity of the other virtual endpoint (the other ABR) and the nonbackbone link that the two routers have in common (the transit area). Virtual links cannot be configured through a stub area.
- Default route: When you specifically configure redistribution of routes into an OSPF routing domain, the route automatically becomes an autonomous system boundary router (ASBR). You can force the ASBR to generate a default route into the OSPF routing domain.
- Domain Name Server (DNS) names for use in all OSPF **show** privileged EXEC command displays makes it easier to identify a router than displaying it by router ID or neighbor ID.
- Default Metrics: OSPF calculates the OSPF metric for an interface according to the bandwidth of the interface. The metric is calculated as *ref-bw* divided by bandwidth, where *ref* is 10 by default, and bandwidth (*bw*) is specified by the **bandwidth** interface configuration command. For multiple links with high bandwidth, you can specify a larger number to differentiate the cost on those links.
- Administrative distance is a rating of the trustworthiness of a routing information source, an integer between 0 and 255, with a higher value meaning a lower trust rating. An administrative distance of 255 means the routing information source cannot be trusted at all and should be ignored. OSPF uses three different administrative distances: routes within an area (interarea), routes to another area (interarea),

and routes from another routing domain learned through redistribution (external). You can change any of the distance values.

- Passive interfaces: Because interfaces between two devices on an Ethernet represent only one network segment, to prevent OSPF from sending hello packets for the sending interface, you must configure the sending device to be a passive interface. Both devices can identify each other through the hello packet for the receiving interface.
- Route calculation timers: You can configure the delay time between when OSPF receives a topology change and when it starts the shortest path first (SPF) calculation and the hold time between two SPF calculations.
- Log neighbor changes: You can configure the router to send a syslog message when an OSPF neighbor state changes, providing a high-level view of changes in the router.

LSA Group Pacing

The OSPF LSA group pacing feature allows the router to group OSPF LSAs and pace the refreshing, check-summing, and aging functions for more efficient router use. This feature is enabled by default with a 4-minute default pacing interval, and you will not usually need to modify this parameter. The optimum group pacing interval is inversely proportional to the number of LSAs the router is refreshing, check-summing, and aging. For example, if you have approximately 10,000 LSAs in the database, decreasing the pacing interval would benefit you. If you have a very small database (40 to 100 LSAs), increasing the pacing interval to 10 to 20 minutes might benefit you slightly.

Loopback Interfaces

OSPF uses the highest IP address configured on the interfaces as its router ID. If this interface is down or removed, the OSPF process must recalculate a new router ID and resend all its routing information out its interfaces. If a loopback interface is configured with an IP address, OSPF uses this IP address as its router ID, even if other interfaces have higher IP addresses. Because loopback interfaces never fail, this provides greater stability. OSPF automatically prefers a loopback interface over other interfaces, and it chooses the highest IP address among all loopback interfaces.

How to Configure OSPF

Default OSPF Configuration

Table 11: Default OSPF Configuration

Feature	Default Setting
Interface parameters	Cost: Retransmit interval: 5 seconds. Transmit delay: 1 second. Priority: 1. Hello interval: 10 seconds. Dead interval: 4 times the hello interval. No authentication. No password specified. MD5 authentication disabled.
Area	Authentication type: 0 (no authentication). Default cost: 1. Range: Disabled. Stub: No stub area defined. NSSA: No NSSA area defined.
Auto cost	100 Mb/s.
Default-information originate	Disabled. When enabled, the default metric setting is 10, and the external rout is Type 2.
Default metric	Built-in, automatic metric translation, as appropriate for each routing protoco
Distance OSPF	dist1 (all routes within an area): 110. dist2 (all routes from one area to anothe dist3 (routes from other routing domains): 110.
OSPF database filter	Disabled. All outgoing link-state advertisements (LSAs) are flooded to the in
IP OSPF name lookup	Disabled.
Log adjacency changes	Enabled.
Neighbor	None specified.
Neighbor database filter	Disabled. All outgoing LSAs are flooded to the neighbor.
Network area	Disabled.

Feature	Default Setting
Nonstop Forwarding (NSF) awareness	Enabled. Allows Layer 3 switches to continue forwarding packets from a neighb NSF-capable router during hardware or software changes.
Router ID	No OSPF routing process defined.
Summary address	Disabled.
Timers LSA group pacing	240 seconds.
Timers shortest path first (spf)	spf delay: 50 miliseconds; spf-holdtime: 200 miliseconds.
Virtual link	No area ID or router ID defined. Hello interval: 10 seconds. Retransmit interval: 5 seconds. Transmit delay: 1 second. Dead interval: 40 seconds. Authentication key: no key predefined. Message-digest key (MD5): no key predefined.

Configuring Basic OSPF Parameters

To enable OSPF, create an OSPF routing process, specify the range of IP addresses to associate with the routing process, and assign area IDs to be associated with that range. For switches running the Network Essentialsimage, you can configure either the Cisco OSPFv2 NSF format or the IETF OSPFv2 NSF format.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device>enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device#configure terminal`	Enters global configuration mode.
Step 3	**router ospf** process-id **Example:** `Device(config)#router ospf 15`	Enables OSPF routing, and enter router configuration mode. The process ID is an internally used identification parameter that is locally assigned and can be any positive integer. Each OSPF routing process has a unique value.

	Command or Action	Purpose
		Note OSPF for Routed Access supports only one OSPFv2 and one OSPFv3 instance with a maximum number of 1000 dynamically learned routes.
Step 4	**nsf cisco** [**enforce global**] **Example:** `Device(config)#nsf cisco enforce global`	(Optional) Enables Cisco NSF operations for OSPF. The **enforce global** keyword cancels NSF restart when non-NSF-aware neighboring networking devices are detected. **Note** Enter the command in Step 3 or Step 4, and go to Step 5.
Step 5	**nsf ietf** [**restart-interval** *seconds*] **Example:** `Device(config)#nsf ietf restart-interval 60`	(Optional) Enables IETF NSF operations for OSPF. The **restart-interval** keyword specifies the length of the graceful restart interval, in seconds. The range is from 1 to 1800. The default is 120. **Note** Enter the command in Step 3 or Step 4, and go to Step 5.
Step 6	**network** address wildcard-mask **area** area-id **Example:** `Device(config)#network 10.1.1.1 255.240.0.0 area 20`	Define an interface on which OSPF runs and the area ID for that interface. You can use the wildcard-mask to use a single command to define one or more multiple interfaces to be associated with a specific OSPF area. The area ID can be a decimal value or an IP address.
Step 7	**end** **Example:** `Device(config)#end`	Returns to privileged EXEC mode.
Step 8	**show ip protocols** **Example:** `Device#show ip protocols`	Verifies your entries.
Step 9	**copy running-config startup-config** **Example:** `Device#copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

Configuring OSPF for IPv6

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.

To configure OSPF routing for IPv6, perform this procedure:

Before you begin

You can customize OSPF for IPv6 for your network. However, the defaults for OSPF in IPv6 are set 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, you must 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, and enable IPv6 on Layer 3 interfaces on which you are enabling IPv6 OSPF.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**ipv6 router ospf** *process-id* **Example:** `Device(config)# ipv6 router ospf 21`	Enables OSPF router configuration mode for the process. The process ID is the number assigned administratively when enabling the OSPF for IPv6 routing process. It is locally assigned and can be a positive integer from 1 to 65535.
Step 4	**area** *area-id* **range** {*ipv6-prefix/prefix length*} [**advertise** | **not-advertise**] [**cost** *cost*] **Example:** `Device(config)# area .3 range 2001:0DB8::/32 not-advertise`	(Optional) Consolidates and summarizes routes at an area boundary. • *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*—The destination IPv6 network and 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. • **advertise**—(Optional) Sets the address range status to advertise and generate a Type 3 summary link-state advertisement (LSA).

	Command or Action	Purpose
		• **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) Sets the metric or cost for this summary route, which is used during OSPF SPF calculation to determine the shortest paths to the destination. The value can be 0 to 16777215.
Step 5	**maximum paths** *number-paths* **Example:** `Device(config)# maximum paths 16`	(Optional) Defines the maximum number of equal-cost routes to the same destination that IPv6 OSPF should enter in the routing table. The range is from 1 to 32, and the default is 16 paths.
Step 6	**exit** **Example:** `Device(config-if)# exit`	Returns to global configuration mode.
Step 7	**interface** *interface-id* **Example:** `Device(config)# interface gigabitethernet 1/0/1`	Enters interface configuration mode, and specifies the Layer 3 interface to configure.
Step 8	**ipv6 ospf** *process-id* **area** *area-id* [**instance** *instance-id*] **Example:** `Device(config-if)# ipv6 ospf 21 area .3`	Enables OSPF for IPv6 on the interface. • **instance** *instance-id*—(Optional) Instance identifier.
Step 9	**end** **Example:** `Device(config-if)# end`	Returns to privileged EXEC mode.

Step 10	Use one of the following:
	• **show ipv6 ospf** [*process-id*] [*area-id*] **interface** [*interface-id*]
	• **show ipv6 ospf** [*process-id*] [*area-id*]
	Example:
	`Device# show ipv6 ospf 21 interface gigabitethernet2/0/1`
	or
	`Device# show ipv6 ospf 21`
		• Displays information about OSPF interfaces.
		• Displays general information about OSPF routing processes.

Step 11	**copy running-config startup-config** **Example:** `Device# copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

Configuring OSPF Interfaces

You can use the **ip ospf** interface configuration commands to modify interface-specific OSPF parameters. You are not required to modify any of these parameters, but some interface parameters (hello interval, dead interval, and authentication key) must be consistent across all routers in an attached network. If you modify these parameters, be sure all routers in the network have compatible values.

Note The **ip ospf** interface configuration commands are all optional.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device#configure terminal`	Enters global configuration mode.
Step 3	**interface** interface-id **Example:** `Device(config)#interface gigabitethernet 1/0/1`	Enters interface configuration mode, and specifies the Layer 3 interface to configure.
Step 4	**ip ospf cost** cost **Example:** `Device(config-if)#ip ospf cost 8`	(Optional) Explicitly specifies the cost of sending a packet on the interface.
Step 5	**ip ospf retransmit-interval** seconds **Example:** `Device(config-if)#ip ospf transmit-interval 10`	(Optional) Specifies the number of seconds between link state advertisement transmissions. The range is 1 to 65535 seconds. The default is 5 seconds.
Step 6	**ip ospf transmit-delay** seconds **Example:** `Device(config-if)#ip ospf transmit-delay 2`	(Optional) Sets the estimated number of seconds to wait before sending a link state update packet. The range is 1 to 65535 seconds. The default is 1 second.

	Command or Action	Purpose
Step 7	**ip ospf priority** number **Example:** `Device(config-if)#ip ospf priority 5`	(Optional) Sets priority to help find the OSPF designated router for a network. The range is from 0 to 255. The default is 1.
Step 8	**ip ospf hello-interval** seconds **Example:** `Device(config-if)#ip ospf hello-interval 12`	(Optional) Sets the number of seconds between hello packets sent on an OSPF interface. The value must be the same for all nodes on a network. The range is 1 to 65535 seconds. The default is 10 seconds.
Step 9	**ip ospf dead-interval** seconds **Example:** `Device(config-if)#ip ospf dead-interval 8`	(Optional) Sets the number of seconds after the last device hello packet was seen before its neighbors declare the OSPF router to be down. The value must be the same for all nodes on a network. The range is 1 to 65535 seconds. The default is 4 times the hello interval.
Step 10	**ip ospf authentication-key** key **Example:** `Device(config-if)#ip ospf authentication-key password`	(Optional) Assign a password to be used by neighboring OSPF routers. The password can be any string of keyboard-entered characters up to 8 bytes in length. All neighboring routers on the same network must have the same password to exchange OSPF information.
Step 11	**ip ospf message digest-key** keyid **md5** key **Example:** `Device(config-if)#ip ospf message digest-key 16 md5 your1pass`	(Optional) Enables MDS authentication. • *keyid*—An identifier from 1 to 255. • *key*—An alphanumeric password of up to 16 bytes.
Step 12	**ip ospf database-filter all out** **Example:** `Device(config-if)#ip ospf database-filter all out`	(Optional) Block flooding of OSPF LSA packets to the interface. By default, OSPF floods new LSAs over all interfaces in the same area, except the interface on which the LSA arrives.
Step 13	**end** **Example:** `Device(config)#end`	Returns to privileged EXEC mode.
Step 14	**show ip ospf interface** [*interface-name*] **Example:** `Device#show ip ospf interface`	Displays OSPF-related interface information.
Step 15	**show ip ospf neighbor detail** **Example:**	Displays NSF awareness status of neighbor switch. The output matches one of these examples: • *Options is 0x52*

	Command or Action	Purpose
	`Device#show ip ospf neighbor detail`	*LLS Options is 0x1 (LR)* When both of these lines appear, the neighbor switch is NSF aware. • *Options is 0x42*—This means the neighbor switch is not NSF aware.
Step 16	**copy running-config startup-config** **Example:** `Device#copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

Configuring OSPF Area Parameters

Before you begin

Note The OSPF **area** router configuration commands are all optional.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device>enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device#configure terminal`	Enters global configuration mode.
Step 3	**router ospf** process-id **Example:** `Device(config)#router ospf 109`	Enables OSPF routing, and enter router configuration mode.
Step 4	**area** *area-id* **authentication** **Example:** `Device(config-router)#area 1 authentication`	(Optional) Allow password-based protection against unauthorized access to the identified area. The identifier can be either a decimal value or an IP address.

	Command or Action	Purpose
Step 5	**area** *area-id* **authentication message-digest** **Example:** `Device(config-router)#area 1 authentication message-digest`	(Optional) Enables MD5 authentication on the area.
Step 6	**area** *area-id* **stub** [**no-summary**] **Example:** `Device(config-router)#area 1 stub`	(Optional) Define an area as a stub area. The **no-summary** keyword prevents an ABR from sending summary link advertisements into the stub area.
Step 7	**area** *area-id* **nssa** [**no-redistribution**] [**default-information-originate**] [**no-summary**] **Example:** `Device(config-router)#area 1 nssa default-information-originate`	(Optional) Defines an area as a not-so-stubby-area. Every router within the same area must agree that the area is NSSA. Select one of these keywords: • **no-redistribution**—Select when the router is an NSSA ABR and you want the **redistribute** command to import routes into normal areas, but not into the NSSA. • **default-information-originate**—Select on an ABR to allow importing type 7 LSAs into the NSSA. • **no-redistribution**—Select to not send summary LSAs into the NSSA.
Step 8	**area** *area-id* **range** *address mask* **Example:** `Device(config-router)#area 1 range 255.240.0.0`	(Optional) Specifies an address range for which a single route is advertised. Use this command only with area border routers.
Step 9	**end** **Example:** `Device(config)#end`	Returns to privileged EXEC mode.
Step 10	**show ip ospf** [*process-id*] **Example:** `Device#show ip ospf`	Displays information about the OSPF routing process in general or for a specific process ID to verify configuration.
Step 11	**show ip ospf** [*process-id* [*area-id*]] **database** **Example:** `Device#show ip osfp database`	Displays lists of information related to the OSPF database for a specific router.
Step 12	**copy running-config startup-config** **Example:**	(Optional) Saves your entries in the configuration file.

	Command or Action	Purpose
	`Device#copy running-config startup-config`	

Configuring Other OSPF Parameters

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device>enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device#configure terminal`	Enters global configuration mode.
Step 3	**router ospf** *process-id* **Example:** `Device(config)#router ospf 10`	Enables OSPF routing, and enter router configuration mode.
Step 4	**summary-address** *address mask* **Example:** `Device(config)#summary-address 10.1.1.1 255.255.255.0`	(Optional) Specifies an address and IP subnet mask for redistributed routes so that only one summary route is advertised.
Step 5	**area** *area-id* **virtual-link** *router-id* [**hello-interval** *seconds*] [**retransmit-interval** *seconds*] [**trans**] [[**authentication-key** *key*] | **message-digest-key** *keyid* **md5** *key*]] **Example:** `Device(config)#area 2 virtual-link 192.168.255.1 hello-interval 5`	(Optional) Establishes a virtual link and set its parameters.
Step 6	**default-information originate** [**always**] [**metric** *metric-value*] [**metric-type** *type-value*] [**route-map** *map-name*] **Example:** `Device(config)#default-information originate metric 100 metric-type 1`	(Optional) Forces the ASBR to generate a default route into the OSPF routing domain. Parameters are all optional.

	Command or Action	Purpose
Step 7	**ip ospf name-lookup** **Example:** `Device(config)#ip ospf name-lookup`	(Optional) Configures DNS name lookup. The default is disabled.
Step 8	**ip auto-cost reference-bandwidth** *ref-bw* **Example:** `Device(config)#ip auto-cost reference-bandwidth 5`	(Optional) Specifies an address range for which a single route will be advertised. Use this command only with area border routers.
Step 9	**distance ospf** {[**inter-area** *dist1*] [**inter-area** *dist2*] [**external** *dist3*]} **Example:** `Device(config)#distance ospf inter-area 150`	(Optional) Changes the OSPF distance values. The default distance for each type of route is 110. The range is 1 to 255.
Step 10	**passive-interface** *type number* **Example:** `Device(config)#passive-interface gigabitethernet 1/0/6`	(Optional) Suppresses the sending of hello packets through the specified interface.
Step 11	**timers throttle spf** *spf-delay spf-holdtime spf-wait* **Example:** `Device(config)#timers throttle spf 200 100 100`	(Optional) Configures route calculation timers. • *spf-delay*—Delay between receiving a change to SPF calculation. The range is from 1 to 600000 miliseconds. • *spf-holdtime*—Delay between first and second SPF calculation. The range is from 1 to 600000 in milliseconds. • *spf-wait*—Maximum wait time in milliseconds for SPF calculations. The range is from 1 to 600000 in milliseconds.
Step 12	**ospf log-adj-changes** **Example:** `Device(config)#ospf log-adj-changes`	(Optional) Sends syslog message when a neighbor state changes.
Step 13	**end** **Example:** `Device(config)#end`	Returns to privileged EXEC mode.

	Command or Action	Purpose
Step 14	**show ip ospf** [*process-id* [*area-id*]] **database** **Example:** `Device#show ip ospf database`	Displays lists of information related to the OSPF database for a specific router.
Step 15	**copy running-config startup-config** **Example:** `Device#copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

Changing LSA Group Pacing

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device>enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device#configure terminal`	Enters global configuration mode.
Step 3	**router ospf** *process-id* **Example:** `Device(config)#router ospf 25`	Enables OSPF routing, and enter router configuration mode.
Step 4	**timers lsa-group-pacing** *seconds* **Example:** `Device(config-router)#timers lsa-group-pacing 15`	Changes the group pacing of LSAs.
Step 5	**end** **Example:** `Device(config)#end`	Returns to privileged EXEC mode.
Step 6	**show running-config** **Example:**	Verifies your entries.

	Command or Action	Purpose
	`Device#show running-config`	
Step 7	**copy running-config startup-config** **Example:** `Device#copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

Configuring a Loopback Interface

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device>enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device#configure terminal`	Enters global configuration mode.
Step 3	**interface loopback 0** **Example:** `Device(config)#interface loopback 0`	Creates a loopback interface, and enter interface configuration mode.
Step 4	**ip address** address mask **Example:** `Device(config-if)#ip address 10.1.1.5 255.255.240.0`	Assign an IP address to this interface.
Step 5	**end** **Example:** `Device(config)#end`	Returns to privileged EXEC mode.
Step 6	**show ip interface** **Example:** `Device#show ip interface`	Verifies your entries.

	Command or Action	Purpose
Step 7	**copy running-config startup-config** **Example:** `Device#copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

Monitoring OSPF

You can display specific statistics such as the contents of IP routing tables, caches, and databases.

Table 12: Show IP OSPF Statistics Commands

Command	Purpose
show ip ospf [*process-id*]	Displays general inform processes.
show ip ospf [*process-id*] **database** [**router**] [*link-state-id*] **show ip ospf** [*process-id*] **database** [**router**] [**self-originate**] **show ip ospf** [*process-id*] **database** [**router**] [**adv-router** [*ip-address*]] **show ip ospf** [*process-id*] **database** [**network**] [*link-state-id*] **show ip ospf** [*process-id*] **database** [**summary**] [*link-state-id*] **show ip ospf** [*process-id*] **database** [**asbr-summary**] [*link-state-id*] **show ip ospf** [*process-id*] **database** [**external**] [*link-state-id*] **show ip ospf** [*process-id area-id*] **database** [**database-summary**]	Displays lists of informa
show ip ospf border-routes	Displays the internal OS entries.
show ip ospf interface [*interface-name*]	Displays OSPF-related
show ip ospf neighbor [*interface-name*] [*neighbor-id*] **detail**	Displays OSPF interfac
show ip ospf virtual-links	Displays OSPF-related

Configuration Examples for OSPF

Configuration Examples for OSPF

Example: Configuring Basic OSPF Parameters

This example shows how to configure an OSPF routing process and assign it a process number of 109:

```
Device(config)#router ospf 109
Device(config-router)#network 131.108.0.0 255.255.255.0 area 24
```

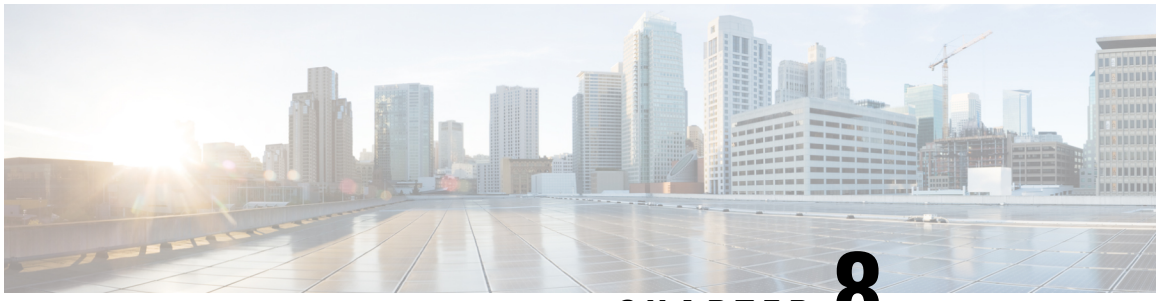
Feature History for Open Shortest Path First

This table provides release and related information for the features explained in this module.

These features are available in all the releases subsequent to the one they were introduced in, unless noted otherwise.

Release	Feature	Feature Information
Cisco IOS XE Everest 16.5.1a	Open Shortest Path First	OSPF is an Interior Gateway Protocol (IGP) designed expressly for IP networks, supporting IP subnetting and tagging of externally derived routing information.

Use the Cisco Feature Navigator to find information about platform and software image support. To access Cisco Feature Navigator, go to http://www.cisco.com/go/cfn.



CHAPTER 8

Configuring OSPFv3 Fast Convergence - LSA and SPF Throttling

- Information About OSPFv3 Fast Convergence: LSA and SPF Throttling, on page 143
- How to Configure OSPFv3 Fast Convergence: LSA and SPF Throttling, on page 143
- Example: Configuring LSA and SPF Throttling for OSPFv3 Fast Convergence, on page 146
- Additional References, on page 146
- Feature History for OSPFv3 Fast Convergence: LSA and SPF Throttling , on page 146

Information About OSPFv3 Fast Convergence: LSA and SPF Throttling

The Open Shortest Path First version 3 (OSPFv3) link-state advertisement (LSAs) and shortest-path first (SPF) throttling feature provides a dynamic mechanism to slow down link-state advertisement updates in OSPFv3 during times of network instability. It also allows faster OSPFv3 convergence by providing LSA rate limiting in milliseconds.

OSPFv3 can use static timers for rate-limiting SPF calculation and LSA generation. Although these timers are configurable, the values used are specified in seconds, which poses a limitation on OSPFv3 convergence. LSA and SPF throttling achieves subsecond convergence by providing a more sophisticated SPF and LSA rate-limiting mechanism that is able to react quickly to changes and also provide stability and protection during prolonged periods of instability.

How to Configure OSPFv3 Fast Convergence: LSA and SPF Throttling

The following sections provide configuration information about OSPFv3 Fast Convergence: LSA and SPF throttling.

Tuning LSA and SPF Timers for OSPFv3 Fast Convergence

To tune LSA and SPF timers for OSPFv3 fast convergence, perform this procedure:

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **router ospfv3** [*process-id*]
4. **timers lsa arrival** *milliseconds*
5. **timers pacing flood** *milliseconds*
6. **timers pacing lsa-group** *seconds*
7. **timers pacing retransmission** *milliseconds*

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**router ospfv3** [*process-id*] **Example:** `Device(config)# router ospfv3 1`	Enables OSPFv3 router configuration mode for the IPv4 or IPv6 address family.
Step 4	**timers lsa arrival** *milliseconds* **Example:** `Device(config-rtr)# timers lsa arrival 300`	Sets the minimum interval at which the software accepts the same LSA from OSPFv3 neighbors.
Step 5	**timers pacing flood** *milliseconds* **Example:** `Device(config-rtr)# timers pacing flood 30`	Configures LSA flood packet pacing.
Step 6	**timers pacing lsa-group** *seconds* **Example:** `Device(config-router)# timers pacing lsa-group 300`	Changes the interval at which OSPFv3 LSAs are collected into a group and refreshed, checksummed, or aged.
Step 7	**timers pacing retransmission** *milliseconds* **Example:** `Device(config-router)# timers pacing retransmission 100`	Configures LSA retransmission packet pacing in IPv4 OSPFv3.

Configuring LSA and SPF Throttling for OSPFv3 Fast Convergence

To configure LSA and SPF throttling for OSPFv3 fast convergence, perform this procedure:

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **ipv6 router ospf** *process-id*
4. **timers throttle spf** *spf-start spf-hold spf-max-wait*
5. **timers throttle lsa** *start-interval hold-interval max-interval*
6. **timers lsa arrival** *milliseconds*
7. **timers pacing flood** *milliseconds*

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**ipv6 router ospf** *process-id* **Example:** `Device(config)# ipv6 router ospf 1`	Enables OSPFv3 router configuration mode.
Step 4	**timers throttle spf** *spf-start spf-hold spf-max-wait* **Example:** `Device(config-rtr)# timers throttle spf 200 200 200`	Turns on SPF throttling.
Step 5	**timers throttle lsa** *start-interval hold-interval max-interval* **Example:** `Device(config-rtr)# timers throttle lsa 300 300 300`	Sets rate-limiting values for OSPFv3 LSA generation.
Step 6	**timers lsa arrival** *milliseconds* **Example:** `Device(config-rtr)# timers lsa arrival 300`	Sets the minimum interval at which the software accepts the same LSA from OSPFv3 neighbors.
Step 7	**timers pacing flood** *milliseconds* **Example:** `Device(config-rtr)# timers pacing flood 30`	Configures LSA flood packet pacing.

Example: Configuring LSA and SPF Throttling for OSPFv3 Fast Convergence

The following example show how to display the configuration values for SPF and LSA throttling timers:

```
Device# show ipv6 ospf

Routing Process "ospfv3 1" with ID 10.9.4.1
Event-log enabled, Maximum number of events: 1000, Mode: cyclic
 It is an autonomous system boundary router
 Redistributing External Routes from,
    ospf 2
 Initial SPF schedule delay 5000 msecs
 Minimum hold time between two consecutive SPFs 10000 msecs
 Maximum wait time between two consecutive SPFs 10000 msecs
 Minimum LSA interval 5 secs
 Minimum LSA arrival 1000 msecs
```

Additional References

Related Documents

Related Topic	Document Title
IPv6 addressing and connectivity	*IPv6 Configuration Guide*
OSPFv3 Fast Convergence: LSA and SPF Throttling	*OSPF Shortest Path First Throttling* module

Standards and RFCs

Standard/RFC	Title
RFCs for IPv6	IPv6 RFCs

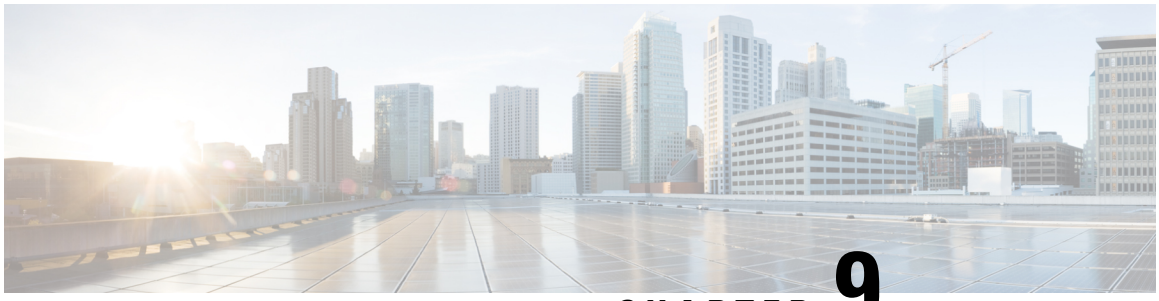
Feature History for OSPFv3 Fast Convergence: LSA and SPF Throttling

This table provides release and related information for the features explained in this module.

These features are available in all the releases subsequent to the one they were introduced in, unless noted otherwise.

Release	Feature	Feature Information
Cisco IOS XE Gibraltar 16.11.1	OSPFv3 Fast Convergence - LSA and SPF Throttling	The Open Shortest Path First version 3 (OSPFv3) LSAs and SPF throttling feature provides a dynamic mechanism to slow down link-state advertisement updates in OSPFv3 during times of network instability

Use the Cisco Feature Navigator to find information about platform and software image support. To access Cisco Feature Navigator, go to http://www.cisco.com/go/cfn.



CHAPTER 9

Configuring OSPFv3 Authentication Support with IPsec

- Information About OSPFv3 Authentication Support with IPsec, on page 149
- How to Configure OSPFv3 Authentication Support with IPsec, on page 151
- How to Configure OSPFv3 IPSec ESP Encryption and Authentication, on page 152
- Configuration Examples for OSPFv3 Authentication Support with IPsec, on page 155
- Configuration Example for OSPFv3 IPSec ESP Encryption and Authentication, on page 155
- Feature History for OSPFv3 Authentication Support with IPsec, on page 156

Information About OSPFv3 Authentication Support with IPsec

The following sections provide information about OSPFv3 authentication support with IPsec and OSPFv3 virtual links.

Overview of OSPFv3 Authentication Support with IPsec

In order to ensure that OSPFv3 packets are not altered and resent to the device, causing the device to behave in a way not desired by its system administrators, OSPFv3 packets must be authenticated. OSPFv3 uses the IPsec secure socket to add authentication to OSPFv3 packets.

OSPFv3 requires the use of IPsec to enable authentication. Crypto images are required to use authentication because only crypto images include the IPsec needed for use with OSPFv3.

In OSPFv3, authentication fields have been removed from OSPFv3 packet headers. When OSPFv3 runs on IPv6, OSPFv3 requires the IPv6 authentication header or IPv6 Encapsulating Security Payload (ESP) header to ensure integrity, authentication, and confidentiality of routing exchanges. IPv6 authentication header and ESP extension headers can be used to provide authentication and confidentiality to OSPFv3.

To use the IPsec authentication header, you must enable the **ipv6 ospf authentication** command. To use the IPsec ESP header, you must enable the **ipv6 ospf encryption** command. The ESP header can be applied alone or along with the authentication header, and when ESP is used, both encryption and authentication are provided. Security services can be provided between a pair of communicating hosts, between a pair of communicating security gateways, or between a security gateway and a host.

To configure IPsec, you should configure a security policy, which is a combination of the security policy index (SPI) and the key (the key is used to create and validate the hash value). IPsec for OSPFv3 can be configured on an interface or on an OSPFv3 area. For higher security, you should configure a different policy

on each interface that is configured with IPsec. If you configure IPsec for an OSPFv3 area, the policy is applied to all the interfaces in that area, except for the interfaces that have IPsec configured directly. After IPsec is configured for OSPFv3, IPsec is invisible to you.

The IPsecure socket is used by applications to secure traffic by allowing the application to open, listen, and close secure sockets. The binding between the application and the secure socket layer also allows the secure socket layer to inform the application of changes to the socket, such as connection open and close events. The IPsecure socket is able to identify the socket, that is, it can identify the local and remote addresses, masks, ports, and protocol that carry the traffic requiring security.

Each interface has a secure socket state, which can be one of the following:

- NULL: Do not create a secure socket for the interface if authentication is configured for the area.
- DOWN: IPsec has been configured for the interface (or the area that contains the interface), but OSPFv3 has either not requested IPsec to create a secure socket for this interface, or there is an error condition.

Note OSPFv3 does not send or accept packets while in the DOWN state.

- GOING UP: OSPFv3 has requested a secure socket from IPsec and is waiting for a CRYPTO_SS_SOCKET_UP message from IPsec.
- UP: OSPFv3 has received a CRYPTO_SS_SOCKET_UP message from IPsec.
- CLOSING: The secure socket for the interface has been closed. A new socket can be opened for the interface, in which case, the current secure socket makes the transition to the DOWN state. Otherwise, the interface becomes UNCONFIGURED.
- UNCONFIGURED: Authentication is not configured on the interface.

OSPFv3 Virtual Links

For each virtual link, a primary security information data block is created. Because a secure socket must be opened on each interface, there will be a corresponding security information datablock for each interface in the transit area. The secure socket state is kept in the interface's security information datablock. The **state** field in the primary security information datablock shows the status of all the secure sockets opened for the corresponding virtual link. If all the secure sockets are UP, the security state for the virtual link is set to UP.

Packets sent on a virtual link with IPsec must use predetermined source and destination addresses. The first local area address found in the device's intra-area-prefix Link-State Advertisement (LSA) for the area is used as the source address. This source address is saved in the area's data structure and used when secure sockets are opened and packets sent over the corresponding virtual link. The virtual link does not transition to the point-to-point state until a source address is selected. Also, when the source or destination address changes, the previous secure sockets must be closed and new secure sockets opened.

Note Virtual links are not supported for the IPv4 address family.

How to Configure OSPFv3 Authentication Support with IPsec

The following sections provide information on how to define authentication on an interface, and how to define authentication in an OSPFv3 area.

Defining Authentication on an Interface

To define authentication on an interface, perform this procedure:

Before you begin

Before you configure IPsec on an interface, you must configure OSPFv3 on that interface.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password, if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface** *type number* **Example:** `Device(config)# interface ethernet 1/0/1`	Configures an interface.

Step 4	Choose one of the following:
	• **ospfv3 authentication** {{**ipsec spi** *spi* {**md5** | **sha1**}} | { *key-encryption-type key* } | **null**}
	• **ipv6 ospf authentication** {**null** | **ipsec spi** *spi* *authentication-algorithm* [*key-encryption-type*] [*key*]}
	Example:
	`Device(config-if)# ospfv3 authentication md5 0 275761340947681324733020312097 27`
	OR
	`Device(config-if)# ipv6 ospf authentication ipsec spi 500 md5 1234567890abcdef1234567890abcdef`
		Specifies the authentication type for an interface.

Defining Authentication in an OSPFv3 Area

To define authentication in an OSPFv3 area, perform this procedure:

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password, if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**ipv6 router ospf** *process-id* **Example:** `Device(config)# ipv6 router ospf 1`	Enables OSPFv3 router configuration mode.
Step 4	**area** *area-id* **authentication ipsec spi** *spi authentication-algorithm* [*key-encryption-type*] *key* **Example:** `Device(config-router)# area 1 authentication ipsec spi 678 md5 1234567890ABCDEF1234567890ABCDEF`	Enables authentication in an OSPFv3 area.

How to Configure OSPFv3 IPSec ESP Encryption and Authentication

The following sections provide information on how to define encryption on an interface, how to define encryption in an OSPFv3 area, and how to defining authentication and encryption for a virtual link in an OSPFv3 area:

Defining Encryption on an Interface

To define encryption on an interface, perform this procedure.

Before you begin

Before you configure IPsec on an interface, you must configure OSPFv3 on that interface.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password, if prompted.

	Command or Action	Purpose
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface** *type number* **Example:** `Device(config)# interface ethernet 1/0/1`	Configures an interface.

Step 4	Choose one of the following:
	• **ospfv3 authentication** {**ipsec spi** *spi* **esp** *encryption-algorithm key-encryption-type key authentication-algorithm key-encryption-type key* | **null**}
	• **ipv6 ospf authentication** {**ipsec spi** *spi* **esp** {*encryption-algorithm* [*key-encryption-type*] *key* | **null**} *authentication-algorithm* [*key-encryption-type*] *key*] | **null**}
	Example:
	`Device(config-if)# ospfv3 encryption ipsec spi 1001 esp null md5 0 275761340947681324733020312097 27`
	OR
	`Device(config-if)# ipv6 ospf encryption ipsec spi 1001 esp null sha1 123456789A123456789B123456789C123456789D`
		Specifies the encryption type for the interface.

Defining Encryption in an OSPFv3 Area

To define encryption in an OSPFv3 area, perform this procedure.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password, if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**ipv6 router ospf** *process-id* **Example:** `Device(config)# ipv6 router ospf 1`	Enables OSPFv3 router configuration mode.

	Command or Action	Purpose
Step 4	**area** *area-id* **encryption ipsec spi** *spi* **esp** {*encryption-algorithm* [*key-encryption-type*] *key* | **null**} *authentication-algorithm* [*key-encryption-type*] *key* **Example:** `Device(config-router)# area 1 encryption ipsec spi 500 esp null md5 1aaa2bbb3ccc4ddd5eee6fff7aaa8bbb`	Enables encryption in an OSPFv3 area.

Defining Authentication and Encryption for a Virtual Link in an OSPFv3 Area

To define authentication and encryption for a virtual link in an OSPFv3 area, perform this procedure:

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password, if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**ipv6 router ospf** *process-id* **Example:** `Device(config)# ipv6 router ospf 1`	Enables OSPFv3 router configuration mode.
Step 4	**area** *area-id* **virtual-link** *router-id* **authentication ipsec spi** *spi authentication-algorithm* [*key-encryption-type*] *key* **Example:** `Device(config-router)# area 1 virtual-link 10.0.0.1 authentication ipsec spi 940 md5 1234567890ABCDEF1234567890ABCDEF`	Enables authentication for virtual links in an OSPFv3 area.
Step 5	**area** *area-id* **virtual-link** *router-id* **authentication ipsec spi** *spi* **esp** {*encryption-algorithm* [*key-encryption-type*] *key* | **null**} *authentication-algorithm* [*key-encryption-type*] *key* **Example:** `Device(config-router)# area 1 virtual-link 10.1.0.1 hello-interval 2 dead-interval 10 encryption ipsec spi 3944 esp null sha1 123456789A123456789B123456789C123456789D`	Enables encryption for virtual links in the OSPFv3 area.

Configuration Examples for OSPFv3 Authentication Support with IPsec

The following sections provide various configuration examples for OSPFv3 authentication support with IPsec.

Example: Defining Authentication on an Interface

The following example shows how to define authentication on Ethernet interface 1/0/1:

```
Device> enable
Device# configure terminal
Device(config)# interface Ethernet1/0/1
Device(config-if)# ipv6 enable
Device(config-if)# ipv6 ospf 1 area 0
Device(config-if)# ipv6 ospf authentication ipsec spi 500 md5 1234567890ABCDEF1234567890ABCDEF
Device(config-if)# exit
Device(config)# interface Ethernet1/0/1
Device(config-if)# ipv6 enable
Device(config-if)# ipv6 ospf authentication null
Device(config-if)# ipv6 ospf 1 area 0
```

Example: Defining Authentication in an OSPFv3 Area

The following example shows how to define authentication on OSPFv3 area 0:

```
Device> enable
Device# configure terminal
Device(config)# ipv6 router ospf 1
Device(config-router)# router-id 10.11.11.1
Device(config-router)# area 0 authentication ipsec spi 1000 md5
1234567890ABCDEF1234567890ABCDEF
```

Configuration Example for OSPFv3 IPSec ESP Encryption and Authentication

The following section provides an example to verify OSPFv3 IPsec ESP encryption and authentication.

Example: Verifying Encryption in an OSPFv3 Area

The following is a sample output of the **show ipv6 ospf interface** command:

```
Device> enable
Device# show ipv6 ospf interface
```

```
Ethernet1/0/1 is up, line protocol is up
  Link Local Address 2001:0DB1:A8BB:CCFF:FE00:6E00, Interface ID 2
  Area 0, Process ID 1, Instance ID 0, Router ID 10.10.10.1
  Network Type BROADCAST, Cost:10
  MD5 Authentication (Area) SPI 1000, secure socket state UP (errors:0)
  Transmit Delay is 1 sec, State BDR, Priority 1
  Designated Router (ID) 10.11.11.1, local address 2001:0DB1:A8BB:CCFF:FE00:6F00
  Backup Designated router (ID) 10.10.10.1, local address
FE80::A8BB:CCFF:FE00:6E00
  Timer intervals configured, Hello 10, Dead 40, Wait 40, Retransmit 5
    Hello due in 00:00:03
  Index 1/1/1, flood queue length 0
  Next 0x0(0)/0x0(0)/0x0(0)
  Last flood scan length is 1, maximum is 1
  Last flood scan time is 0 msec, maximum is 0 msec
  Neighbor Count is 1, Adjacent neighbor count is 1
    Adjacent with neighbor 10.11.11.1  (Designated Router)
  Suppress hello for 0 neighbor(s)
```

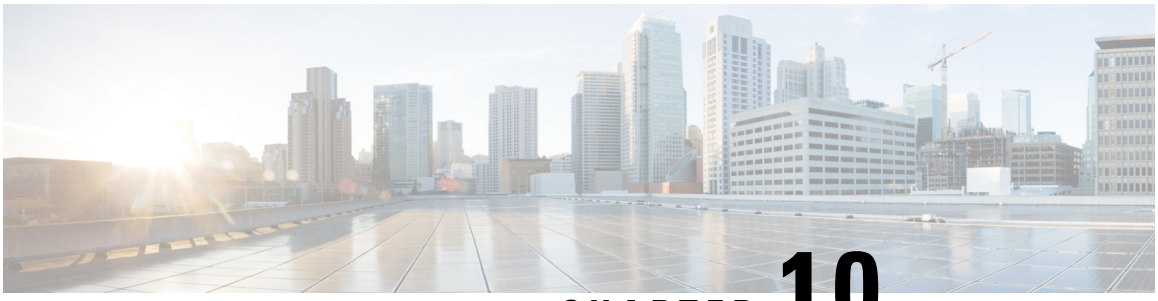
Feature History for OSPFv3 Authentication Support with IPsec

This table provides release and related information for the features explained in this module.

These features are available in all the releases subsequent to the one they were introduced in, unless noted otherwise.

Release	Feature	Feature Information
Cisco IOS XE Fuji 16.8.1a	OSPFv3 Authentication Support with IPsec	OSPFv3 uses the IPsec secure socket to add authentication to OSPFv3 packets.

Use the Cisco Feature Navigator to find information about platform and software image support. To access Cisco Feature Navigator, go to http://www.cisco.com/go/cfn.



CHAPTER 10

Configuring OSPFv3 Authentication Trailer

- Information About the OSPFv3 Authentication Trailer, on page 157
- How to Configure the OSPFv3 Authentication Trailer, on page 158
- Configuration Examples for the OSPFv3 Authentication Trailer, on page 160
- Additional References for OSPFv3 Authentication Trailer, on page 161
- Feature Information for the OSPFv3 Authentication Trailer, on page 162

Information About the OSPFv3 Authentication Trailer

The OSPFv3 authentication trailer feature (as defined in RFC 7166) provides an alternative mechanism to authenticate Open Shortest Path First version 3 (OSPFv3) protocol packets. Prior to the OSPFv3 authentication trailer, OSPFv3 IPsec (as defined in RFC 4552) was the only mechanism for authenticating protocol packets. The OSPFv3 authentication trailer feature also provides packet replay protection through sequence number and do not have platform dependencies.

To perform non-IPsec cryptographic authentication, devices attach a special data block, that is, authentication trailer, to the end of the OSPFv3 packet. The length of the authentication trailer is not included in the length of the OSPFv3 packet but is included in the IPv6 payload length. The Link-Local Signaling (LLS) block is established by the L-bit setting in the **OSPFv3 Options** field in OSPFv3 hello packets and database description packets. If present, the LLS data block is included in the cryptographic authentication computation along with the OSPFv3 packet.

A new authentication trailer bit is introduced into the **OSPFv3 Options** field. OSPFv3 devices must set the authentication trailer bit in OSPFv3 hello packets and database description packets to indicate that all the packets on this link include an authentication trailer. For OSPFv3 hello packets and database description packets, the authentication trailer bit indicates that the authentication trailer is present. For other OSPFv3 packet types, the OSPFv3 authentication trailer bit setting from the OSPFv3 hello and database description setting is preserved in the OSPFv3 neighbor data structure. OSPFv3 packet types that do not include the **OSPFv3 Options** field uses the setting from the neighbor data structure to determine whether the authentication trailer is expected. The authentication trailer bit must be set in all OSPFv3 hello packets and database description packets that contain an authentication trailer.

To configure the authentication trailer, OSPFv3 utilizes the existing Cisco IOS **key chain** command. For outgoing OSPFv3 packets, the following rules are used to select the key from the key chain:

- Select the key that is the last to expire.
- If two keys have the same stop time, select the one with the highest key ID.

The security association ID maps to the authentication algorithm and the secret key that is used to generate and verify the message digest. If the authentication is configured, but the last valid key is expired, the packets are sent using the key. A syslog message is also generated. If no valid key is available, the packet is sent without the authentication trailer. When packets are received, the key ID is used to look up the data for that key. If the key ID is not found in the key chain, or if the security association is not valid, the packet is dropped. Otherwise, the packet is verified using the algorithm and the key that is configured for the key ID. Key chains support rollover using key lifetimes. A new key can be added to a key chain with the send start time set in the future. This setting allows the new key to be configured on all the devices before the keys are actually used.

The hello packets have higher priority than other OSPFv3 packets, and therefore, can get reordered on the outgoing interface. This reordering can create problems with sequence number verification on neighboring devices. To prevent sequence mismatch, OSPFv3 verifies the sequence number separately for each packet type. See RFC 7166 for more details on the authentication procedure.

During the initial rollover of the authentication trailer feature on the network, adjacency can be maintained between the devices that are configured with authentication routes and devices that are yet to be configured by using the deployment mode. When the deployment mode is configured using the **authentication mode deployment** command, the packets are processed differently. For the outgoing packets, OSPF checksum is calculated even if authentication trailer is configured. For incoming packets, the packets without authentication trailer or the wrong authentication hash are dropped. In the deployment mode, the **show ospfv3 neighbor** *detail* command shows the last packet authentication status. This information can be used to verify if the authentication trailer feature is working before the mode is set to normal with the **authentication mode normal** command.

How to Configure the OSPFv3 Authentication Trailer

To configure OSPFv3 authentication trailer, perform this procedure:

Before you begin

An authentication key is required for configuring OSPFv3 authentication trailer. For more information on configuring an authentication key, see *How to Configure Authentication Keys* in *Protocol-Independent Features*.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password, if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface** *type number* **Example:** `Device(config)# interface GigabitEthernet 2/0/1`	Specifies the interface type and number.

	Command or Action	Purpose
Step 4	**ospfv3** [*pid*] [**ipv4** | **ipv6**] **authentication** {**key-chain** *chain-name* | **null**} **Example:** `Device(config-if)# ospfv3 1 ipv6 authentication key-chain ospf-1`	Specifies the authentication type for an OSPFv3 instance.
Step 5	**router ospfv3** [*process-id*] **Example:** `Device(config-if)# router ospfv3 1`	Enters OSPFv3 router configuration mode.
Step 6	**address-family ipv6 unicast** **Example:** `Device(config-router)# address-family ipv6 unicast`	Configures the IPv6 address family in the OSPFv3 process and enters IPv6 address family configuration mode.
Step 7	**area** *area-id* **authentication** {**key-chain** *chain-name* | **null**} **Example:** `Device(config-router-af)# area 1 authentication key-chain ospf-chain-1`	Configures the authentication trailer on all interfaces in the OSPFv3 area.
Step 8	**area** *area-id* **virtual-link** *router-id* **authentication key-chain** *chain-name* **Example:** `Device(config-router-af)# area 1 virtual-link 1.1.1.1 authentication key-chain ospf-chain-1`	Configures the authentication for virtual links.
Step 9	**area** *area-id* **sham-link** *source-address destination-address* **authentication key-chain** *chain-name* **Example:** `Device(config-router-af)# area 1 sham-link 1.1.1.1 1.1.1.0 authentication key-chain ospf-chain-1`	Configures the authentication for sham-links.
Step 10	**authentication mode** {**deployment** | **normal**} **Example:** `Device(config-router-af)# authentication mode deployment`	(Optional) Specifies the type of authentication used for the OSPFv3 instance. The **deployment** keyword provides adjacency between configured and the unconfigured authentication devices.
Step 11	**end** **Example:** `Device(config-router-af)# end`	Exits IPv6 address family configuration mode and returns to privileged EXEC mode.
Step 12	**show ospfv3 interface** **Example:** `Device# show ospfv3`	(Optional) Displays OSPFv3-related interface information.

	Command or Action	Purpose
Step 13	**show ospfv3 neighbor** [*detail*] **Example:** `Device# show ospfv3 neighbor detail`	(Optional) Displays OSPFv3 neighbor information on a per-interface basis.
Step 14	**debug ospfv3** **Example:** `Device# debug ospfv3`	(Optional) Displays debugging information for OSPFv3.

Configuration Examples for the OSPFv3 Authentication Trailer

The following sections provide examples on how to configure the OSPFv3 authentication trailer and how to verify the OSPFv3 authentication trailer configuration.

Example: Configuring the OSPFv3 Authentication Trailer

The following example shows how to define authentication trailer on GigabitEthernet interface 1/0/1:

```
Device> enable
Device# configure terminal
Device(config)# interface GigabitEthernet 1/0/1
Device(config-if)# ospfv3 1 ipv6 authentication key-chain ospf-1
Device(config-if)# router ospfv3 1
Device(config-router)# address-family ipv6 unicast
Device(config-router-af)# area 1 authentication key-chain ospf-1
Device(config-router-af)# area 1 virtual-link 1.1.1.1 authentication key-chain ospf-1
Device(config-router-af)# area 1 sham-link 1.1.1.1 authentication key-chain ospf-1
Device(config-router-af)# authentication mode deployment
Device(config-router-af)# end
Device(config)# key chain ospf-1
Device(config-keychain)# key 1
Device(config-keychain-key)# key-string ospf
Device(config-keychain-key)# cryptographic-algorithm hmac-sha-256
!
```

Example: Verifying OSPFv3 Authentication Trailer

The following example shows the output of the **show ospfv3** command.

```
Device# show ospfv3
 OSPFv3 1 address-family ipv6
 Router ID 1.1.1.1
…
RFC1583 compatibility enabled
 Authentication configured with deployment key lifetime
 Active Key-chains:
```

```
  Key chain ospf-1: Send key 1, Algorithm HMAC-SHA-256, Number of interfaces 1
    Area BACKBONE(0)
```

The following example shows the output of the **show ospfv3 neighbor detail** command.

```
Device# show ospfv3 neighbor detail
OSPFv3 1 address-family ipv6 (router-id 2.2.2.2)
 Neighbor 1.1.1.1
    In the area 0 via interface GigabitEthernet0/0
    Neighbor: interface-id 2, link-local address FE80::A8BB:CCFF:FE01:2D00
    Neighbor priority is 1, State is FULL, 6 state changes
    DR is 2.2.2.2 BDR is 1.1.1.1
    Options is 0x000413 in Hello (V6-Bit, E-Bit, R-Bit, AT-Bit)
    Options is 0x000413 in DBD (V6-Bit, E-Bit, R-Bit, AT-Bit)
    Dead timer due in 00:00:33
    Neighbor is up for 00:05:07
    Last packet authentication succeed
    Index 1/1/1, retransmission queue length 0, number of retransmission 0
    First 0x0(0)/0x0(0)/0x0(0) Next 0x0(0)/0x0(0)/0x0(0)
    Last retransmission scan length is 0, maximum is 0
    Last retransmission scan time is 0 msec, maximum is 0 msec
```

The following example shows the output of the **show ospfv3 interface** command.

```
Device# show ospfv3 interface
GigabitEthernet1/0/1 is up, line protocol is up
  Cryptographic authentication enabled
    Sending SA: Key 25, Algorithm HMAC-SHA-256 - key chain ospf-1
    Last retransmission scan time is 0 msec, maximum is 0 msec
```

Additional References for OSPFv3 Authentication Trailer

Related Documents

Related Topic	Document Title
Configuring OSPF features	*IP Routing: OSPF Configuration Guide*

Standards and RFCs

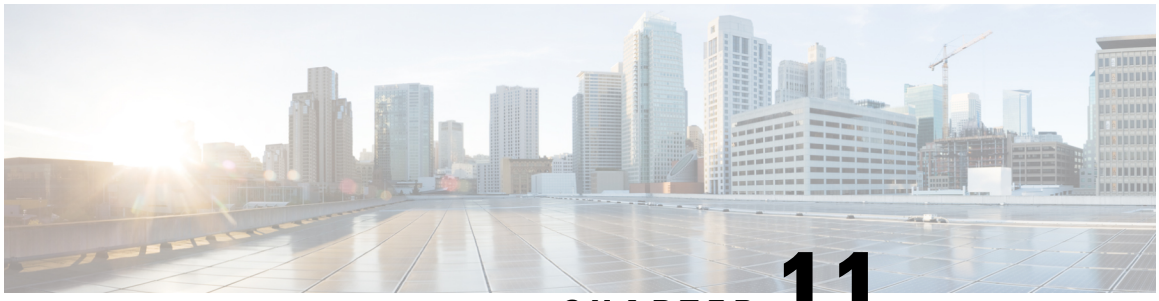
Standard/RFC	Document Title
RFC 7166	RFC for Supporting Authentication Trailer for OSPFv3
RFC 6506	RFC for Supporting Authentication Trailer for OSPFv3
RFC 4552	RFC for Authentication/Confidentiality for OSPFv3

Feature Information for the OSPFv3 Authentication Trailer

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Table 13: Feature Information for the OSPFv3 Authentication Trailer

Feature Name	Releases	Feature Information
OSPFv3 Authentication Trailer	Cisco IOS XE Fuji 16.8.1a	OSPFv3 Authentication Trailer feature provides a mechanism to authenticate OSPFv3 protocol packets as an alternative to existing OSPFv3 IPsec authentication.



CHAPTER 11

Configuring OSPFv3 Limit on Number of Redistributed Routes

- Restrictions for OSPFv3 Limit on Number of Redistributed Routes, on page 163
- Prerequisites for OSPFv3 Limit on Number of Redistributed Routes, on page 163
- Information About OSPFv3 Limit on Number of Redistributed Routes, on page 163
- How to Configure an OSPFv3 Limit on the Number of Redistributed Routes, on page 164
- Configuration Examples for OSPFv3 Limit on Number of Redistributed Routes, on page 166
- Monitoring OSPFv3 Limit on Number of Redistributed Routes, on page 167
- Additional References , on page 167
- Feature History for OSPFv3 Limit on Number of Redistributed Routes, on page 167

Restrictions for OSPFv3 Limit on Number of Redistributed Routes

This feature is supported only for the IPv6 address family.

Prerequisites for OSPFv3 Limit on Number of Redistributed Routes

You must have Open Shortest Path First version 3 (OSPFv3) configured in your network either along with another protocol, or another OSPFv3 process for redistribution.

Information About OSPFv3 Limit on Number of Redistributed Routes

OSPFv3 supports a user-defined maximum number of prefixes (routes) that can be redistributed into OSPFv3 from other protocols or other OSPFv3 processes. Such a limit helps prevent the device from being flooded by too many redistributed routes

For example, if a large number of IP routes are sent into OSPFv3 for a network that allows redistribution of Border Gateway Protocol (BGP) into OSPFv3, the network can get severely flooded. Limiting the number of redistributed routes prevents this potential problem.

How to Configure an OSPFv3 Limit on the Number of Redistributed Routes

The following sections provide information on configuring an OSPFv3 limit on the number of redistributed routes.

Note The following procedures are mutually exclusive, that is, you can either limit the number of redistributed routes, or request a warning about the number of routes redistributed into OSPFv3.

Limiting the Number of OSPFv3 Redistributed Routes

This task describes how to limit the number of OSPFv3 redistributed routes. If the number of redistributed routes reaches the maximum value configured, no more routes are redistributed.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password, if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**router ospfv3** *process-id* **Example:** `Device(config)# router ospfv3 1`	Configures an OSPFv3 routing process.
Step 4	**address-family ipv6** [**unicast**] **Example:** `Device(config-router)# address-family ipv6 unicast`	Enters IPv6 address family configuration mode.
Step 5	**redistribute** *protocol* [*process-id*] [*as-number*] [**include-connected** {**level-1** | **level-1-2** | **level-2**} [**metric** *metric-value*] [**metric-type** *type-value*] [**nssa-only**] [**tag** *tag-value*] [**route-map** *map-tag*] **Example:** `Device(config-router-af)# redistribute eigrp 10`	Redistributes routes from one routing domain into another routing domain.

	Command or Action	Purpose
Step 6	**redistribute maximum-prefix** *maximum* [*threshold*] **Example:** `Device(config-router-af)# redistribute maximum-prefix 100 80`	Sets a maximum number of IPv6 prefixes that are allowed to be redistributed into OSPFv3. • There is no default value for the *maximum* argument. • The *threshold* value defaults to 75 percent. **Note** If the **warning-only** keyword is configured in this command, no limit is enforced; a warning message is logged.
Step 7	**exit-address-family** **Example:** `Device(config-router-af)# exit-address-family`	Exits IPv6 address family configuration mode.
Step 8	**end** **Example:** `Device(config-router)# end`	Exits router configuration mode.

Requesting a Warning Message About the Number of Routes Redistributed into OSPFv3

To request a warning message when the number of routes redistributed into OSPFv3 exceeds the configuration limit, perform this procedure:

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password, if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**router ospfv3** *process-id* **Example:** `Device(config)# router ospfv3 1`	Configures an OSPFv3 routing process.
Step 4	**address-family ipv6** [**unicast**] **Example:** `Device(config-router)# address-family ipv6 unicast`	Enters IPv6 address family configuration mode.

	Command or Action	Purpose
Step 5	**redistribute** *protocol* [*process-id*] [*as-number*] [**include-connected** {**level-1** | **level-1-2** | **level-2**} [**metric** *metric-value*] [**metric-type** *type-value*] [**nssa-only**] [**tag** *tag-value*] [**route-map** *map-tag*] **Example:** `Device(config-router-af)# redistribute eigrp 10`	Redistributes routes from one routing domain into another routing domain.
Step 6	**redistribute maximum-prefix** *maximum* [*threshold*] [**warning-only**] **Example:** `Device(config-router-af)# redistribute maximum-prefix 100 80 warning-only`	Causes a warning message to be logged when the maximum number of IP prefixes have been redistributed to OSPFv3. • Because the **warning-only** keyword is included, no limit is imposed on the number of redistributed prefixes into OSPFv3. • There is no default value for the *maximum* argument. • The *threshold* value defaults to 75 percent. • This example causes two warnings: one at 80 percent of 1000 (800 routes redistributed) and another at 1000 routes redistributed
Step 7	**end** **Example:** `Device(config-router)# end`	Exits router configuration mode.

Configuration Examples for OSPFv3 Limit on Number of Redistributed Routes

The following sections provide configuration examples for OSPFv3 limit on number of redistributed routes.

Example: OSPFv3 Limit on Number of Redistributed Routes

This example shows how to set a maximum of 1200 prefixes that can be redistributed into the OSPFv3 process 1. Prior to reaching the limit, when the number of prefixes that are redistributed reaches 80 percent of 1200 (960 prefixes), a warning message is logged. Another warning message is logged when the limit is reached and no more routes are redistributed.

```
Device> enable
Device# configure terminal
Device(config)# router ospfv3 1
Device(config-router)# address-family ipv6
Device(config-router-af)# redistribute static subnets
Device(config-router-af)# redistribute maximum-prefix 1200 80
```

Example: Requesting a Warning Message About the Number of Redistributed Routes

This example shows how to enable two warning messages to be logged, the first if the number of prefixes that are redistributed reaches 85 percent of 600 (510 prefixes), and the second if the number of redistributed routes reaches 600. However, the number of redistributed routes is not limited.

```
Device> enable
Device# configure terminal
Device(config)# router ospfv3 11
Device(config-router)# address-family ipv6
Device(config-router-af)# redistribute eigrp 10 subnets
Device(config-router-af)# redistribute maximum-prefix 600 85 warning-only
```

Monitoring OSPFv3 Limit on Number of Redistributed Routes

Use the privileged EXEC commands in the following table to monitor the limit on the number of redistributed routes.

Table 14: Commands to Monitor the OSPFv3 Limit on Number of Redistributed Routes

Command	Purpose
show ipv6 ospf [*process-id*] or **show ospfv3 ipv6** [*process-id*]	Displays general information about the OSPFv3 routing processes. The output includes the maximum limit of redistributed prefixes and the threshold for warning messages.

Additional References

Related Documents

Related Topic	Document Title
For complete syntax and usage information for the commands used in this chapter.	See the *Routing* section of the *Command Reference (Catalyst 9300 Series Switches)*

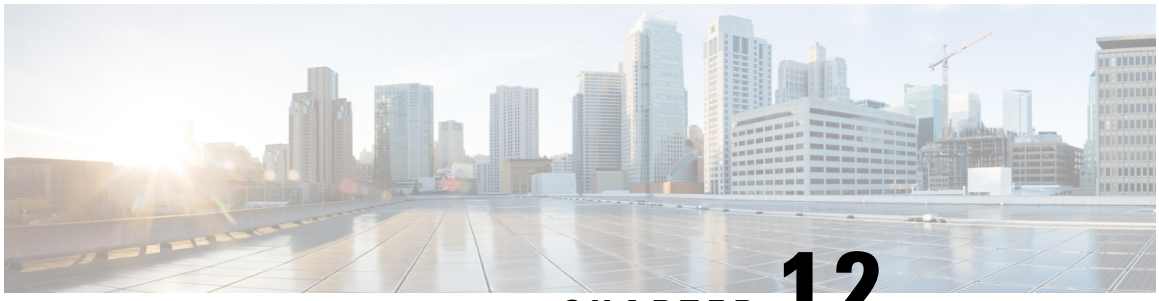
Feature History for OSPFv3 Limit on Number of Redistributed Routes

This table provides release and related information for the features explained in this module.

These features are available in all the releases subsequent to the one they were introduced in, unless noted otherwise.

Release	Feature	Feature Information
Cisco IOS XE Gibraltar 16.11.1	OSPFv3 Limit on Number of Redistributed Routes	OSPFv3 supports a user-defined maximum number of prefixes (routes) that can be redistributed into OSPFv3 from other protocols or other OSPFv3 processes.

Use the Cisco Feature Navigator to find information about platform and software image support. To access Cisco Feature Navigator, go to http://www.cisco.com/go/cfn.



CHAPTER 12

Configuring EIGRP

- Information About EIGRP, on page 169
- How to Configure EIGRP, on page 174
- Monitoring and Maintaining EIGRP, on page 181
- Feature Information for EIGRP, on page 182

Information About EIGRP

Enhanced IGRP (EIGRP) is a Cisco proprietary enhanced version of the IGRP. EIGRP uses the same distance vector algorithm and distance information as IGRP; however, the convergence properties and the operating efficiency of EIGRP are significantly improved.

The convergence technology employs an algorithm referred to as the Diffusing Update Algorithm (DUAL), which guarantees loop-free operation at every instant throughout a route computation and allows all devices involved in a topology change to synchronize at the same time. Routers that are not affected by topology changes are not involved in recomputations.

IP EIGRP provides increased network width. With RIP, the largest possible width of your network is 15 hops. Because the EIGRP metric is large enough to support thousands of hops, the only barrier to expanding the network is the transport-layer hop counter. EIGRP increments the transport control field only when an IP packet has traversed 15 routers and the next hop to the destination was learned through EIGRP. When a RIP route is used as the next hop to the destination, the transport control field is incremented as usual.

EIGRP IPv6

Switches support the Enhanced Interior Gateway Routing Protocol (EIGRP) for IPv6. It is configured on the interfaces on which it runs and does not require a global IPv6 address. Switches running Network Essentials only support EIGRPv6 stub routing.

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.

EIGRP Features

EIGRP offers these features:

- Fast convergence.
- Incremental updates when the state of a destination changes, instead of sending the entire contents of the routing table, minimizing the bandwidth required for EIGRP packets.
- Less CPU usage because full update packets need not be processed each time they are received.
- Protocol-independent neighbor discovery mechanism to learn about neighboring routers.
- Variable-length subnet masks (VLSMs).
- Arbitrary route summarization.
- EIGRP scales to large networks.

EIGRP Components

EIGRP has these four basic components:

- Neighbor discovery and recovery is the process that routers use to dynamically learn of other routers on their directly attached networks. Routers must also discover when their neighbors become unreachable or inoperative. Neighbor discovery and recovery is achieved with low overhead by periodically sending small hello packets. As long as hello packets are received, the Cisco IOS software can learn that a neighbor is alive and functioning. When this status is determined, the neighboring routers can exchange routing information.
- The reliable transport protocol is responsible for guaranteed, ordered delivery of EIGRP packets to all neighbors. It supports intermixed transmission of multicast and unicast packets. Some EIGRP packets must be sent reliably, and others need not be. For efficiency, reliability is provided only when necessary. For example, on a multiaccess network that has multicast capabilities (such as Ethernet), it is not necessary to send hellos reliably to all neighbors individually. Therefore, EIGRP sends a single multicast hello with an indication in the packet informing the receivers that the packet need not be acknowledged. Other types of packets (such as updates) require acknowledgment, which is shown in the packet. The reliable transport has a provision to send multicast packets quickly when there are unacknowledged packets pending. Doing so helps ensure that convergence time remains low in the presence of varying speed links.
- The DUAL finite state machine embodies the decision process for all route computations. It tracks all routes advertised by all neighbors. DUAL uses the distance information (known as a metric) to select efficient, loop-free paths. DUAL selects routes to be inserted into a routing table based on feasible successors. A successor is a neighboring router used for packet forwarding that has a least-cost path to a destination that is guaranteed not to be part of a routing loop. When there are no feasible successors, but there are neighbors advertising the destination, a recomputation must occur. This is the process whereby a new successor is determined. The amount of time it takes to recompute the route affects the convergence time. Recomputation is processor-intensive; it is advantageous to avoid recomputation if it is not necessary. When a topology change occurs, DUAL tests for feasible successors. If there are feasible successors, it uses any it finds to avoid unnecessary recomputation.
- The protocol-dependent modules are responsible for network layer protocol-specific tasks. An example is the IP EIGRP module, which is responsible for sending and receiving EIGRP packets that are

encapsulated in IP. It is also responsible for parsing EIGRP packets and informing DUAL of the new information received. EIGRP asks DUAL to make routing decisions, but the results are stored in the IP routing table. EIGRP is also responsible for redistributing routes learned by other IP routing protocols.

Note To enable EIGRP, the standalone switch or active switch must be running the Network Advantage license.

EIGRP Nonstop Forwarding

The device stack supports two levels of EIGRP nonstop forwarding:

- EIGRP NSF Awareness
- EIGRP NSF Capability

EIGRP NSF Awareness

The Network Advantage license supports EIGRP NSF Awareness for IPv4. When the neighboring router is NSF-capable, the Layer 3 device continues to forward packets from the neighboring router during the interval between the primary Route Processor (RP) in a router failing and the backup RP taking over, or while the primary RP is manually reloaded for a nondisruptive software upgrade. This feature cannot be disabled.

EIGRP NSF Capability

The Network Advantage license supports EIGRP Cisco NSF routing to speed up convergence and to eliminate traffic loss after an active switch change.

The Network Advantage license also supports EIGRP NSF-capable routing for IPv4 for better convergence and lower traffic loss following an active switch change. When an EIGRP NSF-capable active switch restarts or a new active switch starts up and NSF restarts, the device has no neighbors, and the topology table is empty. The device must bring up the interfaces, reacquire neighbors, and rebuild the topology and routing tables without interrupting the traffic directed toward the device stack. EIGRP peer routers maintain the routes learned from the new active switch and continue forwarding traffic through the NSF restart process.

To prevent an adjacency reset by the neighbors, the new active switch uses a new Restart (RS) bit in the EIGRP packet header to show the restart. When the neighbor receives this, it synchronizes the stack in its peer list and maintains the adjacency with the stack. The neighbor then sends its topology table to the active switch with the RS bit set to show that it is NSF-aware and is aiding the new active switch.

If at least one of the stack peer neighbors is NSF-aware, the active switch receives updates and rebuilds its database. Each NSF-aware neighbor sends an end of table (EOT) marker in the last update packet to mark the end of the table content. The active switch recognizes the convergence when it receives the EOT marker, and it then begins sending updates. When the active switch has received all EOT markers from its neighbors or when the NSF converge timer expires, EIGRP notifies the routing information database (RIB) of convergence and floods its topology table to all NSF-aware peers.

EIGRP Stub Routing

The EIGRP stub routing feature improves network stability, reduces resource utilization, and simplifies the stub device configuration.

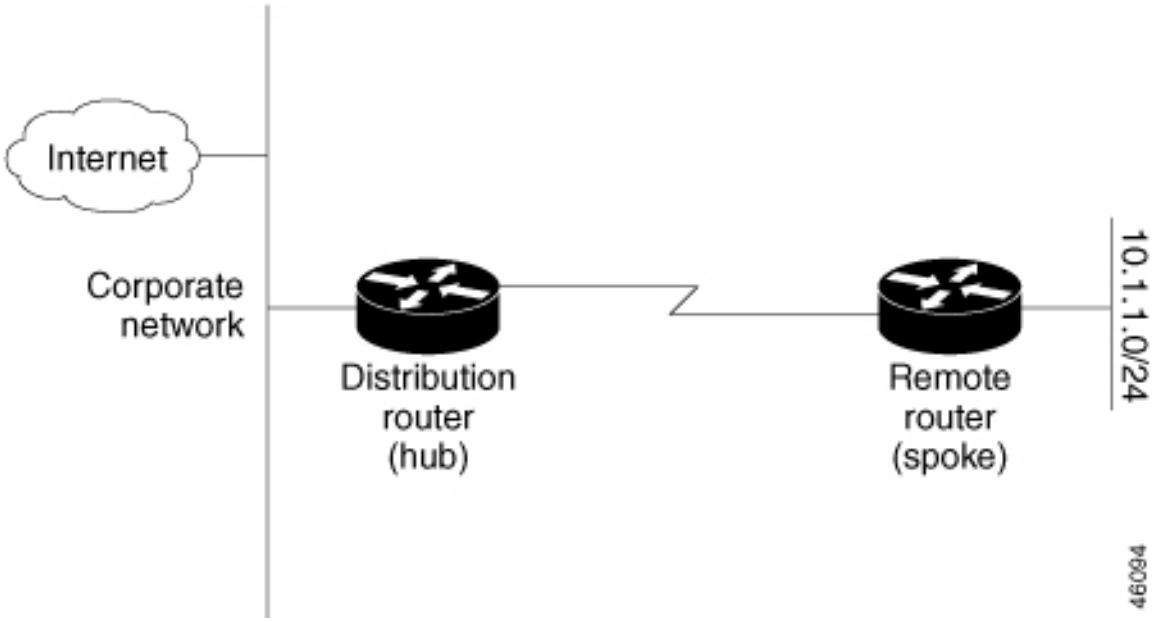
Stub routing is commonly used in hub-and-spoke network topologies. In a hub-and-spoke network, one or more end (stub) networks are connected to a remote device (the spoke) that is connected to one or more distribution devices (the hub). The remote device is adjacent to one or more distribution devices. The only route for IP traffic to reach the remote device is through a distribution device. This type of configuration is commonly used in WAN topologies, where the distribution device is directly connected to a WAN. The distribution device can be connected to many remote devices, which is often the case. In a hub-and-spoke topology, the remote device must forward all nonlocal traffic to a distribution device, so it becomes unnecessary for the remote device to have a complete routing table. Generally, the distribution device need not send anything more than a default route to the remote device.

When using the EIGRP stub routing feature, you need to configure the distribution and remote devices to use EIGRP and configure only the remote device as a stub. Only specified routes are propagated from the remote (stub) device. The stub device responds to all queries for summaries, connected routes, redistributed static routes, external routes, and internal routes with the message “inaccessible.” A device that is configured as a stub will send a special peer information packet to all neighboring devices to report its status as a stub device.

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

The figure below shows a simple hub-and-spoke network.

Figure 5: Simple Hub-and-Spoke Network



The stub routing feature by itself does not prevent routes from being advertised to the remote device. In the above example, the remote device can access the corporate network and the Internet only through the distribution device. Having a complete route table on the remote device would serve no functional purpose because the path to the corporate network and the Internet would always be through the distribution device. The large route table would only reduce the amount of memory required by the remote device. Bandwidth and memory

can be conserved by summarizing and filtering routes in the distribution device. The remote device need not receive routes that have been learned from other networks because the remote device must send all nonlocal traffic, regardless of the destination, to the distribution device. If a true stub network is desired, the distribution device should be configured to send only a default route to the remote device. The EIGRP stub routing feature does not automatically enable summarization on distribution devices. In most cases, the network administrator will need to configure summarization on distribution devices.

Note When configuring the distribution device to send only a default route to the remote device, you must use the **ip classless** command on the remote device. By default, the **ip classless** command is enabled in all Cisco images that support the EIGRP stub routing feature.

Without the EIGRP stub routing feature, even after routes that are sent from the distribution device to the remote device have been filtered or summarized, a problem might occur. If a route is lost somewhere in the corporate network, EIGRP could send a query to the distribution device, which in turn would send a query to the remote device, even if routes are being summarized. If there is a communication problem (over the WAN link) between the distribution device and the remote device, an EIGRP stuck in active (SIA) condition could occur and cause instability elsewhere in the network. The EIGRP stub routing feature allows a network administrator to prevent queries from being sent to the remote device.

EIGRPv6 Stub Routing

The EIGRPv6 stub routing feature, reduces resource utilization by moving routed traffic closer to the end user.

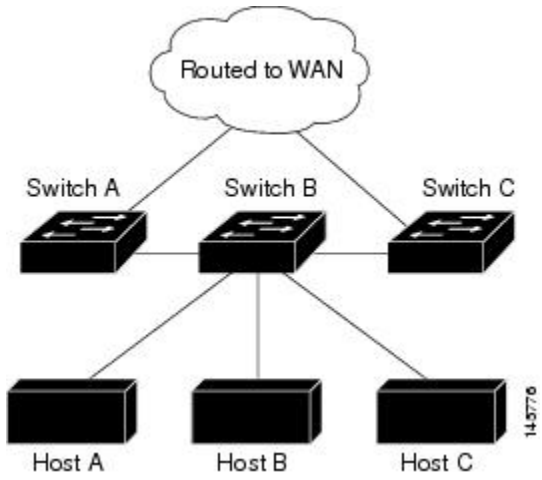
In a network using EIGRPv6 stub routing, the only allowable route for IPv6 traffic to the user is through a switch that is configured with EIGRPv6 stub routing. The switch sends the routed traffic to interfaces that are configured as user interfaces or are connected to other devices.

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.

In the figure given below, 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 6: 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*.

How to Configure EIGRP

To create an EIGRP routing process, you must enable EIGRP and associate networks. EIGRP sends updates to the interfaces in the specified networks. If you do not specify an interface network, it is not advertised in any EIGRP update.

Note If you have routers on your network that are configured for IGRP, and you want to change to EIGRP, you must designate transition routers that have both IGRP and EIGRP configured. In these cases, perform Steps 1 through 3 in the next section and also see the "Configuring Split Horizon" section. You must use the same AS number for routes to be automatically redistributed.

Default EIGRP Configuration

Table 15: Default EIGRP Configuration

Feature	Default Setting
Auto summary	Disabled.
Default-information	Exterior routes are accepted and default information is passed betwee processes when doing redistribution.

Feature	Default Setting
Default metric	Only connected routes and interface static routes can be redistribu default metric. The metric includes: • Bandwidth: 0 or greater kb/s. • Delay (tens of microseconds): 0 or any positive number that is 39.1 nanoseconds. • Reliability: any number between 0 and 255 (255 means 100 p reliability). • Loading: effective bandwidth as a number between 0 and 255 percent loading). • MTU: maximum transmission unit size of the route in bytes. 0 c integer.
Distance	Internal distance: 90. External distance: 170.
EIGRP log-neighbor changes	Disabled. No adjacency changes logged.
IP authentication key-chain	No authentication provided.
IP authentication mode	No authentication provided.
IP bandwidth-percent	50 percent.
IP hello interval	For low-speed nonbroadcast multiaccess (NBMA) networks: 60 sec networks: 5 seconds.
IP hold-time	For low-speed NBMA networks: 180 seconds; all other networks:
IP split-horizon	Enabled.
IP summary address	No summary aggregate addresses are predefined.
Metric weights	tos: 0; k1 and k3: 1; k2, k4, and k5: 0
Network	None specified.
Nonstop Forwarding (NSF) Awareness	Enabled for IPv4 on switches running the Network Advantage lice Layer 3 switches to continue forwarding packets from a neighboring router during hardware or software changes.
NSF capability	Disabled. **Note** The device supports EIGRP NSF-capable routing for
Offset-list	Disabled.
Router EIGRP	Disabled.
Set metric	No metric set in the route map.

Feature	Default Setting
Traffic-share	Distributed proportionately to the ratios of the metrics.
Variance	1 (equal-cost load-balancing).

Configuring Basic EIGRP Parameters

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device>enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device#configure terminal`	Enters global configuration mode.
Step 3	**router eigrp** autonomous-system **Example:** `Device(config)#router eigrp 10`	Enables an EIGRP routing process, and enter router configuration mode. The AS number identifies the routes to other EIGRP routers and is used to tag routing information.
Step 4	**nsf** **Example:** `Device(config-router)#nsf`	(Optional) Enables EIGRP NSF. Enter this command on the active switch and on all of its peers.
Step 5	**network** network-number **Example:** `Device(config-router)#network 192.168.0.0`	Associate networks with an EIGRP routing process. EIGRP sends updates to the interfaces in the specified networks.
Step 6	**eigrp log-neighbor-changes** **Example:** `Device(config-router)#eigrp log-neighbor-changes`	(Optional) Enables logging of EIGRP neighbor changes to monitor routing system stability.
Step 7	**metric weights** *tos k1 k2 k3 k4 k5* **Example:** `Device(config-router)#metric weights 0 2 0 2 0 0`	(Optional) Adjust the EIGRP metric. Although the defaults have been carefully set to provide excellent operation in most networks, you can adjust them.

	Command or Action	Purpose
		Caution Setting metrics is complex and is not recommended without guidance from an experienced network designer.
Step 8	**offset-list** [*access-list number* | *name*] {**in** | **out**} *offset* [*type number*] **Example:** `Device(config-router)#offset-list 21 out 10`	(Optional) Applies an offset list to routing metrics to increase incoming and outgoing metrics to routes learned through EIGRP. You can limit the offset list with an access list or an interface.
Step 9	**auto-summary** **Example:** `Device(config-router)#auto-summary`	(Optional) Enables automatic summarization of subnet routes into network-level routes.
Step 10	**interface** *interface-id* **Example:** `Device(config-router)#interface gigabitethernet 1/0/1`	Enters interface configuration mode, and specifies the Layer 3 interface to configure.
Step 11	**ip summary-address eigrp** *autonomous-system-number address mask* **Example:** `Device(config-if)#ip summary-address eigrp 1 192.168.0.0 255.255.0.0`	(Optional) Configures a summary aggregate.
Step 12	**end** **Example:** `Device(config-if)#end`	Returns to privileged EXEC mode.
Step 13	**show ip protocols** **Example:** `Device#show ip protocols`	Verifies your entries. For NSF awareness, the output shows: *** IP Routing is NSF aware *** EIGRP NSF enabled
Step 14	**copy running-config startup-config** **Example:** `Device#copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

Configuring EIGRP Interfaces

Other optional EIGRP parameters can be configured on an interface basis.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device>enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device#configure terminal`	Enters global configuration mode.
Step 3	**interface** *interface-id* **Example:** `Device(config)#interface gigabitethernet 1/0/1`	Enters interface configuration mode, and specifies the Layer 3 interface to configure.
Step 4	**ip bandwidth-percent eigrp** *percent* **Example:** `Device(config-if)#ip bandwidth-percent eigrp 60`	(Optional) Configures the percentage of bandwidth that can be used by EIGRP on an interface. The default is 50 percent.
Step 5	**ip summary-address eigrp** *autonomous-system-number address mask* **Example:** `Device(config-if)#ip summary-address eigrp 109 192.161.0.0 255.255.0.0`	(Optional) Configures a summary aggregate address for a specified interface (not usually necessary if auto-summary is enabled).
Step 6	**ip hello-interval eigrp** *autonomous-system-number seconds* **Example:** `Device(config-if)#ip hello-interval eigrp 109 10`	(Optional) Change the hello time interval for an EIGRP routing process. The range is 1 to 65535 seconds. The default is 60 seconds for low-speed NBMA networks and 5 seconds for all other networks.
Step 7	**ip hold-time eigrp** *autonomous-system-number seconds* **Example:** `Device(config-if)#ip hold-time eigrp 109 40`	(Optional) Change the hold time interval for an EIGRP routing process. The range is 1 to 65535 seconds. The default is 180 seconds for low-speed NBMA networks and 15 seconds for all other networks. **Caution** Do not adjust the hold time without consulting Cisco technical support.

	Command or Action	Purpose
Step 8	**no ip split-horizon eigrp** *autonomous-system-number* **Example:** `Device(config-if)#no ip split-horizon eigrp 109`	(Optional) Disables split horizon to allow route information to be advertised by a router out any interface from which that information originated.
Step 9	**end** **Example:** `Device(config)#end`	Returns to privileged EXEC mode.
Step 10	**show ip eigrp interface** **Example:** `Device#show ip eigrp interface`	Displays which interfaces EIGRP is active on and information about EIGRP relating to those interfaces.
Step 11	**copy running-config startup-config** **Example:** `Device#copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

Configuring EIGRP for IPv6

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 EIGRP Route Authentication

EIGRP route authentication provides MD5 authentication of routing updates from the EIGRP routing protocol to prevent the introduction of unauthorized or false routing messages from unapproved sources.

Procedure

	Command or Action	Purpose
Step 1	**enable**	Enables privileged EXEC mode.

	Command or Action	Purpose
	Example: `Device>enable`	• Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device#configure terminal`	Enters global configuration mode.
Step 3	**interface** interface-id **Example:** `Device(config)#interface gigabitethernet 1/0/1`	Enters interface configuration mode, and specifies the Layer 3 interface to configure.
Step 4	**ip authentication mode eigrp** *autonomous-system* **md5** **Example:** `Device(config-if)#ip authentication mode eigrp 104 md5`	Enables MD5 authentication in IP EIGRP packets.
Step 5	**ip authentication key-chain eigrp** *autonomous-system key-chain* **Example:** `Device(config-if)#ip authentication key-chain eigrp 105 chain1`	Enables authentication of IP EIGRP packets.
Step 6	**exit** **Example:** `Device(config-if)#exit`	Returns to global configuration mode.
Step 7	**key chain** *name-of-chain* **Example:** `Device(config)#key chain chain1`	Identify a key chain and enter key-chain configuration mode. Match the name configured in Step 4.
Step 8	**key** *number* **Example:** `Device(config-keychain)#key 1`	In key-chain configuration mode, identify the key number.
Step 9	**key-string** *text* **Example:** `Device(config-keychain-key)#key-string key1`	In key-chain key configuration mode, identify the key string.

	Command or Action	Purpose
Step 10	**accept-lifetime** *start-time* {**infinite** | *end-time* | **duration** *seconds*} **Example:** `Device(config-keychain-key)#accept-lifetime 13:30:00 Jan 25 2011 duration 7200`	(Optional) Specifies the time period during which the key can be received. The *start-time* and *end-time* syntax can be either *hh:mm:ss Month date year* or *hh:mm:ss date Month year*. The default is forever with the default *start-time* and the earliest acceptable date as January 1, 1993. The default *end-time* and **duration** is **infinite**.
Step 11	**send-lifetime** *start-time* {**infinite** | *end-time* | **duration** *seconds*} **Example:** `Device(config-keychain-key)#send-lifetime 14:00:00 Jan 25 2011 duration 3600`	(Optional) Specifies the time period during which the key can be sent. The *start-time* and *end-time* syntax can be either *hh:mm:ss Month date year* or *hh:mm:ss date Month year*. The default is forever with the default *start-time* and the earliest acceptable date as January 1, 1993. The default *end-time* and **duration** is **infinite**.
Step 12	**end** **Example:** `Device(config)#end`	Returns to privileged EXEC mode.
Step 13	**show key chain** **Example:** `Device#show key chain`	Displays authentication key information.
Step 14	**copy running-config startup-config** **Example:** `Device#copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

Monitoring and Maintaining EIGRP

You can delete neighbors from the neighbor table. You can also display various EIGRP routing statistics. The table given below lists the privileged EXEC commands for deleting neighbors and displaying statistics.

Table 16: IP EIGRP Clear and Show Commands

Command	Purpose
clear ip eigrp neighbors [*if-address* | *interface*]	Deletes neighbors from the neigh
show ip eigrp interface [*interface*] [*as number*]	Displays information about interf
show ip eigrp neighbors [*type-number*]	Displays EIGRP discovered neig

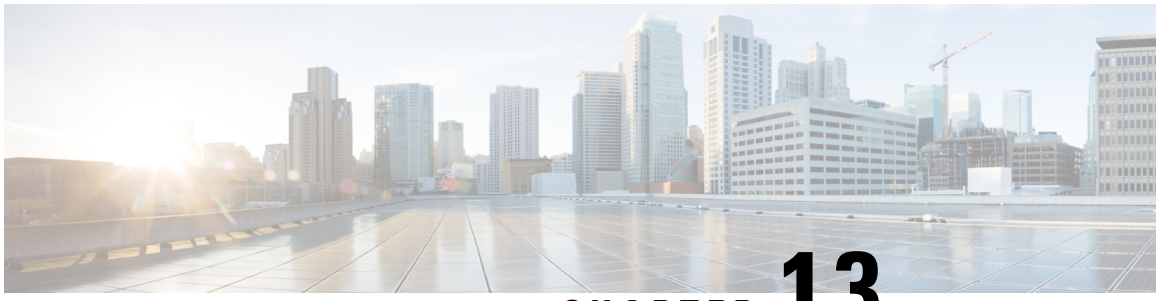
Command	Purpose
show ip eigrp topology [*autonomous-system-number*] | [[*ip-address*] *mask*]]	Displays the EIGRP topology table f
show ip eigrp traffic [*autonomous-system-number*]	Displays the number of packets sent process.

Feature Information for EIGRP

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Table 17: Feature Information for EIGRP Features

Releases	Feature Information
Cisco IOS XE Everest 16.5.1a	The feature was introduced.



CHAPTER 13

Configuring BGP

- Restrictions for BGP, on page 183
- Information About BGP, on page 183
- How to Configure BGP, on page 195
- Configuration Examples for BGP, on page 235
- Monitoring and Maintaining BGP, on page 238
- Feature Information for Border Gateway Protocol, on page 239

Restrictions for BGP

- The BGP hold time must always be configured higher than the Graceful Restart hold time on a device, even with Graceful Restart disabled. A peer device with an unsupported hold time can establish a session with a device through an open message, but once Graceful Restart is enabled the session will flap.
- Layer 3 forwarding is delayed until routing tables are populated on a device when you switch on the device or execute the **clear ip bgp** command.

Note The routing tables require around 80 seconds for population. You can use the **show ip bgp** *ip-address* command, in privileged EXEC mode, to check whether the routing tables are populated or not.

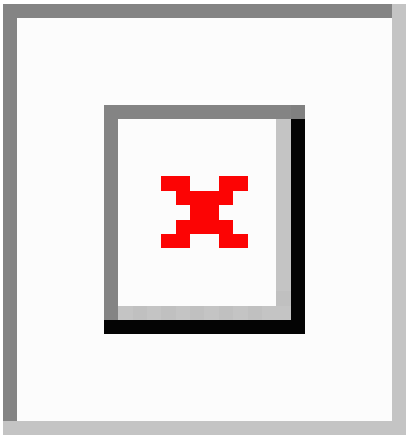
Information About BGP

The Border Gateway Protocol (BGP) is an exterior gateway protocol used to set up an interdomain routing system that guarantees the loop-free exchange of routing information between autonomous systems. Autonomous systems are made up of routers that operate under the same administration and that run Interior Gateway Protocols (IGPs), such as RIP or OSPF, within their boundaries and that interconnect by using an Exterior Gateway Protocol (EGP). BGP Version 4 is the standard EGP for interdomain routing in the Internet. The protocol is defined in RFCs 1163, 1267, and 1771.

BGP Network Topology

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). The figure given below shows a network that is running both EBGP and 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 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) as its transport protocol (specifically port 179). Two BGP speakers that have a TCP connection to each other for exchanging routing information are known as peers or neighbors. In the above figure, Routers A and B are BGP peers, as are Routers B and C and Routers C and D. 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.

The network has these characteristics:

- Routers A and B are running EBGP, and Routers B and C are running IBGP. Note that the EBGP peers are directly connected and that the IBGP peers are not. As long as there is an IGP running that allows the two neighbors to reach one another, IBGP peers do not have to be directly connected.
- 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.
- AS 200 is a transit AS for AS 100 and AS 300—that is, AS 200 is used to transfer packets between AS 100 and AS 300.

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).

In BGP, each route consists of a network number, a list of autonomous systems that information has passed through (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.

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. See the "Configuring BGP Decision Attributes" section for information about BGP attributes.

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

The BGP NSF Awareness feature is supported for IPv4 in the Network Advantage license.. 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 during the interval between the primary Route Processor (RP) in a router failing and the backup RP taking over, or while the primary RP is manually reloaded for a nondisruptive software upgrade.

Information About BGP Routing

To enable BGP routing, you establish a BGP routing process and define the local network. Because BGP must completely recognize the relationships with its neighbors, you must also specify a BGP neighbor.

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 to each other and share a subnet, but internal neighbors can be anywhere in the same AS.

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. The private AS numbers are 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.

If your AS will be passing traffic through it from another AS to a third AS, it is important to be consistent about the routes it advertises. If BGP advertised a route before all routers in the network had learned about the route through the IGP, the AS might receive traffic that some routers could not yet route. To prevent this from happening, BGP must wait until the IGP has propagated information across the AS so that BGP is synchronized with the IGP. 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 systems are running BGP, you can disable synchronization, which allows your network to carry fewer routes in the IGP and allows BGP to converge more quickly.

Routing Policy Changes

Routing policies for a peer include all the configurations that might affect inbound or outbound routing table updates. 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.

There are two types of reset, hard reset and soft reset. Cisco IOS Releases 12.1 and later support a soft reset without any prior configuration. 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.

- When soft reset generates inbound updates from a neighbor, it is called dynamic inbound soft reset.
- When soft reset sends a set of updates to a neighbor, it is called outbound soft reset.

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.

The table given below lists the advantages and disadvantages hard reset and soft reset.

Table 18: 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 t provided by the neighbor are lost. Not recommended.
Outbound soft reset	No configuration, no storing of routing table updates	Does not reset inbound routing table u
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 ro capability (in Cisco IOS Release 12.1

BGP Decision Attributes

When a BGP speaker receives updates from multiple autonomous systems that describe different paths to the same destination, it must choose the single best path for reaching that destination. When chosen, the selected path is entered into the BGP routing table and propagated to its neighbors. The decision is based on the value of attributes that the update contains and other BGP-configurable factors.

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 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.
5. Prefer the route with the shortest AS path.
6. 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.
7. 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.
8. Prefer the external (EBGP) path over the internal (IBGP) path.
9. 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).
10. If the following 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.
11. 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

Within BGP, route maps can be used to control and to modify routing information and to define the conditions by which routes are redistributed between routing domains. Each route map has a name that identifies the route map (*map tag*) and an optional sequence number.

BGP Filtering

You can filter BGP advertisements by using AS-path filters, such as the **as-path access-list** global configuration command and the **neighbor filter-list** router configuration command. You can also use access lists with the **neighbor distribute-list** router configuration command. Distribute-list filters are applied to network numbers. See the “Controlling Advertising and Processing in Routing Updates” section for information about the **distribute-list** command.

You can use route maps on a per-neighbor basis to filter updates and to modify various attributes. A route map 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 requires the **match as-path access-list** route-map command, community based matching requires the **match community-list** route-map command, and network-based matching requires the **ip access-list** global configuration command.

Prefix List for BGP Filtering

You can use prefix lists as an alternative to access lists in many BGP route filtering commands, including the **neighbor distribute-list** router configuration command. The advantages of using prefix lists include performance improvements in loading and lookup of large lists, incremental update support, easier CLI configuration, and greater flexibility.

Filtering by a prefix list involves matching the prefixes of routes with those listed in the prefix list, as when matching access lists. When there is a match, the route is used. Whether a prefix is permitted or denied is based upon 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 very large increments, you might run out of values.

BGP Community Filtering

One way that BGP controls the distribution of routing information based on the value of the COMMUNITIES attribute. The attribute is a way to groups destinations into communities and to apply routing decisions based on the communities. This method simplifies configuration of a BGP speaker to control distribution of routing information.

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. 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

Often many BGP neighbors are configured with the same update policies (that is, the same outbound route maps, distribute lists, filter lists, update source, and so on). Neighbors with the same update policies can be grouped into peer groups to simplify configuration and to make updating more efficient. When you have configured many peers, we recommend this approach.

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 the configuration options of the peer group, including the 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 made to the peer group. Members can also be configured to override the options that do not affect outbound updates.

Aggregate Routes

Classless interdomain routing (CIDR) enables you to create aggregate routes (or supernets) to minimize the size of routing tables. You can configure aggregate routes in BGP either by redistributing an aggregate route into BGP or by creating an aggregate entry in the BGP routing table. An aggregate address is added to the BGP table when there is at least one more specific entry in the BGP table.

Routing Domain Confederations

One way to reduce the IBGP mesh is to divide an autonomous system into multiple subautonomous systems and to group them into a single confederation that appears as a single autonomous system. Each autonomous system is fully meshed within itself and has a few connections to other autonomous systems in the same confederation. Even though the peers in different autonomous systems have EBGP sessions, they exchange routing information as if they were IBGP peers. Specifically, the next hop, MED, and local preference information is preserved. You can then use a single IGP for all of the autonomous systems.

BGP Route Reflectors

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 IBPG speakers must be connected. The internal neighbors do not send routes learned from internal neighbors to other internal neighbors.

With route reflectors, all IBGP speakers need not be fully meshed because another method is used 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 (all the other routers in the autonomous system). A 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, but the client peers need not 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.

Usually a cluster of clients have a single route reflector, and 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 so that a route reflector can recognize updates from route reflectors in the same cluster. All the route reflectors serving a cluster should be fully meshed and should have identical sets of client and nonclient peers.

Route Dampening

Route flap dampening is a BGP feature designed to minimize the propagation of flapping routes across an internetwork. A route is considered to be flapping when it is repeatedly available, then unavailable, then available, then unavailable, and so on. When route dampening is enabled, a numeric penalty value is assigned to a route when it flaps. When a route's accumulated penalties reach a configurable limit, BGP suppresses advertisements of the route, even if the route is running. The reuse limit is a configurable value that is compared with the penalty. If the penalty is less than the reuse limit, a suppressed route that is up is advertised again.

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

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 because a prefix that represents multiple addresses or hosts over a large topological area cannot be accurately reflected in a single route. 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 allows you to originate a prefix into a BGP routing table without the corresponding match. This feature allows more specific routes to be generated based on administrative policy or traffic engineering information in order to provide more specific control over the forwarding of packets to these more specific routes, which are injected into the BGP routing table only if the configured conditions are met. Enabling this feature will allow you to improve the accuracy of common route aggregation by conditionally injecting or replacing less specific prefixes with more specific prefixes. Only prefixes that are equal to or more specific than the original prefix may be injected. BGP conditional route injection is enabled with the **bgp inject-map exist-map**command and 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 matched will be used.

BGP Peer Templates

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.

A peer template is a configuration pattern that can be applied to neighbors that share policies. Peer templates are reusable and support inheritance, which allows the network operator to group and apply distinct neighbor configurations for BGP neighbors that share policies. Peer templates also allow the network operator to define very complex configuration patterns through the capability of a peer template to inherit a configuration from another peer template.

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.

Peer templates improve the flexibility and enhance the capability of neighbor configuration. Peer templates also provide an alternative to peer group configuration and overcome some limitations of peer groups. BGP peer devices using peer templates also benefit from automatic update group configuration. With the configuration of the BGP peer templates and the support of the BGP dynamic update peer groups, the network operator no longer needs to configure peer groups in BGP and the network can benefit 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.

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. 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

The inheritance capability is a key component of peer template operation. Inheritance in a peer template is similar to node and tree structures commonly found in general computing, for example, file and directory trees. A peer template can directly or indirectly inherit the configuration from another peer template. The directly inherited peer template represents the tree in the structure. The indirectly inherited peer template represents a node in 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.

This structure eliminates the need to repeat configuration statements that are commonly reapplied to groups of neighbors because common configuration statements can be applied once and then indirectly inherited by peer templates that are applied to neighbor groups with common configurations. 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 will overwrite any indirectly inherited statements that are duplicated in the directly inherited template.

Inheritance expands the scalability and flexibility of neighbor configuration by allowing you to chain together peer templates configurations to create simple configurations that inherit common configuration statements or complex configurations that apply very specific configuration statements along with common inherited configurations. Specific details about configuring inheritance in peer session templates and peer policy templates are provided in the following sections.

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 associated with a specific template.

Peer Session Templates

Peer session templates are used to group and apply the configuration of general session commands to groups of neighbors that share session configuration elements. General session commands that are common for neighbors that are configured in different address families can be configured within the same peer session template. Peer session templates are created and configured in peer session configuration mode. Only general session commands can be configured in a peer session template. The following general session commands are supported by peer session templates:

- **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. The configuration of peer session templates simplifies the configuration of general session commands that are commonly applied to all neighbors within an autonomous system.

Peer session templates support direct and indirect inheritance. A peer can be configured with only one peer session template at a time, and 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 a maximum of eight peer session configurations to a neighbor: the configuration from the directly inherited peer session template and the configurations from up to seven indirectly inherited peer session templates. Inherited peer session configurations are evaluated first and 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 will have 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. The following examples illustrate the use of this feature.

In the following 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 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.

Peer Policy Templates

Peer policy templates are used to group and apply the configuration of commands that are applied within specific address families and NLRI configuration mode. Peer policy templates are created and configured in peer policy configuration mode. BGP policy commands that are configured for specific address families are configured in a peer policy template. The following BGP policy commands are supported by peer policy templates:

- **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 are used to configure BGP policy commands that are configured for neighbors that belong to specific address families. Like peer session templates, peer policy templates are configured once and then applied to many neighbors through the direct application of a peer policy template or through inheritance from peer policy templates. The configuration of peer policy templates simplifies the configuration of BGP policy commands that are applied to all neighbors within an autonomous system.

Like a peer session template, a peer policy template supports inheritance. However, there are minor differences. 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 with the lowest sequence number and ending with the highest sequence number. However, there is a difference; a peer policy template will not collapse like a route map. Every sequence is evaluated, and if a BGP policy command is reapplied with a different value, it will overwrite any previous value from a lower sequence number.

The directly applied peer policy template and the **inherit peer-policy** statement with the highest sequence number will always have priority and be 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.

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.

BGP Route Map Next Hop Self

The BGP Route Map Next Hop Self feature provides a way to override the settings for bgp next-hop unchanged and bgp next-hop unchanged allpath selectively. These settings are global for an address family. For some routes this may not be appropriate. For example, static routes may need to be redistributed with a next hop of self, but connected routes and routes learned via Interior Border Gateway Protocol (IBGP) or Exterior Border Gateway Protocol (EBGP) may continue to be redistributed with an unchanged next hop.

The BGP route map next hop self functionality modifies the existing route map infrastructure to configure a new ip next-hop self setting, which overrides the bgp next-hop unchanged and bgp next-hop unchanged allpaths settings.

The ip next-hop self setting is applicable only to VPNv4 and VPNv6 address families. Routes distributed by protocols other than BGP are not affected.

You configure a new bgp route-map priority setting to inform BGP that the route map will take priority over the settings for bgp next-hop unchanged and bgp next-hop unchanged allpath. The bgp route-map priority setting only impacts BGP. The bgp route-map priority setting has no impact unless you configure the bgp next-hop unchanged or bgp next-hop unchanged allpaths settings.

How to Configure BGP

The following sections provide configurational information about BGP.

Default BGP Configuration

The table given below shows the basic default BGP configuration.

Table 19: 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 s from external BGP peers. • Compare router ID: Disabled.
BGP community list	• Number: None defined. When you permit a value for the community nu defaults to an implicit deny for everything else that has not been permitt • Format: Cisco default format (32-bit number).
BGP confederation identifier/peers	• Identifier: None configured. • Peers: None identified.

Feature	Default Setting
BGP Fast external fallover	Enabled.
BGP local preference	100. The range is 0 to 4294967295 with the higher value preferred.
BGP network	None specified; no backdoor route advertised.
BGP route dampening	Disabled by default. When enabled: • Half-life is 15 minutes. • Re-use is 750 (10-second increments). • Suppress is 2000 (10-second increments). • Max-suppress-time is 4 times half-life; 60 minutes.
BGP router ID	The IP address of a loopback interface if one is configured or the highest IP address for a physical interface on the router.
Default information originate (protocol or network redistribution)	Disabled.
Default metric	Built-in, automatic metric translations.
Distance	• External route administrative distance: 20 (acceptable values are from 1 to 2 • Internal route administrative distance: 200 (acceptable values are from 1 to • Local route administrative distance: 200 (acceptable values are from 1 to 25
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 neighb 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 inte • 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
NSF[1] Awareness	Disabled[2]. If enabled, allows Layer 3 switches to continue forwarding packet 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.

1 Nonstop Forwarding
2 NSF Awareness can be enabled for IPv4 on switches with the Network Advantage license by enabling Graceful Restart.

Enabling BGP Routing

Before you begin

Note To enable BGP, the standalone switch or active switch must be running the Network Advantage license.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**ip routing** **Example:** `Device(config)# ip routing`	Enables IP routing.
Step 4	**router bgp** *autonomous-system* **Example:** `Device(config)# router bgp 45000`	Enables a BGP routing process, assign it an AS number, and enter router configuration mode. The AS number can be from 1 to 65535, with 64512 to 65535 designated as private autonomous numbers.
Step 5	**network** *network-number* [**mask** *network-mask*] [**route-map** *route-map-name*] **Example:** `Device(config-router)# network 10.108.0.0`	Configures a network as local to this AS, and enter it in the BGP table.
Step 6	**neighbor** {*ip-address* | *peer-group-name*} **remote-as** *number* **Example:** `Device(config-router)# neighbor 10.108.1.2 remote-as 65200`	Adds an entry to the BGP neighbor table specifying that the neighbor identified by the IP address belongs to the specified AS. 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.

	Command or Action	Purpose
Step 7	**neighbor** {*ip-address* | *peer-group-name*} **remove-private-as** **Example:** `Device(config-router)# neighbor 172.16.2.33 remove-private-as`	(Optional) Removes private AS numbers from the AS-path in outbound routing updates.
Step 8	**synchronization** **Example:** `Device(config-router)# synchronization`	(Optional) Enables synchronization between BGP and an IGP.
Step 9	**auto-summary** **Example:** `Device(config-router)# auto-summary`	(Optional) Enables automatic network summarization. When a subnet is redistributed from an IGP into BGP, only the network route is inserted into the BGP table.
Step 10	**bgp graceful-restart** **Example:** `Device(config-router)# bgp graceful-start`	(Optional) Enables NSF awareness on switch. By default, NSF awareness is disabled.
Step 11	**end** **Example:** `Device(config-router)# end`	Returns to privileged EXEC mode.
Step 12	**show ip bgp network** *network-number* **Example:** `Device# show ip bgp network 10.108.0.0`	Verifies the configuration.
Step 13	**show ip bgp neighbor** **Example:** `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*
Step 14	**copy running-config startup-config** **Example:** `Device# copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

Managing Routing Policy Changes

To learn if a BGP peer supports the route refresh capability and to reset the BGP session:

Procedure

	Command or Action	Purpose
Step 1	**show ip bgp neighbors** **Example:** `Device# show ip bgp neighbors`	Displays whether a neighbor supports the route refresh capability. When supported, this message appears for the router: *Received route refresh capability from peer.*
Step 2	**clear ip bgp** {***** | *address* | *peer-group-name*} **Example:** `Device# clear ip bgp *`	Resets the routing table on the specified connection. • 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.
Step 3	**clear ip bgp** {***** | *address* | *peer-group-name*} **soft out** **Example:** `Device# clear ip bgp * soft out`	(Optional) Performs an outbound soft reset to reset the inbound routing table on the specified connection. 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.
Step 4	**show ip bgp** **Example:** `Device# show ip bgp`	Verifies the reset by checking information about the routing table and about BGP neighbors.
Step 5	**show ip bgp neighbors** **Example:** `Device# show ip bgp neighbors`	Verifies the reset by checking information about the routing table and about BGP neighbors.

Configuring BGP Decision Attributes

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**router bgp** *autonomous-system* **Example:** `Device(config)# router bgp 4500`	Enables a BGP routing process, assign it an AS number, and enter router configuration mode.
Step 4	**bgp best-path as-path ignore** **Example:** `Device(config-router)# bgp bestpath as-path ignore`	(Optional) Configures the router to ignore AS path length in selecting a route.
Step 5	**neighbor** {*ip-address* | *peer-group-name*} **next-hop-self** **Example:** `Device(config-router)# neighbor 10.108.1.1 next-hop-self`	(Optional) Disables next-hop processing on BGP updates to a neighbor by entering a specific IP address to be used instead of the next-hop address.
Step 6	**neighbor** {*ip-address* | *peer-group-name*} **weight** *weight* **Example:** `Device(config-router)# neighbor 172.16.12.1 weight 50`	(Optional) Assign a weight to a neighbor connection. 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.
Step 7	**default-metric** *number* **Example:** `Device(config-router)# default-metric 300`	(Optional) Sets a MED metric to set preferred paths to external neighbors. All routes without a MED will also be set to this value. The range is 1 to 4294967295. The lowest value is the most desirable.
Step 8	**bgp bestpath med missing-as-worst** **Example:** `Device(config-router)# bgp bestpath med missing-as-worst`	(Optional) Configures the switch to consider a missing MED as having a value of infinity, making the path without a MED value the least desirable path.

	Command or Action	Purpose
Step 9	**bgp always-compare med** **Example:** `Device(config-router)# bgp always-compare-med`	(Optional) Configures the switch to compare MEDs for paths from neighbors in different autonomous systems. By default, MED comparison is only done among paths in the same AS.
Step 10	**bgp bestpath med confed** **Example:** `Device(config-router)# bgp bestpath med confed`	(Optional) Configures the switch to consider the MED in choosing a path from among those advertised by different subautonomous systems within a confederation.
Step 11	**bgp deterministic med** **Example:** `Device(config-router)# bgp deterministic med`	(Optional) Configures the switch to consider the MED variable when choosing among routes advertised by different peers in the same AS.
Step 12	**bgp default local-preference** *value* **Example:** `Device(config-router)# bgp default local-preference 200`	(Optional) Change the default local preference value. The range is 0 to 4294967295; the default value is 100. The highest local preference value is preferred.
Step 13	**maximum-paths** *number* **Example:** `Device(config-router)# maximum-paths 8`	(Optional) Configures the number of paths to be added to the IP routing table. The default is to only enter the best path in the routing table. The range is from 1 to 16. Having multiple paths allows load-balancing among the paths. (Although the switch software allows a maximum of 32 equal-cost routes, the switch hardware will never use more than 16 paths per route.)
Step 14	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode.
Step 15	**show ip bgp** **Example:** `Device# show ip bgp`	Verifies the reset by checking information about the routing table and about BGP neighbors.
Step 16	**show ip bgp neighbors** **Example:** `Device# show ip bgp neighbors`	Verifies the reset by checking information about the routing table and about BGP neighbors.
Step 17	**copy running-config startup-config** **Example:**	(Optional) Saves your entries in the configuration file.

	Command or Action	Purpose
	`Device# copy running-config startup-config`	

Configuring BGP Filtering with Route Maps

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**route-map** *map-tag* [**permit** | **deny**] [*sequence-number*] **Example:** `Device(config)# route-map set-peer-address permit 10`	Creates a route map, and enter route-map configuration mode.
Step 4	**set ip next-hop** *ip-address* [*...ip-address*] [**peer-address**] **Example:** `Device(config)# set ip next-hop 10.1.1.3`	(Optional) Sets a route map to disable next-hop processing • 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.
Step 5	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode.
Step 6	**show route-map** [*map-name*] **Example:** `Device# show route-map`	Displays all route maps configured or only the one specified to verify configuration.

	Command or Action	Purpose
Step 7	**copy running-config startup-config** **Example:** `Device# copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

Configuring BGP Filtering by Neighbor

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**router bgp** *autonomous-system* **Example:** `Device(config)# router bgp 109`	Enables a BGP routing process, assign it an AS number, and enter router configuration mode.
Step 4	**neighbor** {*ip-address* | *peer-group name*} **distribute-list** {**access-list-number** | *name*} {**in** | **out**} **Example:** `Device(config-router)# neighbor 172.16.4.1 distribute-list 39 in`	(Optional) Filters BGP routing updates to or from neighbors as specified in an access list. **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.
Step 5	**neighbor** {*ip-address* | *peer-group name*} **route-map** *map-tag* {**in** | **out**} **Example:** `Device(config-router)# neighbor 172.16.70.24 route-map internal-map in`	(Optional) Applies a route map to filter an incoming or outgoing route.
Step 6	**end** **Example:**	Returns to privileged EXEC mode.

	Command or Action	Purpose
	`Device(config)# end`	
Step 7	**show ip bgp neighbors** **Example:** `Device# show ip bgp neighbors`	Verifies the configuration.
Step 8	**copy running-config startup-config** **Example:** `Device# copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

Configuring BGP Filtering by Access Lists and 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.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**ip as-path access-list** *access-list-number* {**permit** | **deny**} *as-regular-expressions* **Example:** `Device(config)# ip as-path access-list 1 deny _65535_`	Defines a BGP-related access list.
Step 4	**router bgp** *autonomous-system* **Example:** `Device(config)# router bgp 110`	Enters BGP router configuration mode.

	Command or Action	Purpose
Step 5	**neighbor** {*ip-address* | *peer-group name*} **filter-list** {*access-list-number* | *name*} {**in** | **out** | **weight** *weight*} **Example:** `Device(config-router)# neighbor 172.16.1.1 filter-list 1 out`	Establishes a BGP filter based on an access list.
Step 6	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode.
Step 7	**show ip bgp neighbors** [**paths** *regular-expression*] **Example:** `Device# show ip bgp neighbors`	Verifies the configuration.
Step 8	**copy running-config startup-config** **Example:** `Device# copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

Configuring Prefix Lists for BGP Filtering

You do not need to specify a sequence number when removing a configuration entry. **Show** commands include the sequence numbers in their output.

Before using a prefix list in a command, you must set up the prefix list.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.

	Command or Action	Purpose
Step 3	**ip prefix-list** *list-name* [**seq** *seq-value*] **deny** | **permit** *network/len* [**ge** *ge-value*] [**le** *le-value*] **Example:** `Device(config)# ip prefix-list BLUE permit 172.16.1.0/24`	Creates a prefix list with an optional sequence number to **deny** or **permit** access for matching conditions. You must enter at least one **permit** or **deny** clause. • *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
Step 4	**ip prefix-list** *list-name* **seq** *seq-value* **deny** | **permit** *network/len* [**ge** *ge-value*] [**le** *le-value*] **Example:** `Device(config)# ip prefix-list BLUE seq 10 permit 172.24.1.0/24`	(Optional) Adds an entry to a prefix list, and assign a sequence number to the entry.
Step 5	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode.
Step 6	**show ip prefix list** [**detail** | **summary**] *name* [*network/len*] [**seq** *seq-num*] [**longer**] [**first-match**] **Example:** `Device# show ip prefix list summary test`	Verifies the configuration by displaying information about a prefix list or prefix list entries.
Step 7	**copy running-config startup-config** **Example:** `Device# copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

Configuring BGP Community Filtering

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.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **ip community-list** *community-list-number* {**permit** | **deny**} *community-number*

4. **router bgp** *autonomous-system*
5. **neighbor** {*ip-address* | *peer-group name*} **send-community**
6. **set comm-list** *list-num* **delete**
7. **exit**
8. **ip bgp-community new-format**
9. **end**
10. **show ip bgp community**
11. **copy running-config startup-config**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**ip community-list** *community-list-number* {**permit** | **deny**} *community-number* **Example:** `Device(config)# ip community-list 1 permit 50000:10`	Creates a community list, and assigns it a number. • 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.
Step 4	**router bgp** *autonomous-system* **Example:** `Device(config)# router bgp 108`	Enters BGP router configuration mode.
Step 5	**neighbor** {*ip-address* | *peer-group name*} **send-community** **Example:** `Device(config-router)# neighbor 172.16.70.23 send-community`	Specifies that the COMMUNITIES attribute be sent to the neighbor at this IP address.
Step 6	**set comm-list** *list-num* **delete** **Example:** `Device(config-router)# set comm-list 500 delete`	(Optional) Removes communities from the community attribute of an inbound or outbound update that match a standard or extended community list specified by a route map.

	Command or Action	Purpose
Step 7	**exit** **Example:** `Device(config-router)# end`	Returns to global configuration mode.
Step 8	**ip bgp-community new-format** **Example:** `Device(config)# ip bgp-community new format`	(Optional) Displays and parses BGP communities in the format AA:NN. A BGP community is displayed in a two-part format 2 bytes long. The Cisco default community format is in the format NNAA. In the most recent RFC for BGP, a community takes the form AA:NN, where the first part is the AS number and the second part is a 2-byte number.
Step 9	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode.
Step 10	**show ip bgp community** **Example:** `Device# show ip bgp community`	Verifies the configuration.
Step 11	**copy running-config startup-config** **Example:** `Device# copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

Configuring BGP Neighbors and Peer Groups

To assign configuration options to an individual neighbor, specify any of these router configuration commands by using the neighbor IP address. To assign the options to a peer group, specify any of the commands by using the peer group name. You can disable a BGP peer or peer group without removing all the configuration information by using the **neighbor shutdown** router configuration command.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password if prompted.

	Command or Action	Purpose
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**router bgp** *autonomous-system*	Enters BGP router configuration mode.
Step 4	**neighbor** *peer-group-name* **peer-group**	Creates a BGP peer group.
Step 5	**neighbor** *ip-address* **peer-group** *peer-group-name*	Makes a BGP neighbor a member of the peer group.
Step 6	**neighbor** {*ip-address* | *peer-group-name*} **remote-as** *number*	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.
Step 7	**neighbor** {*ip-address* | *peer-group-name*} **description** *text*	(Optional) Associates a description with a neighbor.
Step 8	**neighbor** {*ip-address* | *peer-group-name*} **default-originate** [**route-map** *map-name*]	(Optional) Allows a BGP speaker (the local router) to send the default route 0.0.0.0 to a neighbor for use as a default route.
Step 9	**neighbor** {*ip-address* | *peer-group-name*} **send-community**	(Optional) Specifies that the COMMUNITIES attribute be sent to the neighbor at this IP address.
Step 10	**neighbor** {*ip-address* | *peer-group-name*} **update-source** *interface*	(Optional) Allows internal BGP sessions to use any operational interface for TCP connections.
Step 11	**neighbor** {*ip-address* | *peer-group-name*} **ebgp-multihop**	(Optional) Allows BGP sessions, even when the neighbor is not on a directly connected segment. The multihop session is not established if the only route to the multihop peer's address is the default route (0.0.0.0).
Step 12	**neighbor** {*ip-address* | *peer-group-name*} **local-as** *number*	(Optional) Specifies an AS number to use as the local AS. The range is 1 to 65535.
Step 13	**neighbor** {*ip-address* | *peer-group-name*} **advertisement-interval** *seconds*	(Optional) Sets the minimum interval between sending BGP routing updates.
Step 14	**neighbor** {*ip-address* | *peer-group-name*} **maximum-prefix** *maximum* [*threshold*]	(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.
Step 15	**neighbor** {*ip-address* | *peer-group-name*} **next-hop-self**	(Optional) Disables next-hop processing on the BGP updates to a neighbor.
Step 16	neighbor {*ip-address* | *peer-group-name*} password *string*	(Optional) Sets MD5 authentication on a TCP connection to a BGP peer. The same password must be configured on

	Command or Action	Purpose
		both BGP peers, or the connection between them is not made.
Step 17	**neighbor** {*ip-address* | *peer-group-name*} **route-map** *map-name* {**in** | **out**}	(Optional) Applies a route map to incoming or outgoing routes.
Step 18	**neighbor** {*ip-address* | *peer-group-name*} **send-community**	(Optional) Specifies that the COMMUNITIES attribute be sent to the neighbor at this IP address.
Step 19	**neighbor** {*ip-address* | *peer-group-name*} **timers** *keepalive holdtime*	(Optional) Sets timers for the neighbor or peer group. • 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.
Step 20	**neighbor** {*ip-address* | *peer-group-name*} **weight** *weight*	(Optional) Specifies a weight for all routes from a neighbor.
Step 21	**neighbor** {*ip-address* | *peer-group-name*} **distribute-list** {*access-list-number* | *name*} {**in** | **out**}	(Optional) Filter BGP routing updates to or from neighbors, as specified in an access list.
Step 22	**neighbor** {*ip-address* | *peer-group-name*} **filter-list** *access-list-number* {**in** | **out** | **weight** *weight*}	(Optional) Establish a BGP filter.
Step 23	**neighbor** {*ip-address* | *peer-group-name*} **version** *value*	(Optional) Specifies the BGP version to use when communicating with a neighbor.
Step 24	**neighbor** {*ip-address* | *peer-group-name*} **soft-reconfiguration inbound**	(Optional) Configures the software to start storing received updates.
Step 25	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode.
Step 26	**show ip bgp neighbors**	Verifies the configuration.
Step 27	**copy running-config startup-config** **Example:** `Device# copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

Configuring Aggregate Addresses in a Routing Table

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**router bgp** *autonomous-system* **Example:** `Device(config)# router bgp 106`	Enters BGP router configuration mode.
Step 4	**aggregate-address** *address mask* **Example:** `Device(config-router)# aggregate-address 10.0.0.0 255.0.0.0`	Creates an aggregate entry in the BGP routing table. The aggregate route is advertised as coming from the AS, and the atomic aggregate attribute is set to indicate that information might be missing.
Step 5	**aggregate-address** *address mask* **as-set** **Example:** `Device(config-router)# aggregate-address 10.0.0.0 255.0.0.0 as-set`	(Optional) Generates AS set path information. This command creates an aggregate entry following the same rules as the previous command, but the advertised path will be an AS_SET consisting of all elements contained in all paths. Do not use this keyword when aggregating many paths because this route must be continually withdrawn and updated.
Step 6	**aggregate-address** *address-mask* **summary-only** **Example:** `Device(config-router)# aggregate-address 10.0.0.0 255.0.0.0 summary-only`	(Optional) Advertises summary addresses only.
Step 7	**aggregate-address** *address mask* **suppress-map** *map-name* **Example:** `Device(config-router)# aggregate-address 10.0.0.0 255.0.0.0 suppress-map map1`	(Optional) Suppresses selected, more specific routes.

	Command or Action	Purpose
Step 8	**aggregate-address** *address mask* **advertise-map** *map-name* **Example:** `Device(config-router)# aggregate-address 10.0.0.0 255.0.0.0 advertise-map map2`	(Optional) Generates an aggregate based on conditions specified by the route map.
Step 9	**aggregate-address** *address mask* **attribute-map** *map-name* **Example:** `Device(config-router)# aggregate-address 10.0.0.0 255.0.0.0 attribute-map map3`	(Optional) Generates an aggregate with attributes specified in the route map.
Step 10	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode.
Step 11	**show ip bgp neighbors** [**advertised-routes**] **Example:** `Device# show ip bgp neighbors`	Verifies the configuration.
Step 12	**copy running-config startup-config** **Example:** `Device# copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

Configuring Routing Domain Confederations

You must specify a confederation identifier that acts as the autonomous system number for the group of autonomous systems.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**configure terminal** **Example:**	Enters global configuration mode.

	Command or Action	Purpose
	`Device# configure terminal`	
Step 3	**router bgp** *autonomous-system* **Example:** `Device(config)# router bgp 100`	Enters BGP router configuration mode.
Step 4	**bgp confederation identifier** *autonomous-system* **Example:** `Device(config)# bgp confederation identifier 50007`	Configures a BGP confederation identifier.
Step 5	**bgp confederation peers** *autonomous-system* [*autonomous-system ...*] **Example:** `Device(config)# bgp confederation peers 51000 51001 51002`	Specifies the autonomous systems that belong to the confederation and that will be treated as special EBGP peers.
Step 6	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode.
Step 7	**show ip bgp neighbor** **Example:** `Device# show ip bgp neighbor`	Verifies the configuration.
Step 8	**show ip bgp network** **Example:** `Device# show ip bgp network`	Verifies the configuration.
Step 9	**copy running-config startup-config** **Example:** `Device# copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

Configuring BGP Route Reflectors

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**router bgp** *autonomous-system* **Example:** `Device(config)# router bgp 101`	Enters BGP router configuration mode.
Step 4	**neighbor** {*ip-address* | *peer-group-name*} **route-reflector-client** **Example:** `Device(config-router)# neighbor 172.16.70.24 route-reflector-client`	Configures the local router as a BGP route reflector and the specified neighbor as a client.
Step 5	**bgp cluster-id** *cluster-id* **Example:** `Device(config-router)# bgp cluster-id 10.0.1.2`	(Optional) Configures the cluster ID if the cluster has more than one route reflector.
Step 6	**no bgp client-to-client reflection** **Example:** `Device(config-router)# no bgp client-to-client reflection`	(Optional) Disables client-to-client route 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.
Step 7	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode.
Step 8	**show ip bgp** **Example:** `Device# show ip bgp`	Verifies the configuration. Displays the originator ID and the cluster-list attributes.

	Command or Action	Purpose
Step 9	**copy running-config startup-config** **Example:** `Device# copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

Configuring Route Dampening

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**router bgp** *autonomous-system* **Example:** `Device(config)# router bgp 100`	Enters BGP router configuration mode.
Step 4	**bgp dampening** **Example:** `Device(config-router)# bgp dampening`	Enables BGP route dampening.
Step 5	**bgp dampening** *half-life reuse suppress max-suppress* [**route-map** *map*] **Example:** `Device(config-router)# bgp dampening 30 1500 10000 120`	(Optional) Changes the default values of route dampening factors.
Step 6	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode.

	Command or Action	Purpose
Step 7	**show ip bgp flap-statistics** [{**regexp** *regexp*} | {**filter-list** *list*} | {*address mask* [**longer-prefix**]}] **Example:** `Device# show ip bgp flap-statistics`	(Optional) Monitors the flaps of all paths that are flapping. The statistics are deleted when the route is not suppressed and is stable.
Step 8	**show ip bgp dampened-paths** **Example:** `Device# show pi bgp dampened-paths`	(Optional) Displays the dampened routes, including the time remaining before they are suppressed.
Step 9	**clear ip bgp flap-statistics** [{**regexp** *regexp*} | {**filter-list** *list*} | {*address mask* [**longer-prefix**]} **Example:** `Device# clear ip bgp flap-statistics`	(Optional) Clears BGP flap statistics to make it less likely that a route will be dampened.
Step 10	**clear ip bgp dampening** **Example:** `Device# clear ip bgp dampening`	(Optional) Clears route dampening information, and unsuppress the suppressed routes.
Step 11	**copy running-config startup-config** **Example:** `Device# copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

Conditionally Injecting BGP Routes

Use this task to inject more specific prefixes into a BGP routing table over less specific prefixes that were selected through normal route aggregation. These more specific prefixes can be used to provide a finer granularity of traffic engineering or administrative control than is possible with aggregated routes.

Before you begin

This task assumes that the IGP is already configured for the BGP peers.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **router bgp** *autonomous-system-number*
4. **bgp inject-map** *inject-map-name* **exist-map** *exist-map-name* [**copy-attributes**]
5. **exit**
6. **route-map** *map-tag* [**permit** | **deny**] [*sequence-number*]

7. **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...*]}
8. **match ip route-source** {*access-list-number* | *access-list-name*} [*access-list-number...*| *access-list-name...*]
9. **exit**
10. **route-map** *map-tag* [**permit** | **deny**] [*sequence-number*]
11. **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...*]}
12. **set community** {*community-number* [**additive**] [*well-known-community*] | **none**}
13. **exit**
14. **ip prefix-list** *list-name* [**seq** *seq-value*] {**deny** *network/length* | **permit** *network/length*} [**ge** *ge-value*] [**le** *le-value*]
15. Repeat Step 14 for every prefix list to be created.
16. **exit**
17. **show ip bgp injected-paths**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**router bgp** *autonomous-system-number* **Example:** `Device(config)# router bgp 40000`	Enters router configuration mode for the specified routing process.
Step 4	**bgp inject-map** *inject-map-name* **exist-map** *exist-map-name* [**copy-attributes**] **Example:** `Device(config-router)# bgp inject-map ORIGINATE exist-map LEARNED_PATH`	Specifies the inject map and the exist map for conditional route injection. • Use the **copy-attributes** keyword to specify that the injected route inherit the attributes of the aggregate route.
Step 5	**exit** **Example:** `Device(config-router)# exit`	Exits router configuration mode and enters global configuration mode.
Step 6	**route-map** *map-tag* [**permit**	**deny**] [*sequence-number*] **Example:**	Configures a route map and enters route map configuration mode.

	Command or Action	Purpose
	`Device(config)# route-map LEARNED_PATH permit 10`	
Step 7	**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...*]} **Example:** `Device(config-route-map)# match ip address prefix-list SOURCE`	Specifies the aggregate route to which a more specific route will be injected. • In this example, the prefix list named SOURCE is used to redistribute the source of the route.
Step 8	**match ip route-source** {*access-list-number* | *access-list-name*} [*access-list-number...*| *access-list-name...*] **Example:** `Device(config-route-map)# match ip route-source prefix-list ROUTE_SOURCE`	Specifies the match conditions for redistributing the source of the route. • 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.
Step 9	**exit** **Example:** `Device(config-route-map)# exit`	Exits route map configuration mode and enters global configuration mode.
Step 10	**route-map** *map-tag* [**permit** | **deny**] [*sequence-number*] **Example:** `Device(config)# route-map ORIGINATE permit 10`	Configures a route map and enters route map configuration mode.
Step 11	**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...*]} **Example:** `Device(config-route-map)# set ip address prefix-list ORIGINATED_ROUTES`	Specifies the routes to be injected. In this example, the prefix list named originated_routes is used to redistribute the source of the route.
Step 12	**set community** {*community-number* [**additive**] [*well-known-community*] | **none**} **Example:**	Sets the BGP community attribute of the injected route.

	Command or Action	Purpose
	`Device(config-route-map)# set community 14616:555 additive`	
Step 13	**exit** **Example:** `Device(config-route-map)# exit`	Exits route map configuration mode and enters global configuration mode.
Step 14	**ip prefix-list** *list-name* [**seq** *seq-value*] {**deny** *network/length* | **permit** *network/length*} [**ge** *ge-value*] [**le** *le-value*] **Example:** `Device(config)# ip prefix-list SOURCE permit 10.1.1.0/24`	Configures a prefix list. In this example, the prefix list named SOURCE is configured to permit routes from network 10.1.1.0/24.
Step 15	Repeat Step 14 for every prefix list to be created.	--
Step 16	**exit** **Example:** `Device(config)# exit`	Exits global configuration mode and returns to privileged EXEC mode.
Step 17	**show ip bgp injected-paths** **Example:** `Device# show ip bgp injected-paths`	(Optional) Displays information about injected paths.

Configuring Peer Session Templates

Use the following tasks to create and configure a peer session template:

Configuring a Basic Peer Session Template

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 The following restrictions apply to the 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.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **router bgp** *autonomous-system-number*
4. **template peer-session** *session-template-name*
5. **remote-as** *autonomous-system-number*
6. **timers** *keepalive-interval hold-time*
7. **end**
8. **show ip bgp template peer-session** [*session-template-name*]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**router bgp** *autonomous-system-number* **Example:** `Device(config)# router bgp 101`	Enters router configuration mode and creates a BGP routing process.
Step 4	**template peer-session** *session-template-name* **Example:** `Device(config-router)# template peer-session INTERNAL-BGP`	Enters session-template configuration mode and creates a peer session template.

	Command or Action	Purpose
Step 5	**remote-as** *autonomous-system-number* **Example:** `Device(config-router-stmp)# remote-as 202`	(Optional) Configures peering with a remote neighbor in the specified autonomous system. **Note** Any supported general session command can be used here. For a list of the supported commands, see the "Restrictions" section.
Step 6	**timers** *keepalive-interval hold-time* **Example:** `Device(config-router-stmp)# timers 30 300`	(Optional) Configures BGP keepalive and hold timers. 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.
Step 7	**end** **Example:** `Device(config-router)# end`	Exits session-template configuration mode and returns to privileged EXEC mode.
Step 8	**show ip bgp template peer-session** [*session-template-name*] **Example:** `Device# show ip bgp template peer-session`	Displays locally configured peer session templates. 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.

Configuring Peer Session Template Inheritance with the inherit peer-session Command

This task configures peer session template inheritance with the **inherit peer-session** command. It creates and configures a peer session template and allows it to inherit a configuration from another peer session template.

Note The commands in Steps 5 and 6 are optional and could be replaced with any supported general session commands.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **router bgp** *autonomous-system-number*
4. **template peer-session** *session-template-name*
5. **description** *text-string*
6. **update-source** *interface-type interface-number*
7. **inherit peer-session** *session-template-name*
8. **end**
9. **show ip bgp template peer-session** [*session-template-name*]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**router bgp** *autonomous-system-number* **Example:** `Device(config)# router bgp 101`	Enters router configuration mode and creates a BGP routing process.
Step 4	**template peer-session** *session-template-name* **Example:** `Device(config-router)# template peer-session CORE1`	Enter session-template configuration mode and creates a peer session template.
Step 5	**description** *text-string* **Example:** `Device(config-router-stmp)# description CORE-123`	(Optional) Configures a description. The text string can be up to 80 characters. **Note** Any supported general session command can be used here. For a list of the supported commands, see the "Restrictions" section.
Step 6	**update-source** *interface-type interface-number* **Example:** `Device(config-router-stmp)# update-source loopback 1`	(Optional) Configures a router to select a specific source or interface to receive routing table updates. 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.
Step 7	**inherit peer-session** *session-template-name* **Example:** `Device(config-router-stmp)# inherit peer-session INTERNAL-BGP`	Configures this peer session template to inherit the configuration of another peer session template. The example configures this peer session template to inherit the configuration from INTERNAL-BGP. This template can be applied to a neighbor, and the configuration INTERNAL-BGP will be applied indirectly. No additional peer session templates can be directly applied. However,

	Command or Action	Purpose
		the directly inherited template can contain up to seven indirectly inherited peer session templates.
Step 8	**end** **Example:** `Device(config-router)# end`	Exits session-template configuration mode and enters privileged EXEC mode.
Step 9	**show ip bgp template peer-session** [*session-template-name*] **Example:** `Device# show ip bgp template peer-session`	Displays locally configured peer session templates. 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.

Configuring Peer Session Template Inheritance with the neighbor inherit peer-session Command

This task configures 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. Use the following steps to send a peer session template configuration to a neighbor to inherit.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **router bgp** *autonomous-system-number*
4. **neighbor** *ip-address* **remote-as** *autonomous-system-number*
5. **neighbor** *ip-address* **inherit peer-session** *session-template-name*
6. **end**
7. **show ip bgp template peer-session** [*session-template-name*]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**router bgp** *autonomous-system-number* **Example:** `Device(config)# router bgp 101`	Enters router configuration mode and creates a BGP routing process.

	Command or Action	Purpose
Step 4	**neighbor** *ip-address* **remote-as** *autonomous-system-number* **Example:** `Device(config-router)# neighbor 172.16.0.1 remote-as 202`	Configures a peering session with the specified neighbor. The explicit **remote-as** statement is required for the neighbor inherit statement in Step 5 to work. If a peering is not configured, the specified neighbor in Step 5 will not accept the session template.
Step 5	**neighbor** *ip-address* **inherit peer-session** *session-template-name* **Example:** `Device(config-router)# neighbor 172.16.0.1 inherit peer-session CORE1`	Sends a peer session template to a neighbor so that the neighbor can inherit the configuration. The example configures a device to send the peer session template named CORE1 to the 172.16.0.1 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.
Step 6	**end** **Example:** `Device(config-router)# end`	Exits router configuration mode and enters privileged EXEC mode.
Step 7	**show ip bgp template peer-session** [*session-template-name*] **Example:** `Device# show ip bgp template peer-session`	Displays locally configured peer session templates. 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.

Configuring Peer Policy Templates

Use the following tasks to create and configure a peer policy template:

Configuring Basic Peer Policy Templates

Perform this task to create a basic peer policy template with BGP policy configuration commands that can be applied to many neighbors using one of the next two tasks.

Note The commands in Steps 5 through 7 are optional and could be replaced with any supported BGP policy configuration commands.

Note 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. A BGP neighbor can be configured to belong only to a peer group or to inherit policies only from peer templates.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **router bgp** *autonomous-system-number*
4. **template peer-policy** *policy-template-name*
5. **maximum-prefix** *prefix-limit* [*threshold*] [**restart** *restart-interval* | **warning-only**]
6. **weight** *weight-value*
7. **prefix-list** *prefix-list-name* {**in** | **out**}
8. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**router bgp** *autonomous-system-number* **Example:** `Device(config)# router bgp 45000`	Enters router configuration mode and creates a BGP routing process.
Step 4	**template peer-policy** *policy-template-name* **Example:** `Device(config-router)# template peer-policy GLOBAL`	Enters policy-template configuration mode and creates a peer policy template.
Step 5	**maximum-prefix** *prefix-limit* [*threshold*] [**restart** *restart-interval*	**warning-only**] **Example:**	(Optional) Configures the maximum number of prefixes that a neighbor will accept from this peer.

	Command or Action	Purpose
	`Device(config-router-ptmp)# maximum-prefix 10000`	**Note** Any supported BGP policy configuration command can be used here. For a list of the supported commands, see the "Peer Policy Templates" section.
Step 6	**weight** *weight-value* **Example:** `Device(config-router-ptmp)# weight 300`	(Optional) Sets the default weight for routes that are sent from this neighbor. **Note** Any supported BGP policy configuration command can be used here. For a list of the supported commands, see the "Peer Policy Templates" section.
Step 7	**prefix-list** *prefix-list-name* {**in** | **out**} **Example:** `Device(config-router-ptmp)# prefix-list NO-MARKETING in`	(Optional) Filters prefixes that are received by the router or sent from the router. The prefix list in the example filters inbound internal addresses. **Note** Any supported BGP policy configuration command can be used here. For a list of the supported commands, see the "Peer Policy Templates" section.
Step 8	**end** **Example:** `Device(config-router-ptmp)# end`	Exits policy-template configuration mode and returns to privileged EXEC mode.

Configuring Peer Policy Template Inheritance with the inherit peer-policy Command

This task configures peer policy template inheritance using the **inherit peer-policy**command. It creates and configure a peer policy template and allows it to inherit a configuration from another peer policy template.

Note The commands in Steps 5 and 6 are optional and could be replaced with any supported BGP policy configuration commands.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **router bgp** *autonomous-system-number*
4. **template peer-policy** *policy-template-name*
5. **route-map** *map-name* {**in**| **out**}
6. **inherit peer-policy** *policy-template-name sequence-number*
7. **end**

8. **show ip bgp template peer-policy** [*policy-template-name*[**detail**]]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**router bgp** *autonomous-system-number* **Example:** `Device(config)# router bgp 45000`	Enters router configuration mode and creates a BGP routing process.
Step 4	**template peer-policy** *policy-template-name* **Example:** `Device(config-router)# template peer-policy NETWORK1`	Enter policy-template configuration mode and creates a peer policy template.
Step 5	**route-map** *map-name* {**in**| **out**} **Example:** `Device(config-router-ptmp)# route-map ROUTE in`	(Optional) Applies the specified route map to inbound or outbound routes. **Note** Any supported BGP policy configuration command can be used here.
Step 6	**inherit peer-policy** *policy-template-name sequence-number* **Example:** `Device(config-router-ptmp)# inherit peer-policy GLOBAL 10`	Configures the peer policy template to inherit the configuration of another peer policy template. • 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. • The example configures this peer policy template to inherit the configuration from GLOBAL. If the template created in these steps is applied to a neighbor, the configuration GLOBAL will also 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.

	Command or Action	Purpose
		• This template in the example will be evaluated first if no other templates are configured with a lower sequence number.
Step 7	**end** **Example:** `Device(config-router-ptmp)# end`	Exits policy-template configuration mode and returns to privileged EXEC mode.
Step 8	**show ip bgp template peer-policy** [*policy-template-name*[**detail**]] **Example:** `Device# show ip bgp template peer-policy NETWORK1 detail`	Displays locally configured peer policy templates. • 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.

Examples

The following sample output of the **show ip bgp template peer-policy** command with the **detail** keyword displays details of the policy named NETWORK1. The output in this example shows that the GLOBAL template was inherited. Details of route map and prefix list configurations are also displayed.

```
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 10.1.1.0/24
  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 10.2.2.0/24
```

Configuring 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. Perform the following steps to send a peer policy template configuration to a neighbor to inherit.

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 the inherited policies and policies configured directly on the specified neighbor.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **router bgp** *autonomous-system-number*
4. **neighbor** *ip-address* **remote-as** *autonomous-system-number*
5. **address-family ipv4** [**multicast** | **unicast** | **vrf** *vrf-name*]
6. **neighbor** *ip-address* **inherit peer-policy** *policy-template-name*
7. **end**
8. **show ip bgp neighbors** [*ip-address*[**policy** [**detail**]]]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**router bgp** *autonomous-system-number* **Example:** `Device(config)# router bgp 45000`	Enters router configuration mode and creates a BGP routing process.
Step 4	**neighbor** *ip-address* **remote-as** *autonomous-system-number* **Example:** `Device(config-router)# neighbor 192.168.1.2 remote-as 40000`	Configures a peering session with the specified neighbor. • The explicit **remote-as** statement is required for the **neighbor inherit** statement in Step 6 to work. If a peering is not configured, the specified neighbor in Step 6 will not accept the session template.

	Command or Action	Purpose
Step 5	**address-family ipv4** [**multicast** | **unicast** | **vrf** *vrf-name*] **Example:** `Device(config-router)# address-family ipv4 unicast`	Enters address family configuration mode to configure a neighbor to accept address family-specific command configurations.
Step 6	**neighbor** *ip-address* **inherit peer-policy** *policy-template-name* **Example:** `Device(config-router-af)# neighbor 192.168.1.2 inherit peer-policy GLOBAL`	Sends a peer policy template to a neighbor so that the neighbor can inherit the configuration. The example configures a router to send the peer policy template named GLOBAL to the 192.168.1.2 neighbor to inherit. This template can be applied to a neighbor, and if another peer policy template is indirectly inherited from GLOBAL, the indirectly inherited configuration will also be applied. Up to seven additional peer policy templates can be indirectly inherited from GLOBAL.
Step 7	**end** **Example:** `Device(config-router-af)# end`	Exits address family configuration mode and returns to privileged EXEC mode.
Step 8	**show ip bgp neighbors** [*ip-address*[**policy** [**detail**]]] **Example:** `Device# show ip bgp neighbors 192.168.1.2 policy`	Displays locally configured peer policy templates. • 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 applied to this neighbor per address family. • Use the **detail** keyword to display detailed policy information.

Examples

The following sample output shows the policies applied to the neighbor at 192.168.1.2. The output displays both inherited policies and policies configured on the neighbor device. Inherited polices are policies that the neighbor inherits from a peer-group or a peer-policy template.

```
Device# show ip bgp neighbors 192.168.1.2 policy
Neighbor: 192.168.1.2, 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
```

Configuring BGP Route Map Next-hop Self

Perform this task to modify the existing route map by adding the ip next-hop self setting and overriding the bgp next-hop unchanged and bgp next-hop unchanged allpaths settings.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **route-map** *map-tag* **permit** *sequence-number*
4. **match source-protocol** *source-protocol*
5. **set ip next-hop self**
6. **exit**
7. **route-map** *map-tag* **permit** *sequence-number*
8. **match route-type internal**
9. **match route-type external**
10. **match source-protocol** *source-protocol*
11. **exit**
12. **router bgp** *autonomous-system-number*
13. **neighbor** {*ip-address* | *ipv6-address* | *peer-group-name*} **remote-as** *autonomous-system-number*
14. **address-family vpnv4**
15. **neighbor** {*ip-address* | *ipv6-address* | *peer-group-name*} **activate**
16. **neighbor** {*ip-address* | *ipv6-address* | *peer-group-name*} **next-hop unchanged allpaths**
17. **neighbor** {*ip-address* | *ipv6-address* | *peer-group-name*} **route-map** *map-name* **out**
18. **exit**
19. **address-family ipv4** [**unicast** | **multicast**| **vrf** *vrf-name*]
20. **bgp route-map priority**
21. **redistribute** *protocol*
22. **redistribute** *protocol*
23. **exit-address-family**
24. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.

	Command or Action	Purpose
Step 3	**route-map** *map-tag* **permit** *sequence-number* **Example:** `Device(config)# route-map static-nexthop-rewrite permit 10`	Defines conditions for redistributing routes from one routing protocol to another routing protocol and enters route-map configuration mode.
Step 4	**match source-protocol** *source-protocol* **Example:** `Device(config-route-map)# match source-protocol static`	Matches Enhanced Interior Gateway Routing Protocol (EIGRP) external routes based on a source protocol.
Step 5	**set ip next-hop self** **Example:** `Device(config-route-map)# set ip next-hop self`	Configure local routes (for BGP only) with next hop of self.
Step 6	**exit** **Example:** `Device(config-route-map)# exit`	Exits route-map configuration mode and enters global configuration mode.
Step 7	**route-map** *map-tag* **permit** *sequence-number* **Example:** `Device(config)# route-map static-nexthop-rewrite permit 20`	Defines conditions for redistributing routes from one routing protocol to another routing protocol and enters route-map configuration mode.
Step 8	**match route-type internal** **Example:** `Device(config-route-map)# match route-type internal`	Redistributes routes of the specified type.
Step 9	**match route-type external** **Example:** `Device(config-route-map)# match route-type external`	Redistributes routes of the specified type.
Step 10	**match source-protocol** *source-protocol* **Example:** `Device(config-route-map)# match source-protocol connected`	Matches Enhanced Interior Gateway Routing Protocol (EIGRP) external routes based on a source protocol.
Step 11	**exit** **Example:**	Exits route-map configuration mode and enters global configuration mode.

	Command or Action	Purpose
	`Device(config-route-map)# exit`	
Step 12	**router bgp** *autonomous-system-number* **Example:** `Device(config)# router bgp 45000`	Enters router configuration mode and creates a BGP routing process.
Step 13	**neighbor** {*ip-address* | *ipv6-address* | *peer-group-name*} **remote-as** *autonomous-system-number* **Example:** `Device(config-router)# neighbor 172.16.232.50 remote-as 65001`	Adds an entry to the BGP or multiprotocol BGP neighbor table.
Step 14	**address-family vpnv4** **Example:** `Device(config-router)# address-family vpnv4`	Specifies the VPNv4 address family and enters address family configuration mode.
Step 15	**neighbor** {*ip-address* | *ipv6-address* | *peer-group-name*} **activate** **Example:** `Device(config-router-af)# neighbor 172.16.232.50 activate`	Enables the exchange of information with a Border Gateway Protocol (BGP) neighbor.
Step 16	**neighbor** {*ip-address* | *ipv6-address* | *peer-group-name*} **next-hop unchanged allpaths** **Example:** `Device(config-router-af)# neighbor 172.16.232.50 next-hop unchanged allpaths`	Enables an external EBGP peer that is configured as multihop to propagate the next hop unchanged.
Step 17	**neighbor** {*ip-address* | *ipv6-address* | *peer-group-name*} **route-map** *map-name* **out** **Example:** `Device(config-router-af)# neighbor 172.16.232.50 route-map static-nexthop-rewrite out`	Applies a route map to an outgoing route.
Step 18	**exit** **Example:** `Device(config-router-af)# exit`	Exits address family configuration mode and enters router configuration mode.
Step 19	**address-family ipv4** [**unicast** | **multicast**| **vrf** *vrf-name*] **Example:**	Specifies the IPv4 address family and enters address family configuration mode.

	Command or Action	Purpose
	`Device(config-router)# address-family ipv4 unicast vrf inside`	
Step 20	**bgp route-map priority** **Example:** `Device(config-router-af)# bgp route-map priority`	Configures the route map priority for the local BGP routing process
Step 21	**redistribute** *protocol* **Example:** `Device(config-router-af)# redistribute static`	Redistributes routes from one routing domain into another routing domain.
Step 22	**redistribute** *protocol* **Example:** `Device(config-router-af)# redistribute connected`	Redistributes routes from one routing domain into another routing domain.
Step 23	**exit-address-family** **Example:** `Device(config-router-af)# exit address-family`	Exits address family configuration mode and enters router configuration mode .
Step 24	**end** **Example:** `Device(config-router)# end`	Exits router configuration mode and enters privileged EXEC mode.

Configuration Examples for BGP

The following sections provide configuration examples for BGP.

Example: Configuring Conditional BGP Route Injection

The following sample output is similar to the output that will be displayed when the **show ip bgp injected-paths** command is entered:

```
Device# show ip bgp injected-paths

BGP table version is 11, local router ID is 10.0.0.1
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
*> 172.16.0.0       10.0.0.2                               0 ?
*> 172.17.0.0/16    10.0.0.2                               0 ?
```

Example: Configuring Peer Session Templates

The following 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
```

The following 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
```

The following example configures the 192.168.3.2 neighbor to inherit the CORE1 peer session template. The 192.168.3.2 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 work. If a peering is not configured, the specified neighbor will not accept the session template.

```
router bgp 45000
 neighbor 192.168.3.2 remote-as 50000
 neighbor 192.168.3.2 inherit peer-session CORE1
```

Examples: Configuring Peer Policy Templates

The following 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
```

The following 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
```

The following example creates a peer policy template named CUSTOMER-A. This peer policy template 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
```

The following example configures the 192.168.2.2 neighbor in address family mode to inherit the peer policy template named CUSTOMER-A. Assuming this example is a continuation of the example above, because the peer policy template named CUSTOMER-A above inherited the configuration from the templates named PRIMARY-IN and GLOBAL, the 192.168.2.2 neighbor will also indirectly inherit the peer policy templates named PRIMARY-IN and GLOBAL.

```
router bgp 45000
 neighbor 192.168.2.2 remote-as 50000
 address-family ipv4 unicast
  neighbor 192.168.2.2 inherit peer-policy CUSTOMER-A
  end
```

Example: Configuring BGP Route Map next-hop self

This section contains an example of how to configure BGP Route Map next-hop self.

In this example, a route map is configured that matches the networks where you wish to override settings for bgp next-hop unchanged and bgp next-hop unchanged allpath. Subsequently, next-hop self is configured. After this, the bgp route map priority is configured for the specified address family so that the previously specified route map takes priority over the settings for bgp next-hop unchanged and bgp next-hop unchanged allpath. This configuration results in static routes being redistributed with a next hop of self, but connected routes and routes learned via IBGP or EBGP continue to be redistributed with an unchanged next hop.

```
route-map static-nexthop-rewrite permit 10
 match source-protocol static
 set ip next-hop self
route-map static-nexthop-rewrite permit 20
 match route-type internal
 match route-type external
 match source-protocol connected
!
router bgp 65000
 neighbor 172.16.232.50 remote-as 65001
 address-family vpnv4
   neighbor 172.16.232.50 activate
   neighbor 172.16.232.50 next-hop unchanged allpaths
   neighbor 172.16.232.50 route-map static-nexthop-rewrite out
 exit-address-family
 address-family ipv4 unicast vrf inside
   bgp route-map priority
   redistribute static
   redistribute connected
 exit-address-family
end
```

Monitoring and Maintaining BGP

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. You can use the information to get resource utilization and solve network problems. You can also display information about node reachability and discover the routing path your device's packets are taking through the network.

The table given below lists the privileged EXEC commands for clearing and displaying BGP.

Table 20: IP BGP Clear and Show Commands

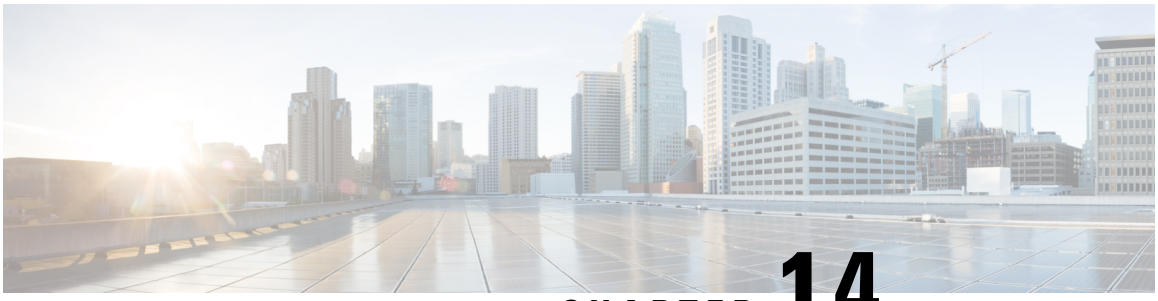
clear ip bgp *address*	Resets a particular BGP connection.
clear ip bgp *	Resets all BGP connections.
clear ip bgp peer-group *tag*	Removes all members of a BGP peer group.
show ip bgp *prefix*	Displays peer groups and peers not in peer groups to which has been advertised. Also displays prefix attributes such as hop and the local prefix.
show ip bgp cidr-only	Displays all BGP routes that contain subnet and supernet ne 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 a
show ip bgp inconsistent-as	Displays the routes with inconsistent originating autonomou
show ip bgp regexp *regular-expression*	Displays the routes that have an AS path that matches the s 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 connec individual neighbors.
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 to log messages that are generated when a BGP neighbor resets, comes up, or goes down.

Feature Information for Border Gateway Protocol

Table 21: Feature Information for Border Gateway Protocol

Feature Name	Release	Feature Information
Border Gateway Protocol	Cisco IOS XE Everest 16.5.1a	This feature was introduced.
Conditional BGP Route Injection	Cisco IOS XE Gibraltar 16.11.1	This feature was introduced.
BGP Peer Templates	Cisco IOS XE Gibraltar 16.11.1	This feature was introduced.
BGP Route Map Next Hop Self	Cisco IOS XE Gibraltar 16.11.1	This feature was introduced.



CHAPTER 14

Configuring BGP Support for 4-byte ASN

- Information About BGP Support for 4-byte ASN, on page 241
- How to Configure BGP Support for 4-byte ASN, on page 246
- Configuration Examples for BGP Support for 4-byte ASN, on page 252
- Additional References for BGP Support for 4-byte ASN, on page 257
- Feature History and Information for BGP Support for 4-byte ASN, on page 257

Information About BGP Support for 4-byte ASN

Prior to January 2009, BGP autonomous system (AS) numbers that were allocated to companies were 2-octet numbers in the range from 1 to 65535 as described in RFC 4271, *A Border Gateway Protocol 4 (BGP-4)*. Due to increased demand for AS numbers, the Internet Assigned Number Authority (IANA) started to allocate four-octet AS numbers in the range from 65536 to 4294967295. RFC 5396, *Textual Representation of Autonomous System (AS) Numbers*, documents three methods of representing AS numbers. Cisco has implemented the following two methods:

- Asplain—Decimal value notation where both 2-byte and 4-byte AS numbers are represented by their decimal value. For example, 65526 is a 2-byte AS number and 234567 is a 4-byte AS number.
- Asdot—Autonomous system dot notation where 2-byte AS numbers are represented by their decimal value and 4-byte AS numbers are represented by a dot notation. For example, 65526 is a 2-byte AS number and 1.169031 is a 4-byte AS number (this is dot notation for the 234567 decimal number).

For details about the third method of representing autonomous system numbers, see RFC 5396.

Asdot Only Autonomous System Number Formatting

The 4-octet (4-byte) AS numbers are entered and displayed only in asdot notation, for example, 1.10 or 45000.64000. When using regular expressions to match 4-byte AS numbers the asdot format includes a period, which is a special character in regular expressions. A backslash must be entered before the period (for example, 1\.14) to ensure the regular expression match does not fail. The table below shows the format in which 2-byte and 4-byte AS numbers are configured, matched in regular expressions, and displayed in **show** command output in Cisco IOS images where only asdot formatting is available.

Table 22: Asdot Only 4-Byte AS Number Format

Format	Configuration Format	Show Command Output and Regular Expression Match Format
asdot	2-byte: 1 to 65535 4-byte: 1.0 to 65535.65535	2-byte: 1 to 65535 4-byte: 1.0 to 65535.65535

Asplain as Default AS Number Formatting

The Cisco implementation of 4-byte AS numbers uses asplain as the default display format for AS numbers, but you can configure 4-byte AS numbers in both the asplain and asdot format. In addition, the default format for matching 4-byte AS numbers in regular expressions is asplain, so you must ensure that any regular expressions to match 4-byte AS numbers are written in the asplain format. If you want to change the default **show** command output to display 4-byte autonomous system numbers in the asdot format, use the **bgp asnotation dot** command under router configuration mode. When the asdot format is enabled as the default, any regular expressions to match 4-byte AS numbers must be written using the asdot format, or the regular expression match will fail. The tables below show that although you can configure 4-byte AS numbers in either asplain or asdot format, only one format is used to display **show** command output and control 4-byte AS number matching for regular expressions, and the default is asplain format. To display 4-byte AS numbers in **show** command output and to control matching for regular expressions in the asdot format, you must configure the **bgp asnotation dot** command. After enabling the **bgp asnotation dot** command, a hard reset must be initiated for all BGP sessions by entering the **clear ip bgp *** command.

Note If you are upgrading to an image that supports 4-byte AS numbers, you can still use 2-byte AS numbers. The **show** command output and regular expression match are not changed and remain in asplain (decimal value) format for 2-byte AS numbers regardless of the format configured for 4-byte AS numbers.

Table 23: Default Asplain 4-Byte AS Number Format

Format	Configuration Format	Show Command Output and Regular Expression Match Format
asplain	2-byte: 1 to 65535 4-byte: 65536 to 4294967295	2-byte: 1 to 65535 4-byte: 65536 to 4294967295
asdot	2-byte: 1 to 65535 4-byte: 1.0 to 65535.65535	2-byte: 1 to 65535 4-byte: 65536 to 4294967295

Table 24: Asdot 4-Byte AS Number Format

Format	Configuration Format	Show Command Output and Regular Expression Match Format
asplain	2-byte: 1 to 65535 4-byte: 65536 to 4294967295	2-byte: 1 to 65535 4-byte: 1.0 to 65535.65535
asdot	2-byte: 1 to 65535 4-byte: 1.0 to 65535.65535	2-byte: 1 to 65535 4-byte: 1.0 to 65535.65535

Reserved and Private AS Numbers

The Cisco implementation of BGP supports RFC 4893. RFC 4893 was developed to allow BGP to support a gradual transition from 2-byte AS numbers to 4-byte AS numbers. A new reserved (private) AS number, 23456, was created by RFC 4893 and this number cannot be configured as an AS number in the Cisco IOS CLI.

RFC 5398, *Autonomous System (AS) Number Reservation for Documentation Use*, describes new reserved AS numbers for documentation purposes. Use of the reserved numbers allow configuration examples to be accurately documented and avoids conflict with production networks if these configurations are literally copied. The reserved numbers are documented in the IANA AS number registry. Reserved 2-byte AS numbers are in the contiguous block, 64496 to 64511 and reserved 4-byte AS numbers are from 65536 to 65551 inclusive.

Private 2-byte AS numbers are still valid in the range from 64512 to 65534 with 65535 being reserved for special use. Private AS numbers can be used for internal routing domains but must be translated for traffic that is routed out to the Internet. BGP should not be configured to advertise private AS numbers to external networks. Cisco IOS software does not remove private AS numbers from routing updates by default. We recommend that ISPs filter private AS numbers.

Note AS number assignment for public and private networks is governed by the IANA. For information about AS numbers, including reserved number assignment, or to apply to register an AS number, see the following URL: http://www.iana.org/.

BGP Autonomous System Number Formats

Prior to January 2009, BGP autonomous system (AS) numbers that were allocated to companies were 2-octet numbers in the range from 1 to 65535 as described in RFC 4271, *A Border Gateway Protocol 4 (BGP-4)*. Due to increased demand for AS numbers, the Internet Assigned Number Authority (IANA) started to allocate four-octet AS numbers in the range from 65536 to 4294967295. RFC 5396, *Textual Representation of Autonomous System (AS) Numbers*, documents three methods of representing AS numbers. Cisco has implemented the following two methods:

- Asplain—Decimal value notation where both 2-byte and 4-byte AS numbers are represented by their decimal value. For example, 65526 is a 2-byte AS number and 234567 is a 4-byte AS number.
- Asdot—Autonomous system dot notation where 2-byte AS numbers are represented by their decimal value and 4-byte AS numbers are represented by a dot notation. For example, 65526 is a 2-byte AS number and 1.169031 is a 4-byte AS number (this is dot notation for the 234567 decimal number).

For details about the third method of representing autonomous system numbers, see RFC 5396.

Asdot Only Autonomous System Number Formatting

The 4-octet (4-byte) AS numbers are entered and displayed only in asdot notation, for example, 1.10 or 45000.64000. When using regular expressions to match 4-byte AS numbers the asdot format includes a period, which is a special character in regular expressions. A backslash must be entered before the period (for example, 1\.14) to ensure the regular expression match does not fail. The table below shows the format in which 2-byte and 4-byte AS numbers are configured, matched in regular expressions, and displayed in **show** command output in Cisco IOS images where only asdot formatting is available.

Table 25: Asdot Only 4-Byte AS Number Format

Format	Configuration Format	Show Command Output and Regular Expression Match Format
asdot	2-byte: 1 to 65535 4-byte: 1.0 to 65535.65535	2-byte: 1 to 65535 4-byte: 1.0 to 65535.65535

Asplain as Default AS Number Formatting

The Cisco implementation of 4-byte AS numbers uses asplain as the default display format for AS numbers, but you can configure 4-byte AS numbers in both the asplain and asdot format. In addition, the default format for matching 4-byte AS numbers in regular expressions is asplain, so you must ensure that any regular expressions to match 4-byte AS numbers are written in the asplain format. If you want to change the default **show** command output to display 4-byte autonomous system numbers in the asdot format, use the **bgp asnotation dot** command under router configuration mode. When the asdot format is enabled as the default, any regular expressions to match 4-byte AS numbers must be written using the asdot format, or the regular expression match will fail. The tables below show that although you can configure 4-byte AS numbers in either asplain or asdot format, only one format is used to display **show** command output and control 4-byte AS number matching for regular expressions, and the default is asplain format. To display 4-byte AS numbers in **show** command output and to control matching for regular expressions in the asdot format, you must configure the **bgp asnotation dot** command. After enabling the **bgp asnotation dot** command, a hard reset must be initiated for all BGP sessions by entering the **clear ip bgp *** command.

Note If you are upgrading to an image that supports 4-byte AS numbers, you can still use 2-byte AS numbers. The **show** command output and regular expression match are not changed and remain in asplain (decimal value) format for 2-byte AS numbers regardless of the format configured for 4-byte AS numbers.

Table 26: Default Asplain 4-Byte AS Number Format

Format	Configuration Format	Show Command Output and Regular Expression Match Format
asplain	2-byte: 1 to 65535 4-byte: 65536 to 4294967295	2-byte: 1 to 65535 4-byte: 65536 to 4294967295
asdot	2-byte: 1 to 65535 4-byte: 1.0 to 65535.65535	2-byte: 1 to 65535 4-byte: 65536 to 4294967295

Table 27: Asdot 4-Byte AS Number Format

Format	Configuration Format	Show Command Output and Regular Expression Match Format
asplain	2-byte: 1 to 65535 4-byte: 65536 to 4294967295	2-byte: 1 to 65535 4-byte: 1.0 to 65535.65535
asdot	2-byte: 1 to 65535 4-byte: 1.0 to 65535.65535	2-byte: 1 to 65535 4-byte: 1.0 to 65535.65535

Reserved and Private AS Numbers

The Cisco implementation of BGP supports RFC 4893. RFC 4893 was developed to allow BGP to support a gradual transition from 2-byte AS numbers to 4-byte AS numbers. A new reserved (private) AS number, 23456, was created by RFC 4893 and this number cannot be configured as an AS number in the Cisco IOS CLI.

RFC 5398, *Autonomous System (AS) Number Reservation for Documentation Use*, describes new reserved AS numbers for documentation purposes. Use of the reserved numbers allow configuration examples to be accurately documented and avoids conflict with production networks if these configurations are literally copied. The reserved numbers are documented in the IANA AS number registry. Reserved 2-byte AS numbers are in the contiguous block, 64496 to 64511 and reserved 4-byte AS numbers are from 65536 to 65551 inclusive.

Private 2-byte AS numbers are still valid in the range from 64512 to 65534 with 65535 being reserved for special use. Private AS numbers can be used for internal routing domains but must be translated for traffic that is routed out to the Internet. BGP should not be configured to advertise private AS numbers to external networks. Cisco IOS software does not remove private AS numbers from routing updates by default. We recommend that ISPs filter private AS numbers.

Note AS number assignment for public and private networks is governed by the IANA. For information about AS numbers, including reserved number assignment, or to apply to register an AS number, see the following URL: http://www.iana.org/.

Cisco Implementation of 4-Byte Autonomous System Numbers

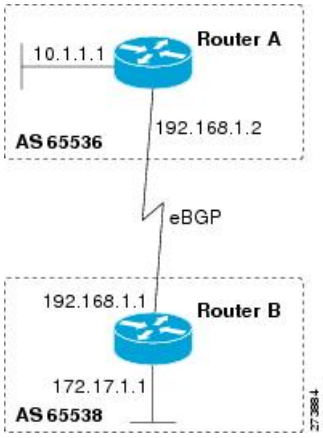
The Cisco implementation of 4-byte autonomous system (AS) numbers uses asplain—65538, for example—as the default regular expression match and output display format for AS numbers, but you can configure 4-byte AS numbers in both the asplain format and the asdot format as described in RFC 5396. To change the default regular expression match and output display of 4-byte AS numbers to asdot format, use the **bgp asnotation dot** command followed by the **clear ip bgp *** command to perform a hard reset of all current BGP sessions. For more details about 4-byte AS number formats, see the "BGP Autonomous System Number Formats" section.

The Cisco implementation of 4-byte AS numbers uses asdot—1.2, for example—as the only configuration format, regular expression match, and output display, with no asplain support. For an example of BGP peers in two autonomous systems using 4-byte numbers, see the figure below. To view a configuration example of the configuration between three neighbor peers in separate 4-byte autonomous systems configured using asdot notation, see the "Example: Configuring a BGP Routing Process and Peers Using 4-Byte Autonomous System Numbers" section.

Cisco also supports RFC 4893, which was developed to allow BGP to support a gradual transition from 2-byte AS numbers to 4-byte AS numbers. To ensure a smooth transition, we recommend that all BGP speakers within an AS that is identified using a 4-byte AS number be upgraded to support 4-byte AS numbers.

Note A new private AS number, 23456, was created by RFC 4893, and this number cannot be configured as an AS number in the Cisco IOS CLI.

Figure 8: BGP Peers in Two Autonomous Systems Using 4-Byte Numbers



How to Configure BGP Support for 4-byte ASN

The following sections provide configurational information about BGP support for 4-byte ASN.

Configuring a BGP Routing Process and Peers Using 4-Byte Autonomous System Numbers

Perform this task to configure a Border Gateway Protocol (BGP) routing process and BGP peers when the BGP peers are located in an autonomous system (AS) that uses 4-byte AS numbers. The address family configured here is the default IPv4 unicast address family, and the configuration is done at Router B in the figure above (in the "Cisco Implementation of 4-Byte Autonomous System Numbers" section). The 4-byte AS numbers in this task are formatted in the default asplain (decimal value) format; for example, Router B is in AS number 65538 in the figure above. Remember to perform this task for any neighbor routers that are to be BGP peers.

Before you begin

Note By default, neighbors that are defined using the **neighbor remote-as** command in router configuration mode exchange only IPv4 unicast address prefixes. To exchange other address prefix types, such as IPv6 prefixes, neighbors must also be activated using the **neighbor activate** command in address family configuration mode for the other prefix types.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **router bgp** *autonomous-system-number*
4. **neighbor** *ip-address* **remote-as** *autonomous-system-number*
5. Repeat Step 4 to define other BGP neighbors, as required.

6. **address-family ipv4** [**unicast** | **multicast** | **vrf** *vrf-name*]
7. **neighbor** *ip-address* **activate**
8. Repeat Step 7 to activate other BGP neighbors, as required.
9. **network** *network-number* [**mask** *network-mask*] [**route-map** *route-map-name*]
10. **end**
11. **show ip bgp** [*network*] [*network-mask*]
12. **show ip bgp summary**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**router bgp** *autonomous-system-number* **Example:** `Device(config)# router bgp 65538`	Enters router configuration mode for the specified routing process. • In this example, the 4-byte AS number, 65538, is defined in asplain notation.
Step 4	**neighbor** *ip-address* **remote-as** *autonomous-system-number* **Example:** `Device(config-router)# neighbor 192.168.1.2 remote-as 65536`	Adds the IP address of the neighbor in the specified AS to the IPv4 multiprotocol BGP neighbor table of the local device. • In this example, the 4-byte AS number, 65536, is defined in asplain notation.
Step 5	Repeat Step 4 to define other BGP neighbors, as required.	--
Step 6	**address-family ipv4** [**unicast**	**multicast**	**vrf** *vrf-name*] **Example:** `Device(config-router)# address-family ipv4 unicast`	Specifies the IPv4 address family and enters address family configuration mode. • The **unicast** keyword specifies the IPv4 unicast address family. By default, the device is placed in configuration mode for the IPv4 unicast address family if the **unicast** keyword is not specified with the **address-family ipv4** command. • The **multicast** keyword specifies IPv4 multicast address prefixes. • The **vrf** keyword and *vrf-name* argument specify the name of the virtual routing and forwarding (VRF)

	Command or Action	Purpose
		instance to associate with subsequent IPv4 address family configuration mode commands.
Step 7	**neighbor** *ip-address* **activate** **Example:** `Device(config-router-af)# neighbor 192.168.1.2 activate`	Enables the neighbor to exchange prefixes for the IPv4 unicast address family with the local device.
Step 8	Repeat Step 7 to activate other BGP neighbors, as required.	--
Step 9	**network** *network-number* [**mask** *network-mask*] [**route-map** *route-map-name*] **Example:** `Device(config-router-af)# network 172.17.1.0 mask 255.255.255.0`	(Optional) Specifies a network as local to this AS and adds it to the BGP routing table. • For exterior protocols the **network** command controls which networks are advertised. Interior protocols use the **network** command to determine where to send updates.
Step 10	**end** **Example:** `Device(config-router-af)# end`	Exits address family configuration mode and returns to privileged EXEC mode.
Step 11	**show ip bgp** [*network*] [*network-mask*] **Example:** `Device# show ip bgp 10.1.1.0`	(Optional) Displays the entries in the BGP routing table. **Note** Only the syntax applicable to this task is used in this example. For more details, see the *Cisco IOS IP Routing: BGP Command Reference*.
Step 12	**show ip bgp summary** **Example:** `Device# show ip bgp summary`	(Optional) Displays the status of all BGP connections.

The following output from the **show ip bgp** command at Router B shows the BGP routing table entry for network 10.1.1.0 learned from the BGP neighbor at 192.168.1.2 in Router A in the figure above with its 4-byte AS number of 65536 displayed in the default asplain format.

```
RouterB# show ip bgp 10.1.1.0

BGP routing table entry for 10.1.1.0/24, version 2
Paths: (1 available, best #1)
Advertised to update-groups:
2
65536
192.168.1.2 from 192.168.1.2 (10.1.1.99)
Origin IGP, metric 0, localpref 100, valid, external, best
```

The following output from the **show ip bgp summary** command shows the 4-byte AS number 65536 for the BGP neighbor 192.168.1.2 of Router A in the figure above after this task has been configured on Router B:

```
RouterB# show ip bgp summary

BGP router identifier 172.17.1.99, local AS number 65538
BGP table version is 3, main routing table version 3
2 network entries using 234 bytes of memory
2 path entries using 104 bytes of memory
3/2 BGP path/bestpath attribute entries using 444 bytes of memory
1 BGP AS-PATH entries using 24 bytes of memory
0 BGP route-map cache entries using 0 bytes of memory
0 BGP filter-list cache entries using 0 bytes of memory
BGP using 806 total bytes of memory
BGP activity 2/0 prefixes, 2/0 paths, scan interval 60 secs
Neighbor        V           AS MsgRcvd MsgSent   TblVer  InQ OutQ  Up/Down  Stated
192.168.1.2     4       65536       6       6        3    0    0 00:01:33        1
```

Modifying the Default Output and Regular Expression Match Format for 4-Byte Autonomous System Numbers

Perform this task to modify the default output format for 4-byte autonomous system (AS) numbers from asplain format to asdot notation format. The **show ip bgp summary** command is used to display the changes in output format for the 4-byte AS numbers.

SUMMARY STEPS

1. **enable**
2. **show ip bgp summary**
3. **configure terminal**
4. **router bgp** *autonomous-system-number*
5. **bgp asnotation dot**
6. **end**
7. **clear ip bgp ***
8. **show ip bgp summary**
9. **show ip bgp regexp** *regexp*
10. **configure terminal**
11. **router bgp** *autonomous-system-number*
12. **no bgp asnotation dot**
13. **end**
14. **clear ip bgp ***

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:**	Enables privileged EXEC mode. • Enter your password if prompted.

	Command or Action	Purpose
	`Device> enable`	
Step 2	**show ip bgp summary** **Example:** `Device# show ip bgp summary`	Displays the status of all Border Gateway Protocol (BGP) connections.
Step 3	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 4	**router bgp** *autonomous-system-number* **Example:** `Device(config)# router bgp 65538`	Enters router configuration mode for the specified routing process. • In this example, the 4-byte AS number, 65538, is defined in asplain notation.
Step 5	**bgp asnotation dot** **Example:** `Device(config-router)# bgp asnotation dot`	Changes the default output format of BGP 4-byte AS numbers from asplain (decimal values) to dot notation. **Note** 4-byte AS numbers can be configured using either asplain format or asdot format. This command affects only the output displayed for **show** commands or the matching of regular expressions.
Step 6	**end** **Example:** `Device(config-router)# end`	Exits address family configuration mode and returns to privileged EXEC mode.
Step 7	**clear ip bgp *** **Example:** `Device# clear ip bgp *`	Clears and resets all current BGP sessions. • In this example, a hard reset is performed to ensure that the 4-byte AS number format change is reflected in all BGP sessions. **Note** Only the syntax applicable to this task is used in this example. For more details, see the *Cisco IOS IP Routing: BGP Command Reference*.
Step 8	**show ip bgp summary** **Example:** `Device# show ip bgp summary`	Displays the status of all BGP connections.
Step 9	**show ip bgp regexp** *regexp*	Displays routes that match the AS path regular expression.

	Command or Action	Purpose
	Example: `Device# show ip bgp regexp ^1\.0$`	• In this example, a regular expression to match a 4-byte AS path is configured using asdot format.
Step 10	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 11	**router bgp** *autonomous-system-number* **Example:** `Device(config)# router bgp 65538`	Enters router configuration mode for the specified routing process. • In this example, the 4-byte AS number, 65538, is defined in asplain notation.
Step 12	**no bgp asnotation dot** **Example:** `Device(config-router)# no bgp asnotation dot`	Resets the default output format of BGP 4-byte AS numbers back to asplain (decimal values). **Note** 4-byte AS numbers can be configured using either asplain format or asdot format. This command affects only the output displayed for **show** commands or the matching of regular expressions.
Step 13	**end** **Example:** `Device(config-router)# end`	Exits router configuration mode and returns to privileged EXEC mode.
Step 14	**clear ip bgp *** **Example:** `Device# clear ip bgp *`	Clears and resets all current BGP sessions. • In this example, a hard reset is performed to ensure that the 4-byte AS number format change is reflected in all BGP sessions. **Note** Only the syntax applicable to this task is used in this example. For more details, see the *Cisco IOS IP Routing: BGP Command Reference*.

Examples

The following output from the **show ip bgp summary** command shows the default asplain format of the 4-byte AS numbers. Note the asplain format of the 4-byte AS numbers, 65536 and 65550.

```
Router# show ip bgp summary

BGP router identifier 172.17.1.99, local AS number 65538
BGP table version is 1, main routing table version 1
```

```
Neighbor        V           AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down  Statd
192.168.1.2     4        65536       7       7        1    0    0 00:03:04       0
192.168.3.2     4        65550       4       4        1    0    0 00:00:15       0
```

After the **bgp asnotation dot** command is configured (followed by the **clear ip bgp *** command to perform a hard reset of all current BGP sessions), the output is converted to asdot notation format as shown in the following output from the **show ip bgp summary** command. Note the asdot format of the 4-byte AS numbers, 1.0 and 1.14 (these are the asdot conversions of the 65536 and 65550 AS numbers.

```
Router# show ip bgp summary

BGP router identifier 172.17.1.99, local AS number 1.2
BGP table version is 1, main routing table version 1
Neighbor        V           AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down  Statd
192.168.1.2     4          1.0       9       9        1    0    0 00:04:13       0
192.168.3.2     4         1.14       6       6        1    0    0 00:01:24       0
```

After the **bgp asnotation dot** command is configured (followed by the **clear ip bgp *** command to perform a hard reset of all current BGP sessions), the regular expression match format for 4-byte AS paths is changed to asdot notation format. Although a 4-byte AS number can be configured in a regular expression using either asplain format or asdot format, only 4-byte AS numbers configured using the current default format are matched. In the first example below, the **show ip bgp regexp** command is configured with a 4-byte AS number in asplain format. The match fails because the default format is currently asdot format and there is no output. In the second example using asdot format, the match passes and the information about the 4-byte AS path is shown using the asdot notation.

Note The asdot notation uses a period, which is a special character in Cisco regular expressions. To remove the special meaning, use a backslash before the period.

```
Router# show ip bgp regexp ^65536$

Router# show ip bgp regexp ^1\.0$

BGP table version is 2, local router ID is 172.17.1.99
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete
   Network          Next Hop            Metric LocPrf Weight Path
*> 10.1.1.0/24      192.168.1.2              0             0 1.0 i
```

Configuration Examples for BGP Support for 4-byte ASN

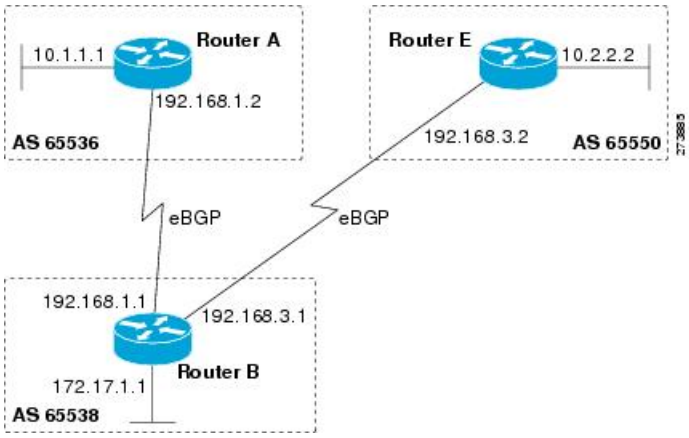
The following sections provide configuration examples for BGP support for 4-byte ASN.

Examples: Configuring a BGP Routing Process and Peers Using 4-Byte Autonomous System Numbers

Asplain Format

The following example shows the configuration for Router A, Router B, and Router E in the figure below with a Border Gateway Protocol (BGP) process configured between three neighbor peers (at Router A, at Router B, and at Router E) in separate 4-byte autonomous systems configured using asplain notation. IPv4 unicast routes are exchanged with all peers.

Figure 9: BGP Peers Using 4-Byte Autonomous System Numbers in Asplain Format



Router A

```
router bgp 65536
bgp router-id 10.1.1.99
no bgp default ipv4-unicast
bgp fast-external-fallover
bgp log-neighbor-changes
timers bgp 70 120
neighbor 192.168.1.1 remote-as 65538
!
address-family ipv4
neighbor 192.168.1.1 activate
no auto-summary
no synchronization
network 10.1.1.0 mask 255.255.255.0
exit-address-family
```

Router B

```
router bgp 65538
bgp router-id 172.17.1.99
no bgp default ipv4-unicast
bgp fast-external-fallover
bgp log-neighbor-changes
timers bgp 70 120
neighbor 192.168.1.2 remote-as 65536
neighbor 192.168.3.2 remote-as 65550
neighbor 192.168.3.2 description finance
```

```
!
address-family ipv4
neighbor 192.168.1.2 activate
neighbor 192.168.3.2 activate
no auto-summary
no synchronization
network 172.17.1.0 mask 255.255.255.0
exit-address-family
```

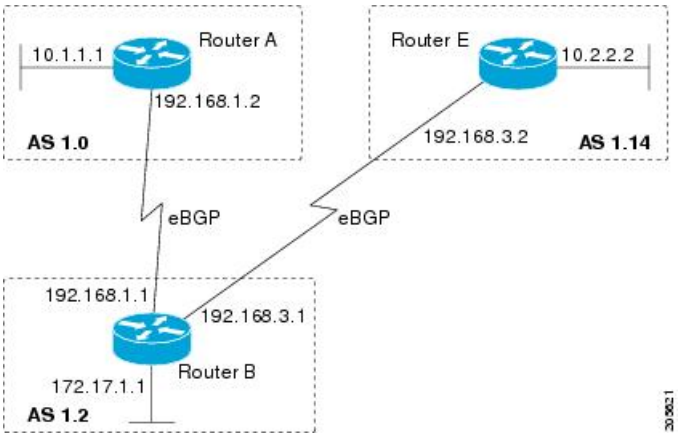
Router E

```
router bgp 65550
bgp router-id 10.2.2.99
no bgp default ipv4-unicast
bgp fast-external-fallover
bgp log-neighbor-changes
timers bgp 70 120
neighbor 192.168.3.1 remote-as 65538
!
address-family ipv4
neighbor 192.168.3.1 activate
no auto-summary
no synchronization
network 10.2.2.0 mask 255.255.255.0
exit-address-family
```

Asdot Format

The following example shows how to create the configuration for Router A, Router B, and Router E in the figure below with a BGP process configured between three neighbor peers (at Router A, at Router B, and at Router E) in separate 4-byte autonomous systems configured using the default asdot format. IPv4 unicast routes are exchanged with all peers.

Figure 10: BGP Peers Using 4-Byte Autonomous System Numbers in Asdot Format



Router A

```
router bgp 1.0
bgp router-id 10.1.1.99
no bgp default ipv4-unicast
bgp fast-external-fallover
bgp log-neighbor-changes
```

```
timers bgp 70 120
neighbor 192.168.1.1 remote-as 1.2
!
address-family ipv4
neighbor 192.168.1.1 activate
no auto-summary
no synchronization
network 10.1.1.0 mask 255.255.255.0
exit-address-family
```

Router B

```
router bgp 1.2
bgp router-id 172.17.1.99
no bgp default ipv4-unicast
bgp fast-external-fallover
bgp log-neighbor-changes
timers bgp 70 120
neighbor 192.168.1.2 remote-as 1.0
neighbor 192.168.3.2 remote-as 1.14
neighbor 192.168.3.2 description finance
!
address-family ipv4
neighbor 192.168.1.2 activate
neighbor 192.168.3.2 activate
no auto-summary
no synchronization
network 172.17.1.0 mask 255.255.255.0
exit-address-family
```

Router E

```
router bgp 1.14
bgp router-id 10.2.2.99
no bgp default ipv4-unicast
bgp fast-external-fallover
bgp log-neighbor-changes
timers bgp 70 120
neighbor 192.168.3.1 remote-as 1.2
!
address-family ipv4
neighbor 192.168.3.1 activate
no auto-summary
no synchronization
network 10.2.2.0 mask 255.255.255.0
exit-address-family
```

Examples: Configuring a VRF and Setting an Extended Community Using a BGP 4-Byte Autonomous System Number

The following example shows how to create a VRF with a route target that uses a 4-byte autonomous system number, 65537, and how to set the route target to extended community value 65537:100 for routes that are permitted by the route map:

```
ip vrf vpn_red
rd 64500:100
route-target both 65537:100
```

```
exit
route-map red_map permit 10
set extcommunity rt 65537:100
end
```

After the configuration is completed, use the **show route-map** command to verify that the extended community is set to the route target that contains the 4-byte autonomous system number of 65537:

```
RouterB# show route-map red_map
route-map red_map, permit, sequence 10
Match clauses:
Set clauses:
extended community RT:65537:100
Policy routing matches: 0 packets, 0 bytes
```

4-Byte Autonomous System Number RD Support

The following example shows how to create a VRF with a route distinguisher that contains a 4-byte AS number 65536, and a route target that contains a 4-byte autonomous system number, 65537:

```
ip vrf vpn_red
rd 65536:100
route-target both 65537:100
exit
```

After the configuration is completed, use the **show vrf** command to verify that the 4-byte AS number route distinguisher is set to 65536:100:

```
RouterB# show vrf vpn_red
Current configuration : 36 bytes
vrf definition x
rd 65536:100
!
```

Asdot Default Format in Cisco IOS Release 12.0(32)S12, and 12.4(24)T

The following example shows how to create a VRF with a route target that uses a 4-byte autonomous system number, 1.1, and how to set the route target to the extended community value 1.1:100 for routes that are permitted by the route map.

Note This example works if you have configured asdot as the default display format using the **bgp asnotation dot** command.

```
ip vrf vpn_red
rd 64500:100
route-target both 1.1:100
exit
route-map red_map permit 10
set extcommunity rt 1.1:100
end
```

After the configuration is completed, use the **show route-map** command to verify that the extended community is set to the route target that contains the 4-byte autonomous system number of 1.1.

```
RouterB# show route-map red_map
```

```
route-map red_map, permit, sequence 10
Match clauses:
Set clauses:
extended community RT:1.1:100
Policy routing matches: 0 packets, 0 bytes
```

Asdot Default Format for 4-Byte Autonomous System Number RD Support

The following example works if you have configured asdot as the default display format using the **bgp asnotation dot** command:

```
ip vrf vpn_red
rd 1.0:100
route-target both 1.1:100
exit
```

Additional References for BGP Support for 4-byte ASN

Related Documents

Related Topic	Document Title
BGP commands	*Cisco IOS IP Routing: BGP Command Reference*

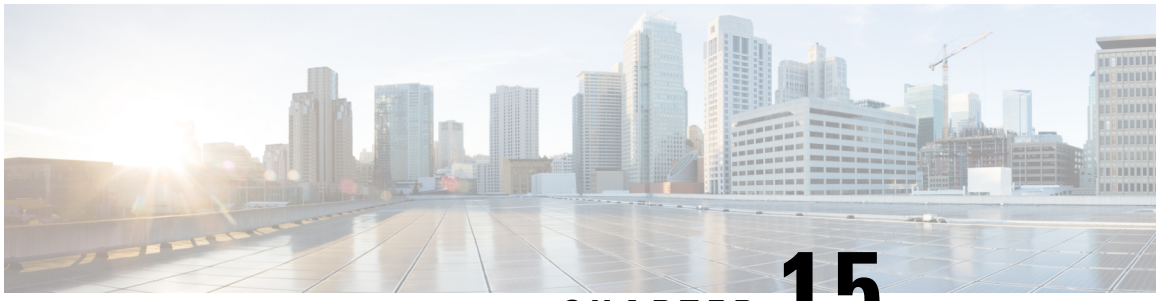
Standards and RFCs

Standard/RFC	Title
RFC 4893	*BGP Support for Four-octet AS Number Space*
RFC 5396	*Textual Representation of Autonomous System (AS) Numbers*
RFC 5398	*Autonomous System (AS) Number Reservation for Documentation Use*
RFC 5668	*4-Octet AS Specific BGP Extended Community*

Feature History and Information for BGP Support for 4-byte ASN

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Release	Modification
Cisco IOS XE Gibraltar 16.11.1	This feature was introduced.



CHAPTER 15

Configuring BGP Next Hop Unchanged

In an external BGP (eBGP) session, by default, the router changes the next hop attribute of a BGP route (to its own address) when the router sends out a route. The BGP Next Hop Unchanged feature allows BGP to send an update to an eBGP multihop peer with the next hop attribute unchanged.

- Restrictions for BGP Next Hop Unchanged, on page 259
- BGP Next Hop Unchanged, on page 259
- How to Configure BGP Next Hop Unchanged, on page 260
- Example: BGP Next Hop Unchanged for an eBGP Peer, on page 262
- Feature Information for BGP Next Hop Unchanged, on page 263

Restrictions for BGP Next Hop Unchanged

The BGP Next Hop Unchanged feature can be configured only between multihop eBGP peers. The following error message will be displayed if you try to configure this feature for a directly connected neighbor:

```
%BGP: Can propagate the nexthop only to multi-hop EBGP neighbor
```

BGP Next Hop Unchanged

In an external BGP (eBGP) session, by default, the router changes the next hop attribute of a BGP route (to its own address) when the router sends out a route. If the BGP Next Hop Unchanged feature is configured, BGP will send routes to an eBGP multihop peer without modifying the next hop attribute. The next hop attribute is unchanged.

Note There is an exception to the default behavior of the router changing the next hop attribute of a BGP route when the router sends out a route. When the next hop is in the same subnet as the peering address of the eBGP peer, the next hop is not modified. This is referred to as third party next-hop.

The BGP Next Hop Unchanged feature provides flexibility when designing and migrating networks. It can be used only between eBGP peers configured as multihop. It can be used in a variety of scenarios between two autonomous systems. One scenario is when multiple autonomous systems are connected that share the same IGP, or at least the routers have another way to reach each other's next hops (which is why the next hop can remain unchanged).

A common use of this feature is to configure Multiprotocol Label Switching (MPLS) inter-AS with multihop MP-eBGP for VPNv4 between RRs.

Another common use of this feature is a VPNv4 inter-AS Option C configuration, as defined in RFC4364, Section 10. In this configuration, VPNv4 routes are passed among autonomous systems between RR of different autonomous systems. The RRs are several hops apart, and have **neighbor next-hop unchanged** configured. PEs of different autonomous systems establish an LSP between them (via a common IGP or by advertising the next-hops--that lead to the PEs--via labeled routes among the ASBRs--routes from different autonomous systems separated by one hop). PEs are able to reach the next hops of the PEs in another AS via the LSPs, and can therefore install the VPNv4 routes in the VRF RIB.

How to Configure BGP Next Hop Unchanged

The following procedures contain the steps of how to configure BGP next hop unchanged.

Configuring the BGP Next Hop Unchanged for an eBGP Peer

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **router bgp** *as-number*
4. **address-family** {**ipv4** | **ipv6** | **l2vpn** | **nsap** | **rtfilter** | **vpnv4** | **vpnv6**}
5. **neighbor** {*ip-address* | *ipv6-address* | *peer-group-name*} **remote-as** *as-number*
6. **neighbor** {*ip-address* | *ipv6-address* | *peer-group-name*} **activate**
7. **neighbor** {*ip-address* | *ipv6-address* | *peer-group-name*} **ebgp-multihop** *ttl*
8. **neighbor** {*ip-address* | *ipv6-address* | *peer-group-name*} **next-hop-unchanged**
9. **end**
10. **show ip bgp**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**router bgp** *as-number* **Example:**	Enters router configuration mode, and creates a BGP routing process.

	Command or Action	Purpose
	`Device(config)# router bgp 65535`	
Step 4	**address-family** {**ipv4** | **ipv6** | **l2vpn** | **nsap** | **rtfilter** | **vpnv4** | **vpnv6**} **Example:** `Device(config-router-af)# address-family vpnv4`	Enters address family configuration mode to configure BGP peers to accept address family specific configurations.
Step 5	**neighbor** {*ip-address* | *ipv6-address* | *peer-group-name*} **remote-as** *as-number* **Example:** `Device(config-router-af)# neighbor 10.0.0.100 remote-as 65600`	Adds an entry to the BGP neighbor table.
Step 6	**neighbor** {*ip-address* | *ipv6-address* | *peer-group-name*} **activate** **Example:** `Device(config-router-af)# neighbor 10.0.0.100 activate`	Enables the exchange of information with the peer.
Step 7	**neighbor** {*ip-address* | *ipv6-address* | *peer-group-name*} **ebgp-multihop** *ttl* **Example:** `Device(config-router-af)# neighbor 10.0.0.100 ebgp-multihop 255`	Configures the local router to accept and initiate connections to external peers that reside on networks that are not directly connected.
Step 8	**neighbor** {*ip-address* | *ipv6-address* | *peer-group-name*} **next-hop-unchanged** **Example:** `Device(config-router-af)# neighbor 10.0.0.100 next-hop-unchanged`	Configures the router to send BGP updates to the specified eBGP peer without modifying the next hop attribute.
Step 9	**end** **Example:** `Device(config-router-af)# end`	Exits address family configuration mode, and enters privileged EXEC mode.
Step 10	**show ip bgp** **Example:** `Device# show ip bgp`	(Optional) Displays entries in the BGP routing table. The output will indicate if the **neighbor next-hop-unchanged** command has been configured for the selected address.

Configuring BGP Next Hop Unchanged using Route-Maps

Configuring outbound route-map for eBGP neighbor

To define the route-map and apply outbound policy for neighbor, use **set ip next-hop unchanged** command.

In the following configuration the next-hop for prefix 1.1.1.1 is not changed while sending to the eBGP neighbor 15.1.1.2:

```
enable
config terminal
router bgp 2
 bgp log-neighbor-changes
 neighbor 15.1.1.2 remote-as 3
 neighbor 15.1.1.2 ebgp-multihop 10
 !
 address-family ipv4
  neighbor 15.1.1.2 activate
  neighbor 15.1.1.2 route-map A out
  exit address-family
!
route-map A permit 10
 match ip address 1
 set ip next-hop unchanged
!
access-list 1 permit 1.1.1.1
end
```

Configuring next-hop unchanged for both iBGP and eBGP path prefixes while sending to eBGP neighbor

To configure next-hop unchanged for both iBGP and eBGP path prefixes while sending to eBGP neighbor, use **next-hop-unchanged allpaths** command.

In the following configuration the next-hop is not changed for both iBGP and eBGP path prefixes while sending to eBGP neighbor 15.1.1.2:

```
enable
config terminal
router bgp 2
 bgp log-neighbor-changes
 neighbor 15.1.1.2 remote-as 3
 neighbor 15.1.1.2 ebgp-multihop 10
!
address-family ipv4
 neighbor 15.1.1.2 activate
 neighbor 15.1.1.2 next-hop-unchanged allpaths
 exit address-family
!
end
```

Example: BGP Next Hop Unchanged for an eBGP Peer

The following example configures a multihop eBGP peer at 10.0.0.100 in a remote AS. When the local router sends updates to that peer, it will send them without modifying the next hop attribute.

```
router bgp 65535
 address-family ipv4
```

```
neighbor 10.0.0.100 remote-as 65600
neighbor 10.0.0.100 activate
neighbor 10.0.0.100 ebgp-multihop 255
neighbor 10.0.0.100 next-hop-unchanged
end
```

Note All address families, such as IPv4, IPv6, VPNv4, VPNv6, L2VPN, and so on support the **next-hop unchanged** command. However, for the address family L2VPN BGP VPLS signaling, you must use the **next-hop self** command for its proper functioning.

Feature Information for BGP Next Hop Unchanged

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Table 28: Feature Information for BGP Next Hop Unchanged

Feature Name	Releases	Feature Information
BGP Next Hop Unchanged	Cisco IOS XE Gibraltar 16.11.1	The BGP Next Hop Unchanged feature allows BGP to send an update to an eBGP multihop peer with the next hop attribute unchanged.

CHAPTER 16

Configuring IS-IS Routing

- Information About IS-IS Routing, on page 265
- How to Configure IS-IS, on page 269
- How to Configure IS-IS Authentication, on page 277
- Monitoring and Maintaining IS-IS, on page 280
- Feature Information for IS-IS, on page 281

Information About IS-IS Routing

Integrated Intermediate System-to-Intermediate System (IS-IS) is an ISO dynamic routing protocol (described in ISO 105890). To enable IS-IS you should create an IS-IS routing process and assign it to a specific interface, rather than to a network. You can specify more than one IS-IS routing process per Layer 3 device by using the multiarea IS-IS configuration syntax. You should then configure the parameters for each instance of the IS-IS routing process.

Small IS-IS networks are built as a single area that includes all the devices in the network. As the network grows larger, the netwok reorganizes itself into a backbone area made up of all the connected set of Level 2 devices still connected to their local areas. Within a local area, devices know how to reach all system IDs. Between areas, devices know how to reach the backbone, and the backbone devices know how to reach other areas.

Devices establish Level 1 adjacencies to perform routing within a local area (station routing). Devices establish Level 2 adjacencies to perform routing between Level 1 areas (area routing).

A single Cisco device can participate in routing in up to 29 areas and can perform Level 2 routing in the backbone. In general, each routing process corresponds to an area. By default, the first instance of the routing process that is configured performs both Level 1 and Level 2 routing. You can configure additional device instances, which are automatically treated as Level 1 areas. You must configure the parameters for each instance of the IS-IS routing process individually.

For IS-IS multiarea routing, you can configure only one process to perform Level 2 routing, although you can define up to 29 Level 1 areas for each Cisco unit. If Level 2 routing is configured on any process, all additional processes are automatically configured as Level 1. You can configure this process to perform Level 1 routing at the same time. If Level 2 routing is not desired for a device instance, remove the Level 2 capability using the **is-type** command in global configuration mode. Use the **is-type** command also to configure a different device instance as a Level 2 device.

IS-IS Authentication

To prevent unauthorized devices from injecting false routing information into the link-state database, you can either set a plain text password for each interface and an area password for each IS-IS area, or you can configure an IS-IS authentication.

Plain text passwords do not provide security against unauthorized users. You can configure a plain text password to prevent unauthorized networking devices from forming adjacencies with the router. The password is exchanged as plain text and is visible to agents having access to view the IS-IS packets.

The new style of IS-IS authentication provides the following advantages over the plain text password configuration commands:

- Passwords are encrypted when the software configuration is displayed.
- Passwords are easier to manage and change.
- Passwords can be changed to new passwords without disrupting network operations.
- Authentication transitions which are nondisruptive.

Authentication modes (IS-IS authentication or plain text password) can either be configured on a given scope (IS-IS instance or interface) or level, but not both. However, different modes can be configured for different scopes or levels. In case mixed modes are configured, different keys must be used for different modes to ensure that the encrypted passwords in the protocol data units (PDUs) are not compromised.

Clear Text Authentication

IS-IS clear text authentication provides the same functionality provided by the **area-password** or **domain-password** command.

HMAC-MD5 Authentication

IS-IS supports message digest algorithm 5 (MD5) authentication, which is more secure than clear text authentication.

Hashed Message Authentication Code (HMAC) is a mechanism for message authentication codes (MACs) using cryptographic hash functions. HMAC-MD5 authentication adds an HMAC-MD5 digest to each IS-IS PDU. The digest allows authentication at the IS-IS routing protocol level, which prevents unauthorized routing messages from being injected into the network routing domain.

The following are the benefits of HMAC-MD5 authentication:

- Passwords can be changed to new passwords without disrupting routing messages.
- Authentication transitions which are nondisruptive. The device accepts PDUs with either no authentication information or stale authentication information and sends PDUs with current authentication information. These transitions are useful when migrating from no authentication to some type of authentication, when changing the authentication type, and when changing the authentication keys.

HMAC-SHA Authentication

IS-IS supports Secure Hash Algorithm (SHA) authentication, that is, SHA-1, SHA-256, SHA-384, and SHA-512, which is more secure than MD5 authentication or clear text authentication.

When you enable the HMAC-SHA authentication method, a shared secret key is configured on all the devices that are connected on a common network. For each packet, this key is used to generate and verify a message digest that gets added to the packet. The message digest is a one-way function of the packet and the secret key.

Hitless Upgrade

Before you migrate from using one type of security authentication to another, you must do the following:

1. All the devices must be loaded with the new image that supports the new authentication type. The devices will continue to use the original authentication method until all the devices have been loaded with the new image that supports the new authentication method, and all the devices have been configured to use the new authentication method.
2. Add a key chain with both the current key and a new key. For example when migrating from HMAC-MD5 to HMAC-SHA1-20, the current key is HMAC-MD5, and the new key is HMAC-SHA1-20. Ensure that the current key has a later end date for the send-lifetime field than the new key so that IS-IS continues to send the current key. Set the accept-lifetime value of both the keys to infinite so that IS-IS accepts both the keys.
3. After step 2 is completed, for all the devices in a link or area the current key can be removed from the key chain.

Nonstop Forwarding Awareness

The integrated IS-IS Nonstop Forwarding (NSF) Awareness feature is supported for IPv4G. The feature allows customer premises equipment (CPE) devices that are NSF-aware to help NSF-capable devices perform nonstop forwarding of packets. The local device is not necessarily performing NSF, but its NSF awareness capability allows the integrity and accuracy of the routing database and the link-state database on the neighboring NSF-capable device to be maintained during the switchover process.

The integrated IS-IS Nonstop Forwarding (NSF) Awareness feature is automatically enabled and requires no configuration.

IS-IS Global Parameters

The following are the optional IS-IS global parameters that you can configure:

- You can force a default route into an IS-IS routing domain by configuring a default route that is controlled by a route map. You can also specify the other filtering options that are configurable under a route map.
- You can configure the device to ignore IS-IS link-state packets (LSPs) that are received with internal checksum errors, or to purge corrupted LSPs, and cause the initiator of the LSP to regenerate it.
- You can assign passwords to areas and domains.
- You can create aggregate addresses that are represented in the routing table by a summary address (based on route summarization). Routes learned from other routing protocols can also be summarized. The metric used to advertise the summary is the smallest metric of all the specific routes.
- You can set an overload bit.
- You can configure the LSP refresh interval and the maximum time that an LSP can remain in the device database without a refresh.

- You can set the throttling timers for LSP generation, shortest path first computation, and partial route computation.
- You can configure the device to generate a log message when an IS-IS adjacency changes state (Up or Down).
- If a link in the network has a maximum transmission unit (MTU) size of less than 1500 bytes, you can lower the LSP MTU so that routing still occurs.
- You can use the **partition avoidance** command to prevent an area from becoming partitioned when full connectivity is lost among a Level 1-2 border device, adjacent Level 1 devices, and end hosts.

IS-IS Interface Parameters

You can optionally configure certain interface-specific IS-IS parameters independently from other attached devices. However, if you change default value, such as multipliers and time intervals, it makes sense to also change them on multiple devices and interfaces. Most of the interface parameters can be configured for level 1, level 2, or both.

The following are the interface-level parameters that you can configure:

- The default metric on the interface that is used as a value for the IS-IS metric and assigned when quality of service (QoS) routing is not performed.
- The hello interval (length of time between hello packets sent on the interface) or the default hello packet multiplier used on the interface to determine the hold time sent in IS-IS hello packets. The hold time determines how long a neighbor waits for another hello packet before declaring the neighbor down. This determines how quickly a failed link or neighbor is detected so that routes can be recalculated. Change the hello multiplier in circumstances where hello packets are lost frequently and IS-IS adjacencies are failing unnecessarily. You can raise the hello multiplier and lower the hello interval correspondingly to make the hello protocol more reliable, without increasing the time required to detect a link failure.
- Other time intervals:
  - Complete sequence number PDU (CSNP) interval—CSNPs are sent by the designated device to maintain database synchronization.
  - Retransmission interval—This is the time between retransmission of IS-IS LSPs for point-to-point links.
  - IS-IS LSP retransmission throttle interval—This is the maximum rate (number of milliseconds between packets) at which IS-IS LSPs are resent on point-to-point links. This interval is different from the retransmission interval, which is the time between successive retransmissions of the same LSP.
- Designated device-election priority, which allows you to reduce the number of adjacencies required on a multiaccess network, which in turn reduces the amount of routing protocol traffic and the size of the topology database.
- The interface circuit type, which is the type of adjacency required for neighbors on the specified interface.
- Password authentication for the interface.

How to Configure IS-IS

The following sections provide information on how to enable IS-IS on an interface, how to configure IS-IS global parameters, and how to configure IS-IS interface parameters.

Default IS-IS Configuration

Table 29: Default IS-IS Configuration

Feature	Default Setting
Ignore link-state PDU (LSP) errors	Enabled.
IS-IS type	Conventional IS-IS—The router acts as both a Level 1 (station) and a router. Multiarea IS-IS—The first instance of the IS-IS routing process is a router. Remaining instances are Level 1 routers.
Default-information originate	Disabled.
Log IS-IS adjacency state changes.	Disabled.
LSP generation throttling timers	Maximum interval between two consecutive occurrences—5000 mi Initial LSP generation delay—50 milliseconds. Hold time between the first and second LSP generation—200 millis
LSP maximum lifetime (without a refresh)	1200 seconds (20 minutes) before the LSP packet is deleted.
LSP refresh interval	Every 900 seconds (15 minutes).
Maximum LSP packet size	1497 bytes.
NSF Awareness	Enabled. Allows Layer 3 devices to continue forwarding packets from Nonstop Forwarding-capable router during hardware or software ch
Partial route computation (PRC) throttling timers	Maximum PRC wait interval—5000 milliseconds. Initial PRC calculation delay after a topology change—50 milliseco Hold time between the first and second PRC calculation—200 milli
Partition avoidance	Disabled.
Password	No area or domain password is defined, and authentication is disabl
Set-overload-bit	Disabled. When enabled, if no arguments are entered, the overload b immediately and remains set until you enter the **no set-overload-bit**

Feature	Default Setting
Shortest path first (SPF) throttling timers	Maximum interval between consecutive SFPs—5000 milliseconds. Initial SFP calculation after a topology change—200 milliseconds. Hold time between the first and second SFP calculation—50 millisecon
Summary-address	Disabled.

Enabling IS-IS Routing

To enable IS-IS, specify a name and a network entity title (NET) for each routing process. Enable IS-IS routing on the interface and specify the area for each instance of the routing process.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device>enable`	Enables privileged EXEC mode. Enter your password, if prompted.
Step 2	**configure terminal** **Example:** `Device#configure terminal`	Enters global configuration mode.
Step 3	**clns routing** **Example:** `Device(config)#clns routing`	Enables ISO connectionless routing on the device.
Step 4	**router isis** [*area tag*] **Example:** `Device(config)#router isis tag1`	Enables IS-IS routing for the specified routing process and enters IS-IS routing configuration mode. (Optional) Use the *area tag* argument to identify the area to which the IS-IS router is assigned. Enter a value if you are configuring multiple IS-IS areas. The first IS-IS instance that is configured is Level 1-2 by default. Later instances are automatically configured as Level 1. You can change the level of routing by using the **is-type** command in global configuration mode.
Step 5	**net** *network-entity-title* **Example:** `Device(config-router)#net 47.0004.004d.0001.0001.0c11.1111.00`	Configures the NETs for the routing process. While configuring multiarea IS-IS, specify a NET for each routing process. Specify a name for a NET and for an address.

	Command or Action	Purpose
Step 6	**is-type** {**level-1** | **level-1-2** | **level-2-only**} **Example:** `Device(config-router)#is-type level-2-only`	(Optional) Configures the router to act as a Level 1 (station) router, a Level 2 (area) router for multiarea routing, or both (the default): • **level 1**—Acts as a station router only. • **level 1-2**—Acts as both a station router and an area router. • **level 2**—Acts as an area router only.
Step 7	**exit** **Example:** `Device(config-router)#end`	Returns to global configuration mode.
Step 8	**interface** *interface-id* **Example:** `Device(config)#interface gigabitethernet 1/0/1`	Specifies an interface to route IS-IS, and enters interface configuration mode. If the interface is not already configured as a Layer 3 interface, enter the **no switchport** command to configure the interface into Layer 3 mode.
Step 9	**ip router isis** [*area tag*] **Example:** `Device(config-if)#ip router isis tag1`	Configures an IS-IS routing process on the interface and attaches an area designator to the routing process.
Step 10	**ip address** *ip-address-mask* **Example:** `Device(config-if)#ip address 10.0.0.5 255.255.255.0`	Defines the IP address for the interface. An IP address is required for all the interfaces in an area, that is enabled for IS-IS, if any one interface is configured for IS-IS routing.
Step 11	**end** **Example:** `Device(config)#end`	Returns to privileged EXEC mode.
Step 12	**show isis** [*area tag*] **database detail** **Example:** `Device#show isis database detail`	Verifies your entries.

Configuring IS-IS Global Parameters

To configure global IS-IS parameters, perform this procedure:

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device>enable`	Enables privileged EXEC mode. Enter your password, if prompted.
Step 2	**configure terminal** **Example:** `Device#configure terminal`	Enters global configuration mode.
Step 3	**router isis** **Example:** `Device(config)#router isis`	Specifies the IS-IS routing protocol and enters router configuration mode.
Step 4	**default-information originate** [**route-map** *map-name*] **Example:** `Device(config-router)#default-information originate route-map map1`	(Optional) Forces a default route into the IS-IS routing domain. When you enter the **route-map** *map-name* command, the routing process generates the default route for a valid route map.
Step 5	**ignore-lsp-errors** **Example:** `Device(config-router)#ignore-lsp-errors`	(Optional) Configures the device to ignore LSPs with internal checksum errors, instead of purging the LSPs. This command is enabled by default (corrupted LSPs are dropped). To purge the corrupted LSPs, enter the **no ignore-lsp-errors** command in router configuration mode.
Step 6	**area-password** *password* **Example:** `Device(config-router)#area-password 1password`	(Optional Configures the area authentication password that is inserted in Level 1 (station router level) LSPs.
Step 7	**domain-password** *password* **Example:** `Device(config-router)#domain-password 2password`	(Optional) Configures the routing domain authentication password that is inserted in Level 2 (area router level) LSPs.
Step 8	**summary-address** *address mask* [**level-1** | **level-1-2** | **level-2**] **Example:** `Device(config-router)#summary-address 10.1.0.0 255.255.0.0 level-2`	(Optional) Creates a summary of addresses for a given level.

	Command or Action	Purpose
Step 9	**set-overload-bit** [**on-startup** {*seconds* | **wait-for-bgp**}] **Example:** `Device(config-router)#set-overload-bit on-startup wait-for-bgp`	(Optional) Sets an overload bit to allow other devices to ignore the device in their shortest path first (SPF) calculations if the device is having problems. • (Optional) **on-startup**—Sets the overload bit only on startup. If **on-startup** is not specified, the overload bit is set immediately and remains set until you enter the **no set-overload-bit** command. If **on-startup** is specified, you must either enter number of seconds or enter **wait-for-bgp**. • *seconds*—When the **on-startup** keyword is configured, it causes the overload bit to be set when the system is started and remains set for the specified number of seconds. The range is from 5 to 86400 seconds. • **wait-for-bgp**—When the **on-startup** keyword is configured, causes the overload bit to be set when the system is started and remains set until BGP has converged. If BGP does not signal the IS-IS that it is converged, the IS-IS will turn off the overload bit after 10 minutes.
Step 10	**lsp-refresh-interval** *seconds* **Example:** `Device(config-router)#lsp-refresh-interval 1080`	(Optional) Sets an LSP refresh interval, in seconds. The range is from 1 to 65535 seconds. The default is to send LSP refreshes every 900 seconds (15 minutes).
Step 11	**max-lsp-lifetime** *seconds* **Example:** `Device(config-router)#max-lsp-lifetime 1000`	(Optional) Sets the maximum time that LSP packets remain in the router database without being refreshed. The range is from 1 to 65535 seconds. The default is 1200 seconds (20 minutes). After the specified time interval, the LSP packet is deleted.
Step 12	**lsp-gen-interval** [**level-1** | **level-2**] *lsp-max-wait* [*lsp-initial-wait lsp-second-wait*] **Example:** `Device(config-router)#lsp-gen-interval level-2 2 50 100`	(Optional) Sets the IS-IS LSP generation throttling timers: • *lsp-max-wait*—Maximum interval (in milliseconds) between two consecutive occurrences of an LSP being generated. The range is from 1 to 120; the default is 5000. • *lsp-initial-wait*—Initial LSP generation delay (in milliseconds). The range is from 1 to 10000; the default is 50. • *lsp-second-wait*—Hold time between the first and second LSP generation (in milliseconds). The range is from 1 to 10000; the default is 200.

	Command or Action	Purpose
Step 13	**spf-interval** [**level-1** | **level-2**] *spf-max-wait* [*spf-initial-wait spf-second-wait*] **Example:** `Device(config-router)#spf-interval level-2 5 10 20`	(Optional) Sets IS-IS SPF throttling timers. • *spf-max-wait*—Maximum interval between consecutive SFPs (in milliseconds). The range is from 1 to 120; the default is 5000. • *spf-initial-wait*—Initial SFP calculation after a topology change (in milliseconds). The range is from 1 to 10000; the default is 50. • *spf-second-wait*—Hold time between the first and second SFP calculation (in milliseconds). The range is from 1 to 10000; the default is 200.
Step 14	**prc-interval** *prc-max-wait* [*prc-initial-wait prc-second-wait*] **Example:** `Device(config-router)#prc-interval 5 10 20`	(Optional) Sets IS-IS PRC throttling timers. • *prc-max-wait*—Maximum interval (in milliseconds) between two consecutive PRC calculations. The range is from 1 to 120; the default is 5000. • *prc-initial-wait*—Initial PRC calculation delay (in milliseconds) after a topology change. The range is from 1 to 10,000; the default is 50. • *prc-second-wait*—Hold time between the first and second PRC calculation (in milliseconds). The range is from 1 to 10,000; the default is 200.
Step 15	**log-adjacency-changes** [**all**] **Example:** `Device(config-router)#log-adjacency-changes all`	(Optional) Sets the router to log IS-IS adjacency state changes. Enter **all** to include all the changes generated by events that are not related to the IS-IS hellos, including End System-to-Intermediate System PDUs and LSPs.
Step 16	**lsp-mtu** *size* **Example:** `Device(config-router)#lsp mtu 1560`	(Optional) Specifies the maximum LSP packet size, in bytes. The range is from 128 to 4352; the default is 1497 bytes. **Note** If a link in the network has a reduced MTU size, you must change the LSP MTU size on all the devices in the network.
Step 17	**partition avoidance** **Example:** `Device(config-router)#partition avoidance`	(Optional) Causes an IS-IS Level 1-2 border router to stop advertising the Level 1 area prefix into the Level 2 backbone when full connectivity is lost among the border router, all adjacent level 1 routers, and end hosts.
Step 18	**end** **Example:** `Device(config)#end`	Returns to privileged EXEC mode.

Configuring IS-IS Interface Parameters

To configure IS-IS interface-specific parameters, perform this procedure:

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device>enable`	Enables privileged EXEC mode. Enter your password, if prompted.
Step 2	**configure terminal** **Example:** `Device#configure terminal`	Enters global configuration mode.
Step 3	**interface** *interface-id* **Example:** `Device(config)#interface gigabitethernet 1/0/1`	Specifies the interface to be configured and enters interface configuration mode. If the interface is not already configured as a Layer 3 interface, enter the **no switchport** command to configure the interface into Layer 3 mode.
Step 4	**isis metric** *default-metric* [**level-1** | **level-2**] **Example:** `Device(config-if)#isis metric 15`	(Optional) Configures the metric (or cost) for the specified interface. The range is from 0 to 63; the default is 10. If no level is entered, the default is applied to both Level 1 and Level 2 routers.
Step 5	**isis hello-interval** {*seconds* | **minimal**} [**level-1** | **level-2**] **Example:** `Device(config-if)#isis hello-interval minimal`	(Optional) Specifies the length of time between the hello packets sent by the device. By default, a value that is three times the hello interval *seconds* is advertised as the *holdtime* in the hello packets sent. With smaller hello intervals, topological changes are detected faster, but there is more routing traffic. • **minimal**—Causes the system to compute the hello interval based on the hello multiplier so that the resulting hold time is 1 second. • *seconds*—Range is from 1 to 65535; default is 10 seconds.
Step 6	**isis hello-multiplier** *multiplier* [**level-1** | **level-2**] **Example:** `Device(config-if)#isis hello-multiplier 5`	(Optional) Specifies the number of IS-IS hello packets a neighbor must miss before the device declares the adjacency as down. The range is from 3 to 1000; default is 3. **Note** Using a smaller hello multiplier causes fast convergence, but might result in routing instability.

	Command or Action	Purpose
Step 7	**isis csnp-interval** *seconds* [**level-1** | **level-2**] **Example:** `Device(config-if)#isis csnp-interval 15`	(Optional) Configures the IS-IS complete sequence number PDU (CSNP) interval for the interface. The range is from 0 to 65535; default is 10 seconds.
Step 8	**isis retransmit-interval** *seconds* **Example:** `Device(config-if)#isis retransmit-interval 7`	(Optional) Configures the number of seconds between the retransmission of IS-IS LSPs for point-to-point links. Specify an integer that is greater than the expected round-trip delay between any two routers on the network. The range is from 0 to 65535; default is 5 seconds.
Step 9	**isis retransmit-throttle-interval** *milliseconds* **Example:** `Device(config-if)#isis retransmit-throttle-interval 4000`	(Optional) Configures the IS-IS LSP retransmission throttle interval, which is the maximum rate (number of milliseconds between packets) at which IS-IS LSPs will be resent on point-to-point links. The range is from 0 to 65535; default is determined by the **isis lsp-interval** command.
Step 10	**isis priority** *value* [**level-1** | **level-2**] **Example:** `Device(config-if)#isis priority 50`	(Optional) Configures the priority for the designated router. The range is from 0 to 127; default is 64.
Step 11	**isis circuit-type** {**level-1** | **level-1-2** | **level-2-only**} **Example:** `Device(config-if)#isis circuit-type level-1-2`	(Optional) Configures the type of adjacency required for neighbors on the specified interface (specify the interface circuit type). • **level-1**—Level 1 adjacency is established if there is at least one area address that is common to both this node and its neighbors. • **level-1-2**—Level 1 and Level 2 adjacency are established if the neighbor is also configured as both Level 1 and Level 2, and there is at least one area in common. If there is no area in common, a Level 2 adjacency is established. This is the default option. • **level 2**—Level 2 adjacency is established. If the neighbor router is a Level 1 router, no adjacency is established.
Step 12	**isis password** *password* [**level-1** | **level-2**] **Example:** `Device(config-if)#isis password secret`	(Optional) Configures the authentication password for an interface. By default, authentication is disabled. Specifying Level 1 or Level 2 enables the password only for Level 1 or Level 2 routing, respectively. If you do not specify a level, the default is Level 1 and Level 2.
Step 13	**end** **Example:**	Returns to privileged EXEC mode.

	Command or Action	Purpose
	`Device(config)#end`	

How to Configure IS-IS Authentication

The following sections provide information on how to generate authentication keys, how to configure IS-IS authentication for an interface, and how to configure IS-IS authentication for an instance.

Configuring Authentication Keys

You can configure multiple keys with lifetimes. To send authentication packets, the key with the latest send lifetime setting is selected. If multiple keys have the same send lifetime setting, the key is randomly selected. Use the **accept-lifetime** command for examining and accepting the authentication packets that are received. The device must be aware of these lifetimes.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device>enable`	Enables privileged EXEC mode. Enter your password, if prompted.
Step 2	**configure terminal** **Example:** `Device#configure terminal`	Enters global configuration mode.
Step 3	**key chain** *name-of-chain* **Example:** `Device(config)#key chain key10`	Identifies a key chain, and enters key chain configuration mode.
Step 4	**key** *number* **Example:** `Device(config-keychain)#key 2000`	Identifies the key number. The range is from 0 to 65535.
Step 5	**key-string** *text* **Example:** `Device(config-keychain-key)#Room 20, 10th floor`	Identifies the key string. The string can contain 1-80 uppercase and lowercase alphanumeric characters, but the first character cannot be a number.
Step 6	**accept-lifetime** *start-time* {**infinite** | *end-time* | **duration** *seconds*}	(Optional) Specifies the time period during which the key can be received.

	Command or Action	Purpose
	Example: `Device(config-keychain-key)#accept-lifetime 12:30:00 Jan 25 1009 infinite`	The *start-time* and *end-time* syntax can be either *hh:mm:ss month date year* or *hh:mm:ss date month year*. The default is forever with the default *start-time* and the earliest acceptable date is January 1, 1993. The default *end-time* and **duration** is **infinite**.
Step 7	**send-lifetime** *start-time* {**infinite** | *end-time* | **duration** *seconds*} **Example:** `Device(config-keychain-key)#accept-lifetime 23:30:00 Jan 25 1019 infinite`	(Optional) Specifies the time period during which the key can be sent. The *start-time* and *end-time* syntax can be either *hh:mm:ss month date year* or *hh:mm:ss date month year*. The default *start-time* is **infinite** and the earliest acceptable date is January 1, 1993. The default *end-time* and **duration** is **infinite**.
Step 8	**cryptographic-algorithm** {**hmac-sha-1** | **hmac-sha-256** | **hmac-sha-384** | **hmac-sha-512** | **md5** } **Example:** `Device(config-keychain-key)#cryptographic-algorithm hmac-sha1-256`	(Optional) Specifies the cryptographic algorithm.
Step 9	**end** **Example:** `Device(config-keychain-key)#end`	Returns to privileged EXEC mode.
Step 10	**show key chain** **Example:** `Device#show key chain`	Displays authentication key information.

Configuring HMAC-MD5 or Clear Text Authentication for an IS-IS Instance

To achieve a smooth transition from one authentication method to another and to allow for continuous authentication of IS-IS PDUs, perform this procedure on each device that communicates in the network.

Before you begin

You should have generated an authentication string key. The same authentication string key should be configured on all the devices in the network.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:**	Enables privileged EXEC mode. Enter your password, if prompted.

	Command or Action	Purpose
	`Device>enable`	
Step 2	**configure terminal** **Example:** `Device#configure terminal`	Enters global configuration mode.
Step 3	**router isis** [**area** *tag*] **Example:** `Device(config)#router isis 1`	Enables IS-IS as an IP routing protocol and assigns a tag to a process, if required. Enters router configuration mode.
Step 4	**authentication send-only** [**level-1** | **level-2**] **Example:** `Device(config-router)#authentication send-only`	Specifies that authentication is performed only on the PDUs that are being sent (not received) for the specified IS-IS instance.
Step 5	**authentication mode** {**md5** | **text**}[**level-1** | **level-2**] **Example:** `Device(config-router)#authentication mode md5`	Specifies the types of authentication to be used in PDUs for the specified IS-IS instance: • **md5**—MD5 authentication. • **text**—Clear text authentication.
Step 6	**authentication key-chain** *name-of-chain* [**level-1** | **level-2**] **Example:** `Device(config-router)#authentication key-chain remote3754`	Enables authentication for the specified IS-IS instance.
Step 7	**no authentication send-only** **Example:** `Device(config-router)#no authentication send-only`	Specifies that authentication is performed only on the PDUs that are being sent and received for the specified IS-IS instance.

Configuring HMAC-MD5 or Clear Text Authentication for an IS-IS Interface

To achieve a smooth transition from one authentication method to another and to allow for continuous authentication of IS-IS PDUs, perform this procedure on each device that communicates in the network.

Before you begin

You should have generated an authentication string key. The same authentication string key should be configured on all the devices in the network.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device>enable`	Enables privileged EXEC mode. Enter your password, if prompted.
Step 2	**configure terminal** **Example:** `Device#configure terminal`	Enters global configuration mode.
Step 3	**interface** *type number* **Example:** `Device(config)#interface ethernet 0`	Configures an interface.
Step 4	**isis authentication send-only** [**level-1** | **level-2**] **Example:** `Device(config-if)#isis authentication send-only`	Specifies that authentication is performed only on the PDUs being sent (not received) for the specified IS-IS interface.
Step 5	**isis authentication mode** {**md5** | **text**}[**level-1** | **level-2**] **Example:** `Device(config-if)#isis authentication mode md5`	Specifies the types of authentication to be used in PDUs for the specified IS-IS interface: • **md5**—MD5 authentication. • **text**—Clear text authentication.
Step 6	**isis authentication key-chain** *name-of-chain* [**level-1** | **level-2**] **Example:** `Device(config-if)#isis authentication key-chain multistate87723`	Enables MD5 authentication for the specified IS-IS interface.
Step 7	**no isis authentication send-only** **Example:** `Device(config-if)#no isis authentication send-only`	Specifies that authentication is performed only on the PDUs that are being sent and received for the IS-IS interface.

Monitoring and Maintaining IS-IS

You can display specific IS-IS statistics, such as the contents of routing tables, caches, and databases. You can also display information about specific interfaces, filters, or neighbors.

The following table lists the privileged EXEC commands for clearing and displaying IS-IS routing.

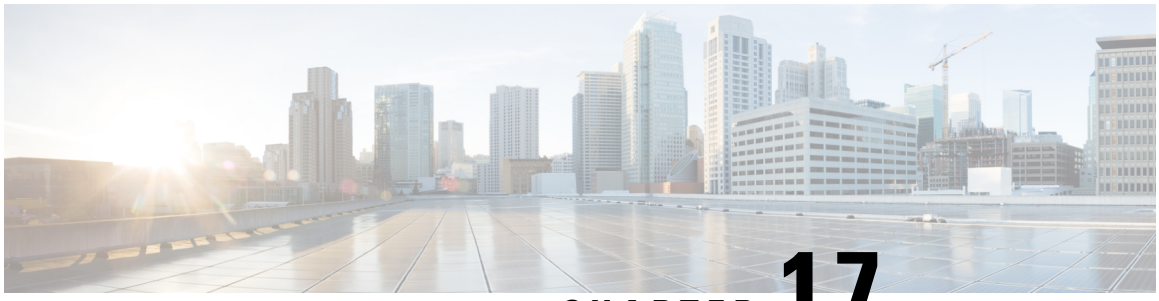
Table 30: IS-IS show Commands

Command
show ip route isis
show isis database
show isis routes
show isis spf-log
show isis topology
show route-map
trace clns *destination*

Feature Information for IS-IS

Table 31: Feature Information for IS-IS

Feature Name	Release	Feature Information
Intermediate System-to-Intermediate System (IS-IS)	Cisco IOS XE Everest 16.5.1a	This feature was introduced.
	Cisco IOS XE Gibraltar 16.10.1	IS-IS now supports Secure Hash Algorithm (SHA) authentication—SHA-1, SHA-256, SHA-384, and SHA-512.



CHAPTER 17

Protocol-Independent Features

- Distributed Cisco Express Forwarding and Load-Balancing Scheme for CEF Traffic , on page 283
- Number of Equal-Cost Routing Paths, on page 288
- Static Unicast Routes, on page 289
- Default Routes and Networks, on page 291
- Route Maps to Redistribute Routing Information, on page 293
- Policy-Based Routing, on page 299
- Filtering Routing Information, on page 303
- Managing Authentication Keys, on page 306

Distributed Cisco Express Forwarding and Load-Balancing Scheme for CEF Traffic

The following sections provide information about distributed Cisco express forwarding (CEF) and load-balancing scheme for CEF traffic.

Restrictions for Configuring a Load-Balancing Scheme for CEF Traffic

- You must globally configure load balancing on device or device stack members in the same way.
- Per-packet load balancing for CEF traffic is not supported.

Information About Cisco Express Forwarding

Cisco Express Forwarding (CEF) is a Layer 3 IP switching technology used to optimize network performance. CEF implements an advanced IP look-up and forwarding algorithm to deliver maximum Layer 3 switching performance. CEF is less CPU-intensive than fast switching route caching, allowing more CPU processing power to be dedicated to packet forwarding. In a switch stack, the hardware uses distributed CEF (dCEF) in the stack. In dynamic networks, fast switching cache entries are frequently invalidated because of routing changes, which can cause traffic to be process switched using the routing table, instead of fast switched using the route cache. CEF and dCEF use the Forwarding Information Base (FIB) lookup table to perform destination-based switching of IP packets.

The two main components in CEF and dCEF are the distributed FIB and the distributed adjacency tables.

- The FIB is similar to a routing table or information base and maintains a mirror image of the forwarding information in the IP routing table. When routing or topology changes occur in the network, the IP routing table is updated, and those changes are reflected in the FIB. The FIB maintains next-hop address information based on the information in the IP routing table. Because the FIB contains all known routes that exist in the routing table, CEF eliminates route cache maintenance, is more efficient for switching traffic, and is not affected by traffic patterns.
- Nodes in the network are said to be adjacent if they can reach each other with a single hop across a link layer. CEF uses adjacency tables to prepend Layer 2 addressing information. The adjacency table maintains Layer 2 next-hop addresses for all FIB entries.

Because the switch or switch stack uses Application Specific Integrated Circuits (ASICs) to achieve Gigabit-speed line rate IP traffic, CEF or dCEF forwarding applies only to the software-forwarding path, that is, traffic that is forwarded by the CPU.

CEF Load-Balancing Overview

CEF load balancing allows you to optimize resources by distributing traffic over multiple paths. CEF load balancing works based on a combination of source and destination packet information.

You can configure load balancing on a per-destination. Because load-balancing decisions are made on the outbound interface, load balancing must be configured on the outbound interface.

Load-Balancing Algorithms for CEF Traffic

The following load-balancing algorithms are provided for use with CEF traffic. Select a load-balancing algorithm with the **ip cef load-sharing algorithm** command.

- Original algorithm—The original load-balancing algorithm produces distortions in load sharing across multiple devices because the same algorithm was used on every device. Depending on your network environment, you should select the algorithm.
- Universal algorithm—The universal load-balancing algorithm allows each device on the network to make a different load sharing decision for each source-destination address pair, which resolves load-sharing imbalances. The device is set to perform universal load sharing by default.

Per-Destination Load Balancing for CEF Traffic

Per-destination load balancing allows the device to use multiple paths to achieve load sharing across multiple source-destination host pairs. Packets for a given source-destination host pair are guaranteed to take the same path, even if multiple paths are available. Traffic streams destined for different pairs tend to take different paths.

Per-destination load balancing is enabled by default when you enable CEF. To use per-destination load balancing, you do not perform any additional tasks once CEF is enabled. Per-destination is the load-balancing method of choice for most situations.

Because per-destination load balancing depends on the statistical distribution of traffic, load sharing becomes more effective as the number of source-destination host pairs increases.

You can use per-destination load balancing to ensure that packets for a given host pair arrive in order. All packets intended for a certain host pair are routed over the same link (or links).

How to Configure Cisco Express Forwarding

CEF or distributed CEF is enabled globally by default. If for some reason it is disabled, you can re-enable it by using the **ip cef** or **ip cef distributed** global configuration command.

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 2	**ip cef** **Example:** `Device(config)# ip cef`	Enables CEF operation on a non-stacking switch. Go to Step 4.
Step 3	**ip cef distributed** **Example:** `Device(config)# ip cef distributed`	Enables CEF operation on a active switch.
Step 4	**interface** *interface-id* **Example:** `Device(config)# interface gigabitethernet 1/0/1`	Enters interface configuration mode, and specifies the Layer 3 interface to configure.
Step 5	**ip route-cache cef** **Example:** `Device(config-if)# ip route-cache cef`	Enables CEF on the interface for software-forwarded traffic. **Note** The **ip route-cache cef** command is enabled by default and it cannot be disabled.
Step 6	**end** **Example:** `Device(config-if)# end`	Returns to privileged EXEC mode.
Step 7	**show ip cef** **Example:** `Device# show ip cef`	Displays the CEF status on all interfaces.
Step 8	**show cef linecard** [**detail**] **Example:** `Device# show cef linecard detail`	(Optional) Displays CEF-related interface information on a non-stacking switch.

	Command or Action	Purpose
Step 9	**show cef linecard** [*slot-number*] [**detail**] **Example:** `Device# show cef linecard 5 detail`	(Optional) Displays CEF-related interface information on a switch by stack member for all switches in the stack or for the specified switch. (Optional) For *slot-number*, enter the stack member switch number.
Step 10	**show cef interface** [*interface-id*] **Example:** `Device# show cef interface gigabitethernet 1/0/1`	Displays detailed CEF information for all interfaces or the specified interface.
Step 11	**show adjacency** **Example:** `Device# show adjacency`	Displays CEF adjacency table information.
Step 12	**copy running-config startup-config** **Example:** `Device# copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

How to Configure a Load-Balancing for CEF Traffic

The following sections provide information on configuring load-balancing for CEF traffic.

Enabling or Disabling CEF Per-Destination Load Balancing

To enable or disable CEF per-destination load balancing, perform the following procedure:

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface** *interface-id*
4. [**no**] **ip load-sharing per-destination**
5. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device# enable`	Enters global configuration mode.

	Command or Action	Purpose
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface** *interface-id* **Example:** `Device(config-if)# interface gigabitethernet 1/0/1`	Enters interface configuration mode, and specifies the Layer 3 interface to configure.
Step 4	[**no**] **ip load-sharing per-destination** **Example:** `Device(config-if)# ip load-sharing per-destination`	Enables per-destination load balancing for CEF on the interface. The **no ip load-sharing per-destination** command disables per-destination load balancing for CEF on the interface.
Step 5	**end** **Example:** `Device(config-if)# end`	Exits interface configuration mode and returns to privileged EXEC mode.

Selecting a Tunnel Load-Balancing Algorithm for CEF Traffic

Select the tunnel algorithm when your network environment contains only a few source and destination pairs. The device is set to perform universal load sharing by default.

To select a tunnel load-balancing algorithm for CEF traffic, perform the following procedure:

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **ip cef load-sharing algorithm** {**original** | **universal** [*id*] }
4. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device# enable`	Enters global configuration mode.
Step 2	**configure terminal** **Example:**	Enters global configuration mode.

	Command or Action	Purpose
	`Device# configure terminal`	
Step 3	**ip cef load-sharing algorithm** {**original** | **universal** [*id*] } **Example:** `Device(config)# ip cef load-sharing algorithm universal`	Selects a CEF load-balancing algorithm. • The **original** keyword sets the load-balancing algorithm to the original algorithm, based on a source IP and destination IP hash. • The **universal** keyword sets the load-balancing algorithm to one that uses a source IP, destination IP, Layer 3 Protocol, Layer 4 source port, Layer 4 destination port and IPv6 flow label (for IPv6 traffic). • The *id* argument is a fixed identifier.
Step 4	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode.

Example: Enabling or Disabling CEF Per-Destination Load Balancing

Per-destination load balancing is enabled by default when you enable CEF. The following example shows how to disable per-destination load balancing:

```
Device> enable
Device# configure terminal
Device(config)# interface Ethernet1/0/1
Device(config-if)# no ip load-sharing per-destination
Device(config-if)# end
```

Number of Equal-Cost Routing Paths

The following sections provide information about number of equal-cost routing paths.

Information About Equal-Cost Routing Paths

When a router has two or more routes to the same network with the same metrics, these routes can be thought of as having an equal cost. The term parallel path is another way to see occurrences of equal-cost routes in a routing table. If a router has two or more equal-cost paths to a network, it can use them concurrently. Parallel paths provide redundancy in case of a circuit failure and also enable a router to load balance packets over the available paths for more efficient use of available bandwidth. Equal-cost routes are supported across switches in a stack.

Even though the router automatically learns about and configures equal-cost routes, you can control the maximum number of parallel paths supported by an IP routing protocol in its routing table. Although the

switch software allows a maximum of 32 equal-cost routes, the switch hardware will never use more than 16 paths per route.

How to Configure Equal-Cost Routing Paths

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**router** {**rip** | **ospf** | **eigrp**} **Example:** `Device(config)# router eigrp`	Enters router configuration mode.
Step 4	**maximum-paths** *maximum* **Example:** `Device(config-router)# maximum-paths 2`	Sets the maximum number of parallel paths for the protocol routing table. The range is from 1 to 16; the default is 4 for most IP routing protocols, but only 1 for BGP.
Step 5	**end** **Example:** `Device(config-router)# end`	Returns to privileged EXEC mode.
Step 6	**show ip protocols** **Example:** `Device# show ip protocols`	Verifies the setting in the *Maximum path* field.
Step 7	**copy running-config startup-config** **Example:** `Device# copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

Static Unicast Routes

The following sections provide information about static unicast routes.

Information About Static Unicast Routes

Static unicast routes are user-defined routes that cause packets moving between a source and a destination to take a specified path. Static routes can be important if the router cannot build a route to a particular destination and are useful for specifying a gateway of last resort to which all unroutable packets are sent.

The switch retains static routes until you remove them. However, you can override static routes with dynamic routing information by assigning administrative distance values. Each dynamic routing protocol has a default administrative distance, as listed in Table 41-16. If you want a static route to be overridden by information from a dynamic routing protocol, set the administrative distance of the static route higher than that of the dynamic protocol.

Table 32: Dynamic Routing Protocol Default Administrative Distances

Route Source	Default Distance
Connected interface	0
Static route	1
Enhanced IRGP summary route	5
Internal Enhanced IGRP	90
IGRP	100
OSPF	110
Internal BGP	200
Unknown	225

Static routes that point to an interface are advertised through RIP, IGRP, and other dynamic routing protocols, whether or not static **redistribute** router configuration commands were specified for those routing protocols. These static routes are advertised because static routes that point to an interface are considered in the routing table to be connected and hence lose their static nature. However, if you define a static route to an interface that is not one of the networks defined in a network command, no dynamic routing protocols advertise the route unless a **redistribute** static command is specified for these protocols.

When an interface goes down, all static routes through that interface are removed from the IP routing table. When the software can no longer find a valid next hop for the address specified as the forwarding router's address in a static route, the static route is also removed from the IP routing table.

Configuring Static Unicast Routes

Static unicast routes are user-defined routes that cause packets moving between a source and a destination to take a specified path. Static routes can be important if the router cannot build a route to a particular destination and are useful for specifying a gateway of last resort to which all unroutable packets are sent.

Follow these steps to configure a static route:

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**ip route prefix mask** {*address* | *interface*} [*distance*] **Example:** `Device(config)# ip route prefix mask gigabitethernet 1/0/4`	Establish a static route.
Step 4	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode.
Step 5	**show ip route** **Example:** `Device# show ip route`	Displays the current state of the routing table to verify the configuration.
Step 6	**copy running-config startup-config** **Example:** `Device# copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

What to do next

Use the **no ip route** *prefix mask* {*address*| *interface*} global configuration command to remove a static route. The device retains static routes until you remove them.

Default Routes and Networks

The following sections provides information about default routes and networks.

Information About Default Routes and Networks

A router might not be able to learn the routes to all other networks. To provide complete routing capability, you can use some routers as smart routers and give the remaining routers default routes to the smart router. (Smart routers have routing table information for the entire internetwork.) These default routes can be dynamically learned or can be configured in the individual routers. Most dynamic interior routing protocols include a mechanism for causing a smart router to generate dynamic default information that is then forwarded to other routers.

If a router has a directly connected interface to the specified default network, the dynamic routing protocols running on that device generate a default route. In RIP, it advertises the pseudonetwork 0.0.0.0.

A router that is generating the default for a network also might need a default of its own. One way a router can generate its own default is to specify a static route to the network 0.0.0.0 through the appropriate device.

When default information is passed through a dynamic routing protocol, no further configuration is required. The system periodically scans its routing table to choose the optimal default network as its default route. In IGRP networks, there might be several candidate networks for the system default. Cisco routers use administrative distance and metric information to set the default route or the gateway of last resort.

If dynamic default information is not being passed to the system, candidates for the default route are specified with the **ip default-network** global configuration command. If this network appears in the routing table from any source, it is flagged as a possible choice for the default route. If the router has no interface on the default network, but does have a path to it, the network is considered as a possible candidate, and the gateway to the best default path becomes the gateway of last resort.

How to Configure Default Routes and Networks

To configure default routes and networks, perform the following steps:

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 2	**ip default-network** *network number* **Example:** `Device(config)# ip default-network 1`	Specifies a default network.
Step 3	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode.
Step 4	**show ip route** **Example:**	Displays the selected default route in the gateway of last resort display.

	Command or Action	Purpose
	`Device# show ip route`	
Step 5	**copy running-config startup-config** **Example:** `Device# copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

Route Maps to Redistribute Routing Information

The following sections provide information about route maps to redistribute routing information.

Information About Route Maps

The switch can run multiple routing protocols simultaneously, and it can redistribute information from one routing protocol to another. Redistributing information from one routing protocol to another applies to all supported IP-based routing protocols.

You can also conditionally control the redistribution of routes between routing domains by defining enhanced packet filters or route maps between the two domains. The **match** and **set** route-map configuration commands define the condition portion of a route map. The **match** command specifies that a criterion must be matched. The **set** command specifies an action to be taken if the routing update meets the conditions defined by the match command. Although redistribution is a protocol-independent feature, some of the **match** and **set** route-map configuration commands are specific to a particular protocol.

One or more **match** commands and one or more **set** commands follow a **route-map** command. If there are no **match** commands, everything matches. If there are no **set** commands, nothing is done, other than the match. Therefore, you need at least one **match** or **set** command.

Note A route map with no **set** route-map configuration commands is sent to the CPU, which causes high CPU utilization.

You can also identify route-map statements as **permit** or **deny**. If the statement is marked as a deny, the packets meeting the match criteria are sent back through the normal forwarding channels (destination-based routing). If the statement is marked as permit, set clauses are applied to packets meeting the match criteria. Packets that do not meet the match criteria are forwarded through the normal routing channel.

How to Configure a Route Map

Although each of Steps 3 through 14 in the following section is optional, you must enter at least one **match** route-map configuration command and one **set** route-map configuration command.

Note The keywords are the same as defined in the procedure to control the route distribution.

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 2	**route-map***map-tag* [**permit** | **deny**] [*sequence number*] **Example:** `Device(config)# route-map rip-to-ospf permit 4`	Defines any route maps used to control redistribution and enter route-map configuration mode. *map-tag*—A meaningful name for the route map. The **redistribute** router configuration command uses this name to reference this route map. Multiple route maps might share the same map tag name. (Optional) If **permit** is specified and the match criteria are met for this route map, the route is redistributed as controlled by the set actions. If **deny** is specified, the route is not redistributed. *sequence number* (Optional)— Number that indicates the position a new route map is to have in the list of route maps already configured with the same name.
Step 3	**match as-path** *path-list-number* **Example:** `Device(config-route-map)# match as-path 10`	Matches a BGP AS path access list.
Step 4	**match community-list** *community-list-number* [**exact**] **Example:** `Device(config-route-map)# match community-list 150`	Matches a BGP community list.
Step 5	**match ip address** {*access-list-number* | *access-list-name*} [*...access-list-number* | *...access-list-name*] **Example:** `Device(config-route-map)# match ip address 5 80`	Matches a standard access list by specifying the name or number. It can be an integer from 1 to 199.
Step 6	**match metric** metric-value **Example:** `Device(config-route-map)# match metric 2000`	Matches the specified route metric. The *metric-value* can be an EIGRP metric with a specified value from 0 to 4294967295.
Step 7	**match ip next-hop** {*access-list-number* | *access-list-name*} [*...access-list-number* | *...access-list-name*] **Example:**	Matches a next-hop router address passed by one of the access lists specified (numbered from 1 to 199).

	Command or Action	Purpose
	`Device(config-route-map)# match ip next-hop 8 45`	
Step 8	**match tag** *tag value* [*...tag-value*] **Example:** `Device(config-route-map)# match tag 3500`	Matches the specified tag value in a list of one or more route tag values. Each can be an integer from 0 to 4294967295.
Step 9	**match interface***type number* [*...type-number*] **Example:** `Device(config-route-map)# match interface gigabitethernet 1/0/1`	Matches the specified next hop route out one of the specified interfaces.
Step 10	**match ip route-source** {*access-list-number* | *access-list-name*} [*...access-list-number* | *...access-list-name*] **Example:** `Device(config-route-map)# match ip route-source 10 30`	Matches the address specified by the specified advertised access lists.
Step 11	**match route-type** {**local** | **internal** | **external** [**type-1** | **type-2**]} **Example:** `Device(config-route-map)# match route-type local`	Matches the specified **route-type**: • **local**—Locally generated BGP routes. • **internal**—OSPF intra-area and interarea routes or EIGRP internal routes. • **external**—OSPF external routes (Type 1 or Type 2) or EIGRP external routes.
Step 12	**set dampening** *halflife reuse suppress max-suppress-time* **Example:** `Device(config-route-map)# set dampening 30 1500 10000 120`	Sets BGP route dampening factors.
Step 13	**set local-preference** *value* **Example:** `Device(config-route-map)# set local-preference 100`	Assigns a value to a local BGP path.
Step 14	**set origin** {**igp** | **egp** *as* | **incomplete**} **Example:** `Device(config-route-map)# set origin igp`	Sets the BGP origin code.

	Command or Action	Purpose
Step 15	**set as-path** {**tag** | **prepend** *as-path-string*} **Example:** `Device(config-route-map)# set as-path tag`	Modifies the BGP autonomous system path.
Step 16	**set level** {**level-1** | **level-2** | **level-1-2** | **stub-area** | **backbone**} **Example:** `Device(config-route-map)# set level level-1-2`	Sets the level for routes that are advertised into the specified area of the routing domain. The **stub-area** and **backbone** are OSPF NSSA and backbone areas.
Step 17	**set metric** *metric value* **Example:** `Device(config-route-map)# set metric 100`	Sets the metric value to give the redistributed routes (for EIGRP only). The *metric value* is an integer from -294967295 to 294967295.
Step 18	**set metric***bandwidth delay reliability loading mtu* **Example:** `Device(config-route-map)# set metric 10000 10 255 1 1500`	Sets the metric value to give the redistributed routes (for EIGRP only): • *bandwidth*—Metric value or IGRP bandwidth of the route in kilobits per second in the range 0 to 4294967295 • *delay*—Route delay in tens of microseconds in the range 0 to 4294967295. • *reliability*—Likelihood of successful packet transmission expressed as a number between 0 and 255, where 255 means 100 percent reliability and 0 means no reliability. • *loading*—Effective bandwidth of the route expressed as a number from 0 to 255 (255 is 100 percent loading). • *mtu*—Minimum maximum transmission unit (MTU) size of the route in bytes in the range 0 to 4294967295.
Step 19	**set metric-type** {**type-1** | **type-2**} **Example:** `Device(config-route-map)# set metric-type type-2`	Sets the OSPF external metric type for redistributed routes.
Step 20	**set metric-type internal** **Example:** `Device(config-route-map)# set metric-type internal`	Sets the multi-exit discriminator (MED) value on prefixes advertised to external BGP neighbor to match the IGP metric of the next hop.

	Command or Action	Purpose
Step 21	**set weight** *number* **Example:** `Device(config-route-map)# set weight 100`	Sets the BGP weight for the routing table. The value can be from 1 to 65535.
Step 22	**end** **Example:** `Device(config-route-map)# end`	Returns to privileged EXEC mode.
Step 23	**show route-map** **Example:** `Device# show route-map`	Displays all route maps configured or only the one specified to verify configuration.
Step 24	**copy running-config startup-config** **Example:** `Device# copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

How to Control Route Distribution

Although each of Steps 3 through 14 in the following section is optional, you must enter at least one **match** route-map configuration command and one **set** route-map configuration command.

Note The keywords are the same as defined in the procedure to configure the route map for redistritbution.

The metrics of one routing protocol do not necessarily translate into the metrics of another. For example, the RIP metric is a hop count, and the IGRP metric is a combination of five qualities. In these situations, an artificial metric is assigned to the redistributed route. Uncontrolled exchanging of routing information between different routing protocols can create routing loops and seriously degrade network operation.

If you have not defined a default redistribution metric that replaces metric conversion, some automatic metric translations occur between routing protocols:

- RIP can automatically redistribute static routes. It assigns static routes a metric of 1 (directly connected).
- Any protocol can redistribute other routing protocols if a default mode is in effect.

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:**	Enters global configuration mode.

	Command or Action	Purpose
	`Device# configure terminal`	
Step 2	**router** {**rip** | **ospf** | **eigrp**} **Example:** `Device(config)# router eigrp 10`	Enters router configuration mode.
Step 3	**redistribute** *protocol* [*process-id*] {**level-1** | **level-1-2** | **level-2**} [**metric** *metric-value*] [**metric-type** *type-value*] [**match internal** | **external** *type-value*] [**tag** *tag-value*] [**route-map** *map-tag*] [**weight** *weight*] [**subnets**] **Example:** `Device(config-router)# redistribute eigrp 1`	Redistributes routes from one routing protocol to another routing protocol. If no route-maps are specified, all routes are redistributed. If the keyword **route-map** is specified with no *map-tag*, no routes are distributed.
Step 4	**default-metric** *number* **Example:** `Device(config-router)# default-metric 1024`	Cause the current routing protocol to use the same metric value for all redistributed routes (RIP and OSPF).
Step 5	**default-metric** bandwidth delay reliability loading mtu **Example:** `Device(config-router)# default-metric 1000 100 250 100 1500`	Cause the EIGRP routing protocol to use the same metric value for all non-EIGRP redistributed routes.
Step 6	**end** **Example:** `Device(config-router)# end`	Returns to privileged EXEC mode.
Step 7	**show route-map** **Example:** `Device# show route-map`	Displays all route maps configured or only the one specified to verify configuration.
Step 8	**copy running-config startup-config** **Example:** `Device# copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

Policy-Based Routing

Restrictions for Configuring PBR

- Policy-based routing (PBR) is not supported to forward traffic into GRE tunnel. This applies to PBR applied on any interface and forwarding traffic into GRE tunnel (by means of PBR next-hop or default next-hop or set interface).
- PBR is not supported on GRE tunnel itself (applied under the GRE tunnel itself).
- Starting Cisco IOS XE Amsterdam 17.3.5 release, PBR does not apply to fragmented traffic. Fragmented traffic will follow a normal routing path.
- PBR and Network Address Translation (NAT) are not supported on the same interface. PBR and NAT work together only if they are configured on different interfaces.

Information About Policy-Based Routing

You can use policy-based routing (PBR) to configure a defined policy for traffic flows. By using PBR, you can have more control over routing by reducing the reliance on routes derived from routing protocols. PBR can specify and implement routing policies that allow or deny paths based on:

- Identity of a particular end system
- Application
- Protocol

You can use PBR to provide equal-access and source-sensitive routing, routing based on interactive versus batch traffic, or routing based on dedicated links. For example, you could transfer stock records to a corporate office on a high-bandwidth, high-cost link for a short time while transmitting routine application data such as e-mail over a low-bandwidth, low-cost link.

With PBR, you classify traffic using access control lists (ACLs) and then make traffic go through a different path. PBR is applied to incoming packets. All packets received on an interface with PBR enabled are passed through route maps. Based on the criteria defined in the route maps, packets are forwarded (routed) to the appropriate next hop.

- Route map statement marked as permit is processed as follows:
  - A match command can match on length or 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 length A B

    match ip address acl1 acl2

    match ip address acl3

  A packet is permitted if it is permitted by match length A B or 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.

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 application, a protocol type, or 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.

Local PBR configuration supports setting DSCP marking for RADIUS packets generated for device administration purposes.

How to Configure PBR

- To use PBR, you must have the Network Essentials license enabled on the standalone switch or active switch.
- Multicast traffic is not policy-routed. PBR applies only to unicast traffic.
- You can enable PBR on a routed port or an SVI.
- The switch supports PBR based on match length.
- You can apply a policy route map to an EtherChannel port channel in Layer 3 mode, but you cannot apply a policy route map to a physical interface that is a member of the EtherChannel. If you try to do so, the command is rejected. When a policy route map is applied to a physical interface, that interface cannot become a member of an EtherChannel.
- You can define a mazimum of 128 IP policy route maps on the switch or switch stack.
- You can define a maximum of 512 access control entries(ACEs) for PBR on the switch or switch stack.
- When configuring match criteria in a route map, follow these guidelines:
  - Do not match ACLs that permit packets destined for a local address.
- 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. The reverse is also true, you cannot enable PBR when WCCP is enabled on an interface.
- 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.
- PBR based on TOS, DSCP and IP Precedence are not supported.
- Set interface, set default next-hop and set default interface are not supported.
- **ip next-hop recursive** and **ip next-hop verify availability** features are not available and the next-hop should be directly connected.

- 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.

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, you must enable PBR for that route map on an interface. All packets arriving on the specified interface matching the match clauses are subject to PBR.

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.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**route-map** *map-tag* [**permit**] [*sequence number*] **Example:** `Device(config)# route-map pbr-map permit`	Defines route maps that are used to control where packets are output, and enters route-map configuration mode. • *map-tag* — A meaningful name for the route map. The **ip policy route-map** interface configuration command uses this name to reference the route map. Multiple route-map statements with the same map tag define a single 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.
Step 4	**match ip address** {*access-list-number* | *access-list-name*} [*access-list-number* |*...access-list-name*] **Example:** `Device(config-route-map)# match ip address 110 140`	Matches the source and destination IP addresses that are permitted by one or more standard or extended access lists. ACLs can match on more than one source and destination IP address. If you do not specify a **match** command, the route map is applicable to all packets.
Step 5	**match length min max** **Example:**	Matches the length of the packet.

	Command or Action	Purpose
	`Device(config-route-map)# match length 64 1500`	
Step 6	**set ip next-hop** *ip-address* [*...ip-address*] **Example:** `Device(config-route-map)# set ip next-hop 10.1.6.2`	Specifies the action to be taken on the packets that match the criteria. Sets next hop to which to route the packet (the next hop must be adjacent).
Step 7	**exit** **Example:** `Device(config-route-map)# exit`	Returns to global configuration mode.
Step 8	**interface** *interface-id* **Example:** `Device(config)# interface gigabitethernet 1/0/1`	Enters interface configuration mode, and specifies the interface to be configured.
Step 9	**ip policy route-map** *map-tag* **Example:** `Device(config-if)# ip policy route-map pbr-map`	Enables PBR on a Layer 3 interface, and identify the route map to use. 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.
Step 10	**ip route-cache policy** **Example:** `Device(config-if)# ip route-cache policy`	(Optional) Enables fast-switching PBR. You must enable PBR before enabling fast-switching PBR.
Step 11	**exit** **Example:** `Device(config-if)# exit`	Returns to global configuration mode.
Step 12	**ip local policy route-map** *map-tag* **Example:** `Device(config)# ip local policy route-map local-pbr`	(Optional) Enables local PBR to perform policy-based routing on packets originating at the switch. This applies to packets generated by the switch, and not to incoming packets.
Step 13	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode.
Step 14	**show route-map** [*map-name*] **Example:** `Device# show route-map`	(Optional) Displays all the route maps configured or only the one specified to verify configuration.
Step 15	**show ip policy** **Example:** `Device# show ip policy`	(Optional) Displays policy route maps attached to the interface.

	Command or Action	Purpose
Step 16	**show ip local policy** **Example:** `Device# show ip local policy`	(Optional) Displays whether or not local policy routing is enabled and, if so, the route map being used.

Filtering Routing Information

You can filter routing protocol information by performing the tasks described in this section.

Note When routes are redistributed between OSPF processes, no OSPF metrics are preserved.

Setting Passive Interfaces

To prevent other routers on a local network from dynamically learning about routes, you can use the **passive-interface** router configuration command to keep routing update messages from being sent through a router interface. When you use this command in the OSPF protocol, the interface address you specify as passive appears as a stub network in the OSPF domain. OSPF routing information is neither sent nor received through the specified router interface.

In networks with many interfaces, to avoid having to manually set them as passive, you can set all interfaces to be passive by default by using the **passive-interface default** router configuration command and manually setting interfaces where adjacencies are desired.

Use a network monitoring privileged EXEC command such as **show ip ospf interface** to verify the interfaces that you enabled as passive, or use the **show ip interface** privileged EXEC command to verify the interfaces that you enabled as active.

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 2	**router** {**rip** | **ospf** | **eigrp**} **Example:** `Device(config)# router ospf`	Enters router configuration mode.
Step 3	**passive-interface** *interface-id* **Example:**	Suppresses sending routing updates through the specified Layer 3 interface.

	Command or Action	Purpose
	`Device(config-router)# passive-interface gigabitethernet 1/0/1`	
Step 4	**passive-interface default** **Example:** `Device(config-router)# passive-interface default`	(Optional) Sets all interfaces as passive by default.
Step 5	**no passive-interface** *interface type* **Example:** `Device(config-router)# no passive-interface gigabitethernet1/0/3 gigabitethernet 1/0/5`	(Optional) Activates only those interfaces that need to have adjacencies sent.
Step 6	**network** *network-address* **Example:** `Device(config-router)# network 10.1.1.1`	(Optional) Specifies the list of networks for the routing process. The *network-address* is an IP address.
Step 7	**end** **Example:** `Device(config-router)# end`	Returns to privileged EXEC mode.
Step 8	**copy running-config startup-config** **Example:** `Device# copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

Controlling Advertising and Processing in Routing Updates

You can use the **distribute-list** router configuration command with access control lists to suppress routes from being advertised in routing updates and to prevent other routers from learning one or more routes. When used in OSPF, this feature applies to only external routes, and you cannot specify an interface name.

You can also use a **distribute-list** router configuration command to avoid processing certain routes listed in incoming updates. (This feature does not apply to OSPF.)

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password if prompted.

	Command or Action	Purpose
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**router** {**rip** | **eigrp**} **Example:** `Device(config)# router eigrp 10`	Enters router configuration mode.
Step 4	**distribute-list** {*access-list-number* | *access-list-name*} **out** [*interface-name* | *routing process* | *autonomous-system-number*] **Example:** `Device(config-router)# distribute 120 out gigabitethernet 1/0/7`	Permits or denies routes from being advertised in routing updates, depending upon the action listed in the access list.
Step 5	**distribute-list** {*access-list-number* | *access-list-name*} **in** [*type-number*] **Example:** `Device(config-router)# distribute-list 125 in`	Suppresses processing in routes listed in updates.
Step 6	**end** **Example:** `Device(config-router)# end`	Returns to privileged EXEC mode.
Step 7	**copy running-config startup-config** **Example:** `Device# copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

Filtering Sources of Routing Information

Because some routing information might be more accurate than others, you can use filtering to prioritize information coming from different sources. An administrative distance is a rating of the trustworthiness of a routing information source, such as a router or group of routers. In a large network, some routing protocols can be more reliable than others. By specifying administrative distance values, you enable the router to intelligently discriminate between sources of routing information. The router always picks the route whose routing protocol has the lowest administrative distance.

Because each network has its own requirements, there are no general guidelines for assigning administrative distances.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**router** {**rip** | **ospf** | **eigrp**} **Example:** `Device(config)# router eigrp 10`	Enters router configuration mode.
Step 4	**distance** *weight* {*ip-address* {*ip-address mask*}} [*ip access list*] **Example:** `Device(config-router)# distance 50 10.1.5.1`	Defines an administrative distance. *weight*—The administrative distance as an integer from 10 to 255. Used alone, *weight* specifies a default administrative distance that is used when no other specification exists for a routing information source. Routes with a distance of 255 are not installed in the routing table. (Optional) *ip access list*—An IP standard or extended access list to be applied to incoming routing updates.
Step 5	**end** **Example:** `Device(config-router)# end`	Returns to privileged EXEC mode.
Step 6	**show ip protocols** **Example:** `Device# show ip protocols`	Displays the default administrative distance for a specified routing process.
Step 7	**copy running-config startup-config** **Example:** `Device# copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

Managing Authentication Keys

Key management is a method of controlling authentication keys used by routing protocols. Not all protocols can use key management. Authentication keys are available for EIGRP and RIP Version 2.

Prerequisites

Before you manage authentication keys, you must enable authentication. See the appropriate protocol section to see how to enable authentication for that protocol. To manage authentication keys, define a key chain, identify the keys that belong to the key chain, and specify how long each key is valid. Each key has its own key identifier (specified with the **key** *number* key chain configuration command), which is stored locally. The combination of the key identifier and the interface associated with the message uniquely identifies the authentication algorithm and Message Digest 5 (MD5) authentication key in use.

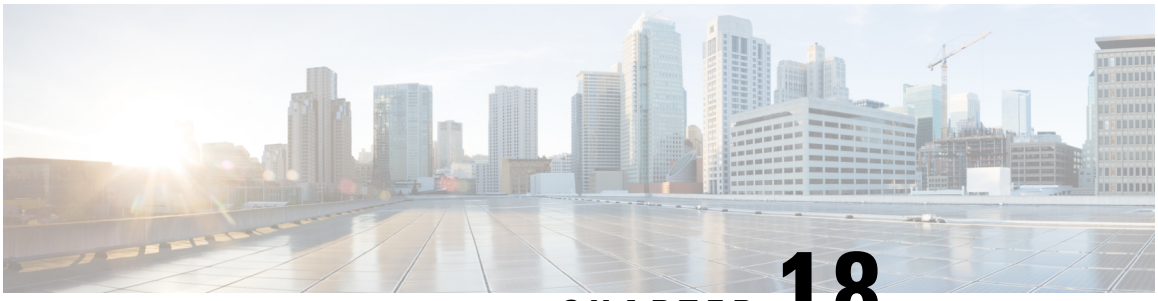
How to Configure Authentication Keys

You can configure multiple keys with life times. Only one authentication packet is sent, regardless of how many valid keys exist. The software examines the key numbers in order from lowest to highest, and uses the first valid key it encounters. The lifetimes allow for overlap during key changes. Note that the router must know these lifetimes.

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 2	**key chain** *name-of-chain* **Example:** `Device(config)# key chain key10`	Identifies a key chain, and enter key chain configuration mode.
Step 3	**key** *number* **Example:** `Device(config-keychain)# key 2000`	Identifies the key number. The range is 0 to 2147483647.
Step 4	**key-string** *text* **Example:** `Device(config-keychain)# Room 20, 10th floor`	Identifies the key string. The string can contain from 1 to 80 uppercase and lowercase alphanumeric characters, but the first character cannot be a number.
Step 5	**accept-lifetime** *start-time* {**infinite** | *end-time* | **duration** *seconds*} **Example:** `Device(config-keychain)# accept-lifetime 12:30:00 Jan 25 1009 infinite`	(Optional) Specifies the time period during which the key can be received. The *start-time* and *end-time* syntax can be either *hh:mm:ss Month date year* or *hh:mm:ss date Month year*. The default is forever with the default *start-time* and the earliest acceptable date as January 1, 1993. The default *end-time* and **duration** is **infinite**.
Step 6	**send-lifetime** *start-time* {**infinite** | *end-time* | **duration** *seconds*}	(Optional) Specifies the time period during which the key can be sent.

	Command or Action	Purpose
	Example: `Device(config-keychain)# accept-lifetime 23:30:00 Jan 25 1019 infinite`	The *start-time* and *end-time* syntax can be either *hh:mm:ss Month date year* or *hh:mm:ss date Month year*. The default is forever with the default *start-time* and the earliest acceptable date as January 1, 1993. The default *end-time* and **duration** is **infinite**.
Step 7	**end** **Example:** `Device(config-keychain)# end`	Returns to privileged EXEC mode.
Step 8	**show key chain** **Example:** `Device# show key chain`	Displays authentication key information.
Step 9	**copy running-config startup-config** **Example:** `Device# copy running-config startup-config`	(Optional) Saves your entries in the configuration file.



CHAPTER 18

Configuring VRF-lite

- Information About VRF-lite, on page 309
- Guidelines for Configuring VRF-lite, on page 310
- How to Configure VRF-lite, on page 312
- Additional Information for VRF-lite, on page 332
- Verifying VRF-lite Configuration, on page 332
- Configuration Examples for VRF-lite, on page 333
- Additional References for VRF-Lite, on page 337
- Feature History and Information for Multicast VRF-lite, on page 337

Information About VRF-lite

VRF-lite is a feature that enables a service provider to support two or more VPNs, where IP addresses can be overlapped among the VPNs. VRF-lite uses input interfaces to distinguish routes for different VPNs and forms virtual packet-forwarding tables by associating one or more Layer 3 interfaces with each VRF. Interfaces in a VRF can be either physical, such as Ethernet ports, or logical, such as VLAN SVIs, but a Layer 3 interface cannot belong to more than one VRF at any time.

Note VRF-lite interfaces must be Layer 3 interfaces.

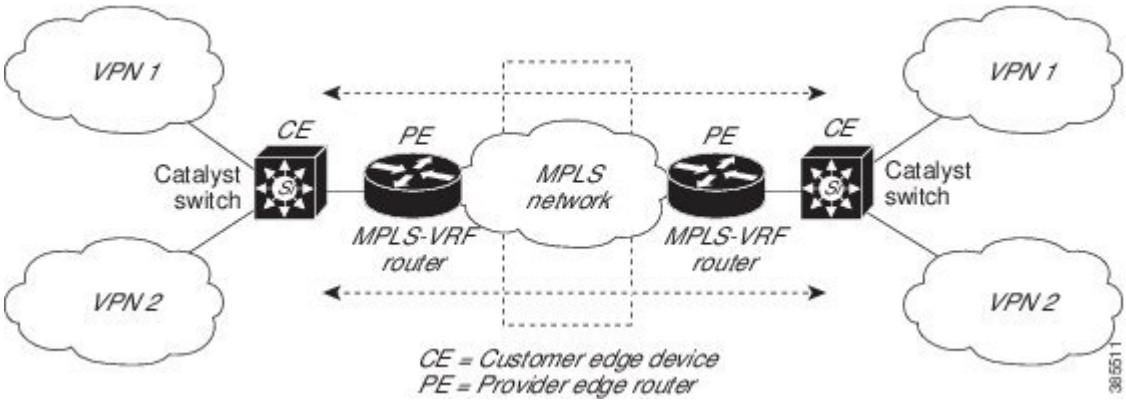
VRF-lite includes these devices:

- Customer edge (CE) devices provide customer access to the service provider network over a data link to one or more provider edge routers. The CE device advertises the site's local routes to the provider edge router and learns the remote VPN routes from it. A Cisco Catalyst Switch can be a CE.
- Provider routers (or core routers) are any routers in the service provider network that do not attach to CE devices.

With VRF-lite, multiple customers can share one CE, and only one physical link is used between the CE and the PE. The shared CE maintains separate VRF tables for each customer and switches or routes packets for each customer based on its own routing table. VRF-lite extends limited PE functionality to a CE device, giving it the ability to maintain separate VRF tables to extend the privacy and security of a VPN to the branch office.

The following figure displays a configuration where each Cisco Catalyst switch acts as multiple virtual CEs. Because VRF-lite is a Layer 3 feature, each interface in a VRF must be a Layer 3 interface.

Figure 11: Cisco Catalyst Switches Acting as Multiple Virtual CEs



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

- 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.
- 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.
- 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.
- 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. You then configure the routing protocols in the VPN and between the CE and the PE. BGP is the preferred routing protocol used to distribute VPN routing information across the providers' backbone. The VRF-lite network has 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

IPv4 and IPv6

- 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 Information About VRF-lite, on page 309, 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 as long because they do not overlap with those of other customers. A customer's VLANs are mapped to a specific routing table ID that is used to identify the appropriate routing tables stored on the switch.
- The Layer 3 TCAM resource is shared between all VRFs. To ensure that any one VRF has sufficient CAM space, use the **maximum routes** command.
- 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

- 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 makes simplifies passing attributes of the routes to the CE.
- The Cisco Catalyst switch supports PIM-SM and PIM-SSM protocols.
- 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

- VRF-aware OSPFv3, BGPv6, EIGRPv6, and IPv6 static routing are supported.
- VRF-aware IPv6 route applications include: ping, telnet, ssh, tftp, ftp and traceroute. (This list does not include the management interface, which is handled differently even though you can configure both IPv4 or IPv6 VRF under it.)

How to Configure VRF-lite

This section provides information about configuring VRF-lite.

Configuring VRF-lite for IPv4

This section provides information about configuring VRF-lite for IPv4.

Configuring VRF-Aware Services

IP services can be configured on global interfaces and within the global routing instance. IP services are enhanced to run on multiple routing instances; they are VRF-aware. Any configured VRF in the system can be specified for a VRF-aware service.

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 the following 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.

Configuring the User Interface for ARP

Procedure

	Command or Action	Purpose
Step 1	**show ip arp vrf** *vrf-name* **Example:** `Device# show ip arp vrf vrf-name`	Displays the ARP table (static and dynamic entries) in the specified VRF.
Step 2	**arp vrf** *vrf-name ip-address mac-address ARPA* **Example:** `Device(config)# arp vrf vrf-name ip-address mac-address ARPA`	Creates a static ARP entry in the specified VRF.

Configuring Per-VRF for TACACS+ Servers

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

You can create the VRF routing table (shown in Steps 3 and 4) and configure the interface (Steps 6, 7, and 8). The actual configuration of per-VRF on a TACACS+ server is done in Steps 10 through 13.

Before you begin

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

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**vrf definition** *vrf-name* **Example:** `Device(config)# vrf definition vrf-name`	Configures a VRF table and enters VRF configuration mode.
Step 4	**rd** *route-distinguisher* **Example:** `Device(config-vrf)# rd route-distinguisher`	Creates routing and forwarding tables for a VRF instance.
Step 5	**exit** **Example:** `Device(config-vrf)# exit`	Exits VRF configuration mode.
Step 6	**interface** *interface-name* **Example:** `Device(config)# interface interface-name`	Configures an interface and enters interface configuration mode.
Step 7	**vrf forwarding** *vrf-name* **Example:** `Device(config-if)# vrf forwarding vrf-name`	Configures a VRF for the interface.
Step 8	**ip address** *ip-address mask* [**secondary**] **Example:** `Device(config-if)# ip address ip-address mask [secondary]`	Sets a primary or secondary IP address for an interface.
Step 9	**exit** **Example:** `Device(config-vrf)# exit`	Exits interface configuration mode.
Step 10	**aaa group server tacacs+** *group-name* **Example:** `Device(config)# aaa group server tacacs+ tacacs1`	Groups different TACACS+ server hosts into distinct lists and distinct methods and enters server-group configuration mode.
Step 11	**server-private** {*ip-address* | *name*} [**nat**] [**single-connection**] [**port** *port-number*] [**timeout** *seconds*] [**key** [**0** | **7**] *string*]	Configures the IP address of the private TACACS+ server for the group server.

	Command or Action	Purpose
	Example: `Device(config-sg-tacacs+)# server-private 10.1.1.1 port 19 key cisco`	
Step 12	**vrf forwarding** *vrf-name* **Example:** `Device(config-sg-tacacs+)# vrf forwarding vrf-name`	Configures the VRF reference of a AAA TACACS+ server group.
Step 13	**ip tacacs source-interface** *subinterface-name* **Example:** `Device(config-sg-tacacs+)# ip tacacs source-interface subinterface-name`	Uses the IP address of a specified interface for all outgoing TACACS+ packets.
Step 14	**exit** **Example:** `Device(config-sg-tacacs)# exit`	Exits server-group configuration mode.

Example

The following example lists all the steps to configure per-VRF TACACS+:

```
Device> enable
Device# configure terminal
Device(config)# vrf definition cisco
Device(config-vrf)# rd 100:1
Device(config-vrf)# exit
Device(config)# interface Loopback0
Device(config-if)# vrf forwarding cisco
Device(config-if)# ip address 10.0.0.2 255.0.0.0
Device(config-if)# exit
Device(config-sg-tacacs+)# vrf forwarding cisco
Device(config-sg-tacacs+)# ip tacacs source-interface Loopback0
Device(config-sg-tacacs)# exit
```

Configuring Multicast VRFs

SUMMARY STEPS

1. **configure terminal**
2. **ip routing**
3. **vrf definition** *vrf-name*
4. **ip multicast-routing vrf** *vrf-name*
5. **rd** *route-distinguisher*
6. **route-target** {**export** | **import** | **both**} *route-target-ext-community*
7. **import map** *route-map*
8. **interface** *interface-id*
9. **vrf forwarding** *vrf-name*
10. **ip address** *ip-address* **mask**

11. **ip pim sparse-mode**
12. **end**
13. **show vrf definition** [**brief** | **detail** | **interfaces**] [*vrf-name*]
14. **copy running-config startup-config**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 2	**ip routing** **Example:** `Device(config)# ip routing`	Enables IP routing.
Step 3	**vrf definition** *vrf-name* **Example:** `Device(config)# vrf definition vrf-name`	Configures a VRF table and enters VRF configuration mode.
Step 4	**ip multicast-routing vrf** *vrf-name* **Example:** `Device(config-vrf)# ip multicast-routing vrf vrf-name`	(Optional) Enables global multicast routing for VRF table.
Step 5	**rd** *route-distinguisher* **Example:** `Device(config-vrf)# rd route-distinguisher`	Creates a VRF table by specifying a route distinguisher. Enter either an AS number and an arbitrary number (xxx:y) or an IP address and arbitrary number (A.B.C.D:y).
Step 6	**route-target** {**export** | **import** | **both**} *route-target-ext-community* **Example:** `Device(config-vrf)# route-target {export | import | both} route-target-ext-community`	Creates a list of import, export, or import and export route target communities for the specified VRF. 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). The route-target-ext-community value should be the same as the route-distinguisher value entered in Step 4.
Step 7	**import map** *route-map* **Example:** `Device(config-vrf)# import map route-map`	(Optional) Associates a route map with the VRF.
Step 8	**interface** *interface-id* **Example:** `Device(config)# interface interface-id`	Enters interface configuration mode and specifies the Layer 3 interface to be associated with the VRF. The interface can be a routed port or a SVI.
Step 9	**vrf forwarding** *vrf-name* **Example:**	Associates the VRF with the Layer 3 interface.

	Command or Action	Purpose
	`Device(config-if)# vrf forwarding vrf-name`	
Step 10	**ip address** *ip-address* **mask** **Example:** `Device(config-if)# ip address ip-address mask`	Configures IP address for the Layer 3 interface.
Step 11	**ip pim sparse-mode** **Example:** `Device(config-if)# ip pim sparse-mode`	Enables PIM on the VRF-associated Layer 3 interface.
Step 12	**end** **Example:** `Device(config-if)# end`	Returns to privileged EXEC mode.
Step 13	**show vrf definition** [**brief** | **detail** | **interfaces**] [*vrf-name*] **Example:** `Device# show vrf definition brief`	Verifies the configuration. Display information about the configured VRFs.
Step 14	**copy running-config startup-config** **Example:** `Device# copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

Example

The following example shows how to configure multicast within a VRF table:

```
Device(config)# ip routing
Device(config)# vrf definition multiVrfA
Device(config-vrf)# ip multicast-routing vrf multiVrfA
Device(config-vrf)# interface GigabitEthernet3/1/0
Device(config-if)# vrf forwarding multiVrfA
Device(config-if)# ip address 172.21.200.203 255.255.255.0
Device(config-if)# ip pim sparse-mode
```

Configuring a VPN Routing Session

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 2	**router ospf** *process-id* **vrf vrf-name** **Example:**	Enables OSPF routing, specifies a VPN forwarding table, and enters router configuration mode.

	Command or Action	Purpose
	`Device(config)# router ospf process-id vrf vrf-name`	
Step 3	**log-adjacency-changes** **Example:** `Device(config-router)# log-adjacency-changes`	(Optional) Logs changes in the adjacency state (the default state).
Step 4	**redistribute bgp** *autonomous-system-number subnets* **Example:** `Device(config-router)# redistribute bgp autonomous-system-number subnets`	Sets the switch to redistribute information from the BGP network to the OSPF network.
Step 5	**network** *network-number* **area** *area-id* **Example:** `Device(config-router)# network network-number area area-id`	Defines a network address and mask on which OSPF runs and the area ID for that network address.
Step 6	**end** **Example:** `Device(config-router)# end`	Returns to privileged EXEC mode.
Step 7	**show ip ospf** *process-id* **Example:** `Device# show ip ospf process-id`	Verifies the configuration of the OSPF network.
Step 8	copy running-config startup-config **Example:** `Device# copy running-config startup-config`	(Optional) Saves your entries in the configuration file. Use the **no router ospf** *process-id* **vrf** *vrf-name* global configuration command to disassociate the VPN forwarding table from the OSPF routing process.

Example

```
Device(config)# vrf definition VRF-RED
Device(config-vrf)# rd 1:1
Device(config-vrf)# exit
Device(config)# router eigrp virtual-name
Device(config-router)# address-family ipv4 vrf VRF-RED autonomous-system 1
Device(config-router-af)# network 10.0.0.0 0.0.0.255
Device(config-router-af)# topology base
Device(config-router-topology)# default-metric 10000 100 255 1 1500
Device(config-router-topology)# exit-af-topology
Device(config-router-af)# exit-address-family
```

Configuring BGP PE to CE Routing Sessions

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 2	**router bgp** *autonomous-system-number* **Example:** `Device(config)# router bgp autonomous-system-number`	Configures the BGP routing process with the AS number passed to other BGP routers and enters router configuration mode.
Step 3	**network** *network-number* **mask** *network-mask* **Example:** `Device(config-router)# network network-number mask network-mask`	Specifies a network and mask to announce using BGP.
Step 4	**redistribute ospf** *process-id match internal* **Example:** `Device(config-router)# redistribute ospf process-id match internal`	Sets the switch to redistribute OSPF internal routes.
Step 5	**network** *network-number* **area** *area-id* **Example:** `Device(config-router)# network network-number area area-id`	Defines a network address and mask on which OSPF runs and the area ID for that network address.
Step 6	**address-family ipv4 vrf** *vrf-name* **Example:** `Device(config-router-af)# address-family ipv4 vrf vrf-name`	Defines BGP parameters for PE to CE routing sessions and enters VRF address-family mode.
Step 7	**neighbor** *address* **remote-as** *as-number* **Example:** `Device(config-router-af)# neighbor address remote-as as-number`	Defines a BGP session between PE and CE routers.
Step 8	**neighbor** *address* **activate** **Example:** `Device(config-router-af)# neighbor address activate`	Activates the advertisement of the IPv4 address family.
Step 9	**end** **Example:** `Device(config-router-af)# end`	Returns to privileged EXEC mode.

	Command or Action	Purpose
Step 10	**show ip bgp** [**ipv4**] [**neighbors**] **Example:** `Device# show ip bgp [ipv4] [neighbors]`	Verifies BGP configuration. Use the **no router bgp** *autonomous-system-number* global configuration command to delete the BGP routing process. Use the command with keywords to delete routing characteristics.

Configuring IPv4 VRFs

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 2	**ip routing** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**vrf definition** *vrf-name* **Example:** `Device(config)# vrf definition vrf-name`	Names the VRF and enters VRF configuration mode.
Step 4	**rd** *route-distinguisher* **Example:** `Device(config-vrf)# rd route-distinguisher`	Creates a VRF table by specifying a route distinguisher. 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).
Step 5	**route-target** {**export** | **import** | **both**} *route-target-ext-community* **Example:** `Device(config-vrf)# route-target {export | import | both} route-target-ext-community`	Creates a list of import, export, or import and export route target communities for the specified VRF. 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.
Step 6	**import map** *route-map* **Example:** `Device(config-vrf)# import map route-map`	(Optional) Associates a route map with the VRF.
Step 7	**interface** *interface-id* **Example:** `Device(config-vrf)# interface interface-id`	Enters interface configuration mode and specify the Layer 3 interface to be associated with the VRF. The interface can be a routed port or SVI.

	Command or Action	Purpose
Step 8	**vrf forwarding** *vrf-name* **Example:** `Device(config-if)# vrf forwarding vrf-name`	Associates the VRF with the Layer 3 interface.
Step 9	**end** **Example:** `Device(config-if)# end`	Returns to privileged EXEC mode.
Step 10	**show vrf definition** [**brief** | **detail** | **interfaces**] [*vrf-name*] **Example:** `Device# show vfr definition [brief | detail | interfaces] [vrf-name]`	Verifies the configuration. Displays information about the configured VRFs.
Step 11	**copy running-config startup-config** **Example:** `Device# copy running-config startup-config`	(Optional) Saves your entries in the configuration file. 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.

Configuring VRF-lite for IPv6

This section provides information about configuring VRF-lite for IPv6.

Configuring VRF-Aware Services

IPv6 services can be configured on global interfaces and within the global routing instance. IPv6 services are enhanced to run on multiple routing instances; they are VRF-aware. Any configured VRF in the system can be specified for a VRF-aware service.

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 the following 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.

The following services are VRF-aware:

- Ping
- Unicast Reverse Path Forwarding (uRPF)
- Traceroute
- FTP and TFTP
- Telnet and SSH

- NTP

Configuring the User Interface for PING

Perform the following task to configure a VRF-aware ping:

Procedure

	Command or Action	Purpose
Step 1	**ping vrf** *vrf-name* **ipv6-host** **Example:** `Device# ping vrf vrf-name ipv6-host`	Pings an IPv6 host or address in the specified VRF.

Configuring the User Interface for uRPF

You can configure uRPF on an interface assigned to a VRF. Source lookup is performed in the VRF table

SUMMARY STEPS

1. **configure terminal**
2. **interface** *interface-id*
3. **no switchport**
4. **vrf forwarding** *vrf-name*
5. **ipv6 address** *ip-address* **subnet-mask**
6. **ipv6 verify unicast source reachable-via rx allow-default**
7. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 2	**interface** *interface-id* **Example:** `Device(config)# interface interface-id`	Enters interface configuration mode and specifies the Layer 3 interface to configure.
Step 3	**no switchport** **Example:** `Device(config-if)# no switchport`	Removes the interface from Layer 2 configuration mode if it is a physical interface.
Step 4	**vrf forwarding** *vrf-name* **Example:** `Device(config-if)# vrf forwarding vrf-name`	Configures VRF on the interface.

	Command or Action	Purpose
Step 5	**ipv6 address** *ip-address* **subnet-mask** **Example:** `Device(config-if)# ip address ip-address mask`	Enters the IPv6 address for the interface.
Step 6	**ipv6 verify unicast source reachable-via rx allow-default** **Example:** `Device(config-if)# ipv6 verify unicast source reachable-via rx allow-default`	Enables uRPF on the interface.
Step 7	**end** **Example:** `Device(config-if)# end`	Returns to privileged EXEC mode.

Configuring the User Interface for Traceroute

SUMMARY STEPS

1. **traceroute vrf** *vrf-name ipv6address*

DETAILED STEPS

	Command or Action	Purpose
Step 1	**traceroute vrf** *vrf-name ipv6address* **Example:** `Device# traceroute vrf vrf-name ipv6address`	Specifies the name of a VPN VRF in which to find the destination address.

Configuring the User Interface for Telnet and SSH

SUMMARY STEPS

1. **telnet** *ipv6-address*/**vrf** *vrf-name*
2. **ssh -l** *username* **-vrf** *vrf-name* **ipv6-host**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**telnet** *ipv6-address*/**vrf** *vrf-name* **Example:** `Device# telnet ipv6-address/vrf vrf-name`	Connects through Telnet to an IPv6 host or address in the specified VRF.
Step 2	**ssh -l** *username* **-vrf** *vrf-name* **ipv6-host** **Example:** `Device# ssh -l username -vrf  vrf-name ipv6-host`	Connects through SSH to an IPv6 host or address in the specified VRF.

Configuring the User Interface for NTP

SUMMARY STEPS

1. **configure terminal**
2. **ntp server vrf** *vrf-name* **ipv6-host**
3. **ntp peer vrf** *vrf-name* **ipv6-host**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 2	**ntp server vrf** *vrf-name* **ipv6-host** **Example:** `Device(config)# ntp server vrf vrf-name ipv6-host`	Configure the NTP server in the specified VRF.
Step 3	**ntp peer vrf** *vrf-name* **ipv6-host** **Example:** `Device(config)# ntp peer vrf vrf-name ipv6-host`	Configure the NTP peer in the specified VRF.

Configuring IPv6 VRFs

SUMMARY STEPS

1. **configure terminal**
2. **vrf definition** *vrf-name*
3. **rd** *route-distinguisher*
4. **address-family** *ipv4* | *ipv6*
5. **route-target** {**export** | **import** | **both**} *route-target-ext-community*
6. **exit-address-family**
7. **vrf definition** *vrf-name*
8. **ipv6 multicast multitopology**
9. **address-family ipv6 multicast**
10. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.

	Command or Action	Purpose
Step 2	**vrf definition** *vrf-name* **Example:** `Device(config)# vrf definition vrf-name`	Names the VRF and enters VRF configuration mode.
Step 3	**rd** *route-distinguisher* **Example:** `Device(config-vrf)# rd route-distinguisher`	(Optional) Creates a VRF table by specifying a route distinguisher. Enter either an Autonomous System number and an arbitrary number (xxx:y) or an IP address and arbitrary number (A.B.C.D:y).
Step 4	**address-family** *ipv4* | *ipv6* **Example:** `Device(config-vrf)# address-family ipv4 | ipv6`	(Optional) IPv4 by default. Configuration MUST for IPv6.
Step 5	**route-target** {**export** | **import** | **both**} *route-target-ext-community* **Example:** `Device(config-vrf)# route-target {export | import | both} route-target-ext-community`	Creates a list of import, export, or import and export route target communities for the specified VRF. 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.
Step 6	**exit-address-family** **Example:** `Device(config-vrf)# exit-address-family`	Exits VRF address-family configuration mode and return to VRF configuration mode.
Step 7	**vrf definition** *vrf-name* **Example:** `Device(config)# vrf definition vrf-name`	Enters VRF configuration mode.
Step 8	**ipv6 multicast multitopology** **Example:** `Device(config-vrf-af)# ipv6 multicast multitopology`	Enables multicast specific RPF topology.
Step 9	**address-family ipv6 multicast** **Example:** `Device(config-vrf)# address-family ipv6 multicast`	Enter multicast IPv6 address-family.
Step 10	**end** **Example:** `Device(config-vrf-af)# end`	Returns to privileged EXEC mode.

Example

This example shows how to configure VRFs:

```
Device(config)#  vrf definition red
Device(config-vrf)# rd 100:1
Device(config-vrf)# address family ipv6
Device(config-vrf-af)# route-target both 200:1
Device(config-vrf)# exit-address-family
Device(config-vrf)# vrf definition red
Device(config-vrf)# ipv6 multicast multitopology
Device(config-vrf)# address-family ipv6 multicast
Device(config-vrf-af)# end
```

Associating Interfaces to the Defined VRFs

SUMMARY STEPS

1. **interface** *interface-id*
2. **no switchport**
3. **vrf forwarding** *vrf-name*
4. **ipv6 enable**
5. **ipv6 address** *ip-address subnet-mask*
6. **show ipv6 vrf** [**brief** | **detail** | **interfaces**] [*vrf-name*]
7. **copy running-config startup-config**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**interface** *interface-id* **Example:** `Device(config-vrf)# interface interface-id`	Enters interface configuration mode and specify the Layer 3 interface to be associated with the VRF. The interface can be a routed port or SVI.
Step 2	**no switchport** **Example:** `Device(config-if)# no switchport`	Removes the interface from configuration mode if it is a physical interface.
Step 3	**vrf forwarding** *vrf-name* **Example:** `Device(config-if)# vrf forwarding vrf-name`	Associates the VRF with the Layer 3 interface.
Step 4	**ipv6 enable** **Example:** `Device(config-if)# ipv6 enable`	Enable IPv6 on the interface.
Step 5	**ipv6 address** *ip-address subnet-mask* **Example:** `Device(config-if)# ipv6 address ip-address subnet-mask`	Enters the IPv6 address for the interface.
Step 6	**show ipv6 vrf** [**brief** | **detail** | **interfaces**] [*vrf-name*] **Example:**	Verifies the configuration. Displays information about the configured VRFs.

	Command or Action	Purpose
	`Device# show ipv6 vrf [brief | detail | interfaces] [vrf-name]`	
Step 7	**copy running-config startup-config** **Example:** `Device# copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

Example

This example shows how to associate an interface to VRFs:

```
Switch(config-vrf)# interface ethernet0/1
Switch(config-if)# vrf forwarding red
Switch(config-if)# ipv6 enable
Switch(config-if)# ipv6 address 5000::72B/64
```

Populate VRF with Routes via Routing Protocols

This section provides information about populating VRF with routes via routing protocols.

Configuring VRF Static Routes

SUMMARY STEPS

1. **configure terminal**
2. **ipv6 route** [**vrf** *vrf-name*] *ipv6-prefix*/*prefix-length* {*ipv6-address* | **interface-type** *interface-number* [*ipv6-address*]}

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 2	**ipv6 route** [**vrf** *vrf-name*] *ipv6-prefix*/*prefix-length* {*ipv6-address* | **interface-type** *interface-number* [*ipv6-address*]} **Example:** `Device(config)# ipv6 route [vrf vrf-name] ipv6-prefix/prefix-length {ipv6-address | interface-type interface-number [ipv6-address]}`	To configure static routes specific to VRF.

Example

```
Device(config)# ipv6 route vrf v6a 7000::/64 TenGigabitEthernet32 4000::2
```

Configuring OSPFv3 Router Process

SUMMARY STEPS

1. **configure terminal**
2. **router ospfv3** *process-id*
3. **area** *area-ID* [**default-cot** | **nssa** | **stub**]
4. **router-id** *router-id*
5. **address-family ipv6 unicast vrf** *vrf-name*
6. **redistribute source-protocol** [*process-id*] **options**
7. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 2	**router ospfv3** *process-id* **Example:** `Device(config)# router ospfv3 process-id`	Enables OSPFv3 router configuration mode for the IPv6 address family.
Step 3	**area** *area-ID* [**default-cot**	**nssa**	**stub**] **Example:** `Device(config-router)# area area-ID [default-cot | nssa | stub]`	Configures the OSPFv3 area.
Step 4	**router-id** *router-id* **Example:** `Device(config-router)# router-id router-id`	Use a fixed router ID.
Step 5	**address-family ipv6 unicast vrf** *vrf-name* **Example:** `Device(config-router)# address-family ipv6 unicast vrf vrf-name`	Enters IPv6 address family configuration mode for OSPFv3 in VRF vrf-name
Step 6	**redistribute source-protocol** [*process-id*] **options** **Example:** `Device(config-router)# redistribute source-protocol [process-id] options`	Redistributes IPv6 routes from one routing domain into another routing domain.
Step 7	**end** **Example:** `Device(config-router)# end`	Returns to privileged EXEC mode.

Example

This example shows how configure the OSPFv3 router process:

```
Device(config-router)# router ospfv3 1
Device(config-router)# router-id 1.1.1.1
Device(config-router)# address-family ipv6 unicast
Device(config-router-af)# exit-address-family
```

Enabling OSPFv3 on an Interface

SUMMARY STEPS

1. **configure terminal**
2. **interface** *type-number*
3. **ospfv3** *process-id* **area** *area-ID* **ipv6** [**instance** *instance-id*]
4. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 2	**interface** *type-number* **Example:** `Device(config-vrf)# interface type-number`	Specifies an interface type and number, and places the switch in interface configuration mode.
Step 3	**ospfv3** *process-id* **area** *area-ID* **ipv6** [**instance** *instance-id*] **Example:** `Device(config-if)# ospfv3 process-id area area-ID ipv6 [instance instance-id]`	Enables OSPFv3 on an interface with IPv6 AF.
Step 4	**end** **Example:** `Device(config-if)# end`	Returns to privileged EXEC mode.

Example

This example show how to enable OSPFv3 on an interface:

```
Device(config)# interface GigabitEthernet2/1
Device(config-if)# no switchport
Device(config-if)# ipv6 address 4000::2/64
Device(config-if)# ipv6 enable
Device(config-if)# ipv6 ospf 1 area 0
Device(config-if)# end
```

Configuring EIGRPv6 Routing Process

SUMMARY STEPS

1. **configure terminal**
2. **router eigrp** *virtual-instance-name*
3. **address-family ipv6 vrf** *vrf-name* **autonomous-system** *autonomous-system-number*
4. **topology** {**base** | **topology-name tid number**
5. **exit-aftopology**
6. **eigrp router-id** *ip-address*
7. end

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 2	**router eigrp** *virtual-instance-name* **Example:** `Device(config)# router eigrp virtual-instance-name`	Configures the EIGRP routing process and enters router configuration mode.
Step 3	**address-family ipv6 vrf** *vrf-name* **autonomous-system** *autonomous-system-number* **Example:** `Device(config-router)# address-family ipv6 vrf vrf-name autonomous-system autonomous-system-number`	Enables EIGRP IPv6 VRF-Lite and enters address family configuration mode.
Step 4	**topology** {**base**	**topology-name tid number** **Example:** `Device(config-router-af)# topology {base | topology-name tid number`	Configures an EIGRP process to route IP traffic under the specified topology instance and enters address family topology configuration mode.
Step 5	**exit-aftopology** **Example:** `Device(config-router-af-topology)# exit-aftopology`	Exits address family topology configuration mode.
Step 6	**eigrp router-id** *ip-address* **Example:** `Device(config-router)# eigrp router-id ip-address`	Enables the use of a fixed router-id.
Step 7	end **Example:** `Device(config-router)# end`	Exits router configuration mode.

Example

This example shows how to configure an EIGRP routing process:

```
Device(config)# router eigrp test
Device(config-router)# address-family ipv6 unicast vrf b1 autonomous-system 10
Device(config-router-af)# topology base
Device(config-router-af-topology)# exit-af-topology
Device(config-router)# eigrp router-id 2.3.4.5
Device(config-router)# exit-address-family
```

Configuring EBGPv6 Routing Process

SUMMARY STEPS

1. **configure terminal**
2. **router bgp** *as-number*
3. **neighbor** *peer-group-name peer-group*
4. **neighbor** {*ip-address* | *ipv6-address*[*%*] | *peer-group-name*}**remote-as** *autonomous-system-number* [**alternate-as** *autonomous-system-number ...*]
5. **address-family ipv6** [**vrf** *vrf-name*] [**unicast** | **multicast** | **vpnv6**]
6. **neighbor** *ipv6-address* **peer-group** *peer-group-name*
7. **neighbor** {*ip-address* | *peer-group-name* | *ipv6-address*[*%*]}**route-map** *map-name* {**in** | **out**}
8. exit

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 2	**router bgp** *as-number* **Example:** `Device(config)# router bgp as-number`	Enters router configuration mode for the specified routing process.
Step 3	**neighbor** *peer-group-name peer-group* **Example:** `Device(config-router)# neighbor peer-group-name peer-group`	Creates a multiprotocol BGP peer group.
Step 4	**neighbor** {*ip-address* | *ipv6-address*[*%*] | *peer-group-name*}**remote-as** *autonomous-system-number* [**alternate-as** *autonomous-system-number ...*] **Example:** `Device(config-router)# neighbor {ip-address | ipv6-address[%] | peer-group-name}remote-as`	Adds the IPv6 address of the neighbor in the specified autonomous system to the IPv6 multiprotocol BGP neighbor table of the local router.

	Command or Action	Purpose
	`autonomous-system-number [alternate-as autonomous-system-number ...]`	
Step 5	**address-family ipv6** [**vrf** *vrf-name*] [**unicast** | **multicast** | **vpnv6**] **Example:** `Device(config-router)# address-family ipv6 [vrf vrf-name] [unicast | multicast | vpnv6]`	Specifies the IPv6 address family, and enters address family configuration mode. • 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 ipv6command. • The multicast keyword specifies IPv6 multicast address prefixes.
Step 6	**neighbor** *ipv6-address* **peer-group** *peer-group-name* **Example:** `Device(config-router-af)# neighbor ipv6-address peer-group peer-group-name`	Assigns the IPv6 address of a BGP neighbor to a peer group.
Step 7	**neighbor** {*ip-address* | *peer-group-name* | *ipv6-address*[*%*]}**route-map** *map-name* {**in** | **out**} **Example:** `Device(config-router-af)# neighbor {ip-address | peer-group-name | ipv6-address[%]}route-map map-name {in | out}`	Applies a route map to incoming or outgoing routes. Changes to the route map will not take effect for existing peers until the peering is reset or a soft reset is performed. Using the clear bgp ipv6 command with the soft and in keywords will perform a soft reset.
Step 8	exit **Example:** `Device(config-router-af)# exit`	Exits address family configuration mode, and returns the router to router configuration mode.

Example

This example shows how to configure EBGPv6:

```
Device(config)# router bgp 2
Device(config-router)# bgp router-id 2.2.2.2
Device(config-router)# bgp log-neighbor-changes
Device(config-router)# no bgp default ipv4-unicast
Device(config-router)# neighbor 2500::1 remote-as 1
Device(config-router)# neighbor 4000::2 remote-as 3
Device(config-router)# address-family ipv6 vrf b1
Device(config-router-af)# network 2500::/64
Device(config-router-af)# network 4000::/64
Device(config-router-af)# neighbor 2500::1 remote-as 1
Device(config-router-af)# neighbor 2500::1 activate
Device(config-router-af)# neighbor 4000::2 remote-as 3
Device(config-router-af)# neighbor 4000::2 activate
Device(config-router-af)# exit-address-family
```

Additional Information for VRF-lite

This section provides additional information about VRF-lite.

VPN Co-existence Between IPv4 and IPv6

Backward compatibility between the “older” CLI for configuring IPv4 and the “new” CLI for IPv6 exists. This means that a configuration might contain both CLI. The IPv4 CLI retains the ability to have on the same interface, an IP address defined within a VRF as well as an IPv6 address defined in the global routing table.

For example:

```
vrf definition red
 rd 100:1
 address family ipv6
 route-target both 200:1
 exit-address-family
!
vrf definition 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.

Verifying VRF-lite Configuration

This section provides steps for verifying VRF-lite configuration.

Displaying IPv4 VRF-lite Status

To display information about VRF-lite configuration and status, perform one of the following tasks:

Command	Purpose
`Device# show ip protocols vrf vrf-name`	Displays routing protocol information associated with a VRF.
`Device# 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.

Command	Purpose
Device# **show vrf definition** [**brief** | **detail** | **interfaces**] [*vrf-name*]	Displays information about the defined VRF instances.
Device# **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
```

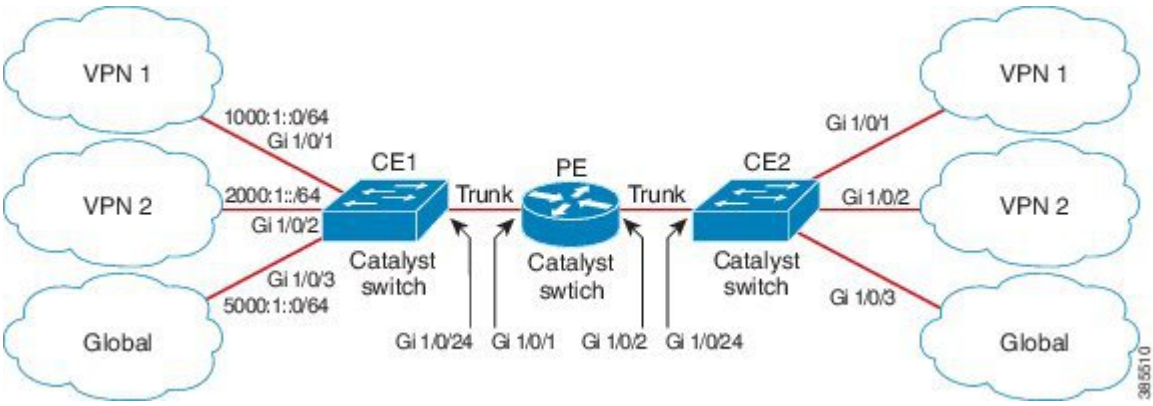
Configuration Examples for VRF-lite

This section provides configuration examples for VRF-lite.

Configuration Example for IPv6 VRF-lite

The following topology illustrates how to use OSPFv3 for CE-PE routing.

Figure 12: 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
!
```

Additional References for VRF-Lite

Related Documents

Related Topic	Document Title
For complete syntax and usage information for the commands used in this chapter.	See the IP Multicast Routing Commands section of the *Command Reference (Catalyst 9300 Series Switches)*

Standards and RFCs

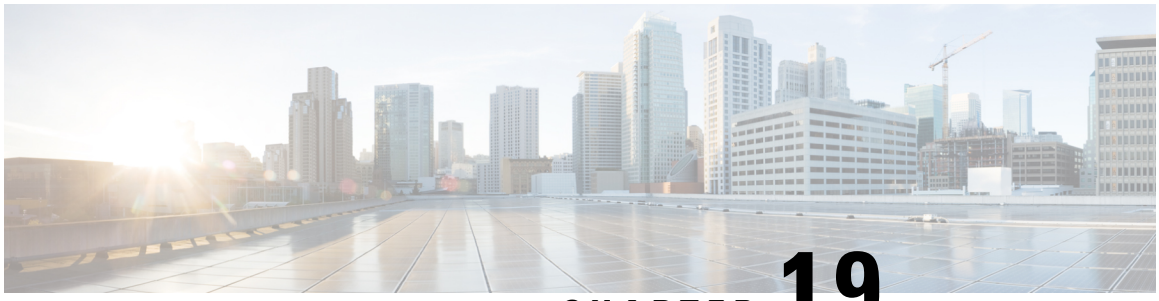
Standard/RFC	Title
RFC 6763	*DNS-Based Service Discovery*
Multicast DNS Internet-Draft	Multicast

Feature History and Information for Multicast VRF-lite

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Feature Name	Release	Feature Information
IPv6 Multicast support with VRF-Lite	Cisco IOS XE Everest 16.6.1	IPv6 VRF-Lite allows a service provider to support two or more VPNs with overlapping IP addresses using one interface.



CHAPTER 19

Configuring Multi-VRF CE

- Information About Multi-VRF CE, on page 339
- How to Configure Multi-VRF CE, on page 342
- Monitoring Multi-VRF CE, on page 357
- Configuration Example: Multi-VRF CE, on page 357
- Feature Information for Multi-VRF CE, on page 361

Information About Multi-VRF CE

Virtual Private Networks (VPNs) provide a secure way for customers to share bandwidth over an ISP backbone network. A VPN is a collection of sites sharing a common routing table. A customer site is connected to the service-provider network by one or more interfaces, and the service provider associates each interface with a VPN routing table, called a VPN routing/forwarding (VRF) table.

The switch supports multiple VPN routing/forwarding (multi-VRF) instances in customer edge (CE) devices (multi-VRF CE) when the it is running the Network Advantage license. Multi-VRF CE allows a service provider to support two or more VPNs with overlapping IP addresses.

Note The switch does not use Multiprotocol Label Switching (MPLS) to support VPNs.

Understanding Multi-VRF CE

Multi-VRF CE is a feature that allows a service provider to support two or more VPNs, where IP addresses can be overlapped among the VPNs. Multi-VRF CE uses input interfaces to distinguish routes for different VPNs and forms virtual packet-forwarding tables by associating one or more Layer 3 interfaces with each VRF. Interfaces in a VRF can be either physical, such as Ethernet ports, or logical, such as VLAN SVIs, but an interface cannot belong to more than one VRF at any time.

Note Multi-VRF CE interfaces must be Layer 3 interfaces.

Multi-VRF CE includes these devices:

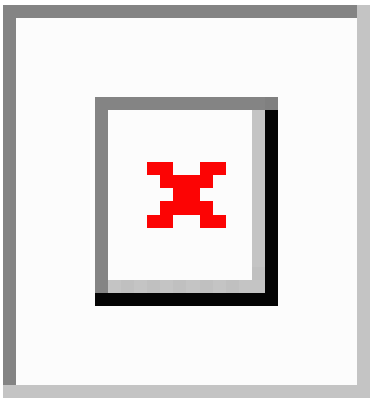
- Customer edge (CE) devices provide customers access to the service-provider network over a data link to one or more provider edge routers. The CE device advertises the site's local routes to the router and learns the remote VPN routes from it. A switch can be a CE.
- Provider edge (PE) routers exchange routing information with CE devices by using static routing or a routing protocol such as BGP, RIPv2, OSPF, or EIGRP. The PE is only required to maintain VPN routes for those VPNs to which it is directly attached, eliminating the need for the PE to maintain all of the service-provider VPN routes. Each PE router maintains a VRF for each of its directly connected sites. Multiple interfaces on a PE router can be associated with a single VRF if all of these sites participate in the same VPN. Each VPN is mapped to a specified VRF. After learning local VPN routes from CEs, a PE router exchanges VPN routing information with other PE routers by using internal BGP (IBPG).
- Provider routers or core routers are any routers in the service provider network that do not attach to CE devices.

With multi-VRF CE, multiple customers can share one CE, and only one physical link is used between the CE and the PE. The shared CE maintains separate VRF tables for each customer and switches or routes packets for each customer based on its own routing table. Multi-VRF CE extends limited PE functionality to a CE device, giving it the ability to maintain separate VRF tables to extend the privacy and security of a VPN to the branch office.

Network Topology

The figure shows a configuration using switches as multiple virtual CEs. This scenario is suited for customers who have low bandwidth requirements for their VPN service, for example, small companies. In this case, multi-VRF CE support is required in the switches. Because multi-VRF CE is a Layer 3 feature, each interface in a VRF must be a Layer 3 interface.

Figure 13: Switches Acting as Multiple Virtual CEs



When the CE switch receives a command to add a Layer 3 interface to a VRF, it sets up the appropriate mapping between the VLAN ID and the policy label (PL) in multi-VRF-CE-related data structures and adds the VLAN ID and PL to the VLAN database.

When multi-VRF CE is configured, the Layer 3 forwarding table is conceptually partitioned into two sections:

- The multi-VRF CE routing section contains the routes from different VPNs.
- The global routing section contains routes to non-VPN networks, such as the Internet.

VLAN IDs from different VRFs are mapped into different policy labels, which are used to distinguish the VRFs during processing. For each new VPN route learned, the Layer 3 setup function retrieves the policy label by using the VLAN ID of the ingress port and inserts the policy label and new route to the multi-VRF CE routing section. If the packet is received from a routed port, the port internal VLAN ID number is used; if the packet is received from an SVI, the VLAN number is used.

Packet-Forwarding Process

This is the packet-forwarding process in a multi-VRF-CE-enabled network:

- When the switch receives a packet from a VPN, the switch looks up the routing table based on the input policy label number. When a route is found, the switch forwards the packet to the PE.
- 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.
- 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. Then it performs the normal route lookup. When a route is found, it forwards the packet to the correct adjacency.
- When a CE receives a packet from an egress PE, it uses the input policy label to look up the correct VPN routing table. If a route is found, it forwards the packet within the VPN.

Network Components

To configure VRF, you create a VRF table and specify the Layer 3 interface associated with the VRF. Then configure the routing protocols in the VPN and between the CE and the PE. BGP is the preferred routing protocol used to distribute VPN routing information across the provider's backbone. The multi-VRF CE network has three major components:

- VPN route target communities—lists of 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.

VRF-Aware Services

IP services can be configured on global interfaces, and these services run within the global routing instance. IP services are enhanced to run on multiple routing instances; they are VRF-aware. Any configured VRF in the system can be specified for a VRF-aware service.

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

VRF-aware services have the following 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.

Multi-VRF CE Configuration Guidelines

Note To use multi-VRF CE, you must have the Network Advantage license enabled on your switch.

- A switch with multi-VRF CE is shared by multiple customers, and each customer has its own routing table.
- Because customers use different VRF tables, the same IP addresses can be reused. Overlapped IP addresses are allowed in different VPNs.
- Multi-VRF CE lets multiple customers share the same physical link between the PE and the CE. Trunk ports with multiple VLANs separate packets among customers. Each customer has its own VLAN.
- Multi-VRF CE does not support all MPLS-VRF functionality. It does not support label exchange, LDP adjacency, or labeled packets.
- For the PE router, there is no difference between using multi-VRF CE or using multiple CEs. In Figure 41-6, multiple virtual Layer 3 interfaces are connected to the multi-VRF CE device.
- The switch supports configuring VRF by using physical ports, VLAN SVIs, or a combination of both. The SVIs can be connected through an access port or a trunk port.
- A customer can use multiple VLANs as long as they do not overlap with those of other customers. A customer's VLANs are mapped to a specific routing table ID that is used to identify the appropriate routing tables stored on the switch.
- The switch supports one global network and up to 256 VRFs.
- Most routing protocols (BGP, OSPF, RIP, and static routing) can be used 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 makes it easy to pass attributes of the routes to the CE.
- Multi-VRF CE does not affect the packet switching rate.
- VPN multicast is not supported.
- You can enable VRF on a private VLAN, and the reverse.
- You cannot enable VRF when policy-based routing (PBR) is enabled on an interface, and the reverse.
- You cannot enable VRF when Web Cache Communication Protocol (WCCP) is enabled on an interface, and the reverse.

How to Configure Multi-VRF CE

The following sections provide configurational information about Multi-VRF CE.

Default Multi-VRF CE Configuration

Table 33: Default VRF Configuration

Feature	Default Setting
VRF	Disabled. No VRFs are defined.
Maps	No import maps, export maps, or route maps are defined.
VRF maximum routes	Fast Ethernet switches: 8000 Gigabit Ethernet switches: 12000.
Forwarding table	The default for an interface is the global routing table.

Configuring VRFs

Perform the following steps:

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device>enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device#configure terminal`	Enters global configuration mode.
Step 3	**ip routing** **Example:** `Device(config)#ip routing`	Enables IP routing.
Step 4	**ip vrf** *vrf-name* **Example:** `Device(config)#ip vrf vpn1`	Names the VRF, and enter VRF configuration mode.
Step 5	**rd** *route-distinguisher* **Example:** `Device(config-vrf)#rd 100:2`	Creates a VRF table by specifying a route distinguisher. Enter either an AS number and an arbitrary number (xxx:y) or an IP address and arbitrary number (A.B.C.D:y)

	Command or Action	Purpose
Step 6	**route-target** {**export** | **import** | **both**} *route-target-ext-community* **Example:** `Device(config-vrf)#route-target both 100:2`	Creates a list of import, export, or import and export route target communities for the specified VRF. 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). The *route-target-ext-community* should be the same as the *route-distinguisher* entered in Step 4.
Step 7	**import map** *route-map* **Example:** `Device(config-vrf)#import map importmap1`	(Optional) Associates a route map with the VRF.
Step 8	**interface** *interface-id* **Example:** `Device(config-vrf)#interface gigabitethernet 1/0/1`	Specifies the Layer 3 interface to be associated with the VRF, and enter interface configuration mode. The interface can be a routed port or SVI.
Step 9	**ip vrf forwarding** *vrf-name* **Example:** `Device(config-if)#ip vrf forwarding vpn1`	Associates the VRF with the Layer 3 interface. **Note** When **ip vrf forwarding** is enabled in the Management Interface, the access point does not join.
Step 10	**end** **Example:** `Device(config)#end`	Returns to privileged EXEC mode.
Step 11	**show ip vrf** [**brief** | **detail** | **interfaces**] [*vrf-name*] **Example:** `Device#show ip vrf interfaces vpn1`	Verifies the configuration. Displays information about the configured VRFs.
Step 12	**copy running-config startup-config** **Example:** `Device#copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

Configuring Multicast VRFs

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:**	Enables privileged EXEC mode. Enter your password if prompted.

	Command or Action	Purpose
	`Device>enable`	
Step 2	**configure terminal** **Example:** `Device#configure terminal`	Enters global configuration mode.
Step 3	**ip routing** **Example:** `Device(config)#ip routing`	Enables IP routing mode.
Step 4	**ip vrf** *vrf-name* **Example:** `Device(config)#ip vrf vpn1`	Names the VRF, and enter VRF configuration mode.
Step 5	**rd** *route-distinguisher* **Example:** `Device(config-vrf)#rd 100:2`	Creates a VRF table by specifying a route distinguisher. Enter either an AS number and an arbitrary number (xxx:y) or an IP address and an arbitrary number (A.B.C.D:y)
Step 6	**route-target** {**export** | **import** | **both**} *route-target-ext-community* **Example:** `Device(config-vrf)#route-target import 100:2`	Creates a list of import, export, or import and export route target communities for the specified VRF. 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). The *route-target-ext-community* should be the same as the *route-distinguisher* entered in Step 4.
Step 7	**import map** *route-map* **Example:** `Device(config-vrf)#import map importmap1`	(Optional) Associates a route map with the VRF.
Step 8	**ip multicast-routing vrf** *vrf-name* **distributed** **Example:** `Device(config-vrf)#ip multicast-routing vrf vpn1 distributed`	(Optional) Enables global multicast routing for VRF table.
Step 9	**interface** *interface-id* **Example:** `Device(config-vrf)#interface gigabitethernet 1/0/2`	Specifies the Layer 3 interface to be associated with the VRF, and enter interface configuration mode. The interface can be a routed port or an SVI.

	Command or Action	Purpose
Step 10	**ip vrf forwarding** *vrf-name* **Example:** `Device(config-if)#ip vrf forwarding vpn1`	Associates the VRF with the Layer 3 interface.
Step 11	**ip address** *ip-address* **mask** **Example:** `Device(config-if)#ip address 10.1.5.1 255.255.255.0`	Configures IP address for the Layer 3 interface.
Step 12	**ip pim sparse-dense mode** **Example:** `Device(config-if)#ip pim sparse-dense mode`	Enables PIM on the VRF-associated Layer 3 interface.
Step 13	**end** **Example:** `Device(config)#end`	Returns to privileged EXEC mode.
Step 14	**show ip vrf** [**brief** | **detail** | **interfaces**] [*vrf-name*] **Example:** `Device#show ip vrf detail vpn1`	Verifies the configuration. Displays information about the configured VRFs.
Step 15	**copy running-config startup-config** **Example:** `Device#copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

Configuring a VPN Routing Session

Routing within the VPN can be configured with any supported routing protocol (RIP, OSPF, EIGRP, or BGP) or with static routing. The configuration shown here is for OSPF, but the process is the same for other protocols.

Note To configure an EIGRP routing process to run within a VRF instance, you must configure an autonomous-system number by entering the **autonomous-system** *autonomous-system-number* address-family configuration mode command.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device>enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device#configure terminal`	Enters global configuration mode.
Step 3	**router ospf** *process-id* **vrf** *vrf-name* **Example:** `Device(config)#router ospf 1 vrf vpn1`	Enables OSPF routing, specifies a VPN forwarding table, and enter router configuration mode.
Step 4	**log-adjacency-changes** **Example:** `Device(config-router)#log-adjacency-changes`	(Optional) Logs changes in the adjacency state. This is the default state.
Step 5	**redistribute bgp** *autonomous-system-number* **subnets** **Example:** `Device(config-router)#redistribute bgp 10 subnets`	Sets the switch to redistribute information from the BGP network to the OSPF network.
Step 6	**network** *network-number* **area** *area-id* **Example:** `Device(config-router)#network 1 area 2`	Defines a network address and mask on which OSPF runs and the area ID for that network address.
Step 7	**end** **Example:** `Device(config-router)#end`	Returns to privileged EXEC mode.
Step 8	**show ip ospf** *process-id* **Example:** `Device#show ip ospf 1`	Verifies the configuration of the OSPF network.
Step 9	**copy running-config startup-config** **Example:** `Device#copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

Configuring BGP PE to CE Routing Sessions

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device#configure terminal`	Enters global configuration mode.
Step 2	**router bgp** *autonomous-system-number* **Example:** `Device(config)#router bgp 2`	Configures the BGP routing process with the AS number passed to other BGP routers, and enter router configuration mode.
Step 3	**network** *network-number* **mask** *network-mask* **Example:** `Device(config-router)#network 5 mask 255.255.255.0`	Specifies a network and mask to announce using BGP.
Step 4	**redistribute ospf** *process-id* **match internal** **Example:** `Device(config-router)#redistribute ospf 1 match internal`	Sets the switch to redistribute OSPF internal routes.
Step 5	**network** *network-number* **area** *area-id* **Example:** `Device(config-router)#network 5 area 2`	Defines a network address and mask on which OSPF runs and the area ID for that network address.
Step 6	**address-family ipv4 vrf** *vrf-name* **Example:** `Device(config-router)#address-family ipv4 vrf vpn1`	Defines BGP parameters for PE to CE routing sessions, and enter VRF address-family mode.
Step 7	**neighbor** *address* **remote-as** *as-number* **Example:** `Device(config-router)#neighbor 10.1.1.2 remote-as 2`	Defines a BGP session between PE and CE routers.
Step 8	**neighbor** *address* **activate** **Example:** `Device(config-router)#neighbor 10.2.1.1 activate`	Activates the advertisement of the IPv4 address family.

	Command or Action	Purpose
Step 9	**end** **Example:** `Device(config-router)#end`	Returns to privileged EXEC mode.
Step 10	**show ip bgp** [**ipv4**] [**neighbors**] **Example:** `Device#show ip bgp ipv4 neighbors`	Verifies BGP configuration.
Step 11	**copy running-config startup-config** **Example:** `Device#copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

Configuring VRF-Aware Services

These services are VRF-Aware:

- ARP
- Ping
- Simple Network Management Protocol (SNMP)
- Unicast Reverse Path Forwarding (uRPF)
- Syslog
- Traceroute
- FTP and TFTP

Configuring VRF-Aware Services for SNMP

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device>enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device#configure terminal`	Enters global configuration mode.

	Command or Action	Purpose
Step 3	**snmp-server trap authentication vrf** **Example:** `Device(config)#snmp-server trap authentication vrf`	Enables SNMP traps for packets on a VRF.
Step 4	**snmp-server engineID remote** *host* **vrf** *vpn-instance engine-id string* **Example:** `Device(config)#snmp-server engineID remote 172.16.20.3 vrf vpn1 80000009030000B064EFE100`	Configures a name for the remote SNMP engine on a switch.
Step 5	**snmp-server host** *host* **vrf** *vpn-instance* **traps** *community* **Example:** `Device(config)#snmp-server host 172.16.20.3 vrf vpn1 traps comaccess`	Specifies the recipient of an SNMP trap operation and specifies the VRF table to be used for sending SNMP traps.
Step 6	**snmp-server host** *host* **vrf** *vpn-instance* **informs** *community* **Example:** `Device(config)#snmp-server host 172.16.20.3 vrf vpn1 informs comaccess`	Specifies the recipient of an SNMP inform operation and specifies the VRF table to be used for sending SNMP informs.
Step 7	**snmp-server user** *user group* **remote** *host* **vrf** *vpn-instance security model* **Example:** `Device(config)#snmp-server user abcd remote 172.16.20.3 vrf vpn1 priv v2c 3des secure3des`	Adds a user to an SNMP group for a remote host on a VRF for SNMP access.
Step 8	**end** **Example:** `Device(config-if)#end`	Returns to privileged EXEC mode.

Configuring VRF-Aware Servcies for NTP

Configuring VRF-aware services for NTP comprises configuring the NTP servers and the NTP client interfaces connected to the NTP servers.

Before you begin

Ensure connectivity between the NTP client and servers. Configure a valid IP address and subnet on the client interfaces that are connected to the NTP servers.

Configuring VRF-Aware Servcies for NTP on NTP Client

Perform the following steps on the client interface that is connected to the NTP server.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device>enable`	Enables privileged EXEC mode. • Enter your password, if prompted.
Step 2	**configure terminal** **Example:** `Device#configure terminal`	Enters global configuration mode.
Step 3	**interface** *interface-id* **Example:** `Device(config)#interface gigabitethernet 1/0/1`	Specifies the Layer 3 interface to be associated with the VRF, and enters the interface configuration mode.
Step 4	**vrf forwarding** *vrf-name* **Example:** `Device(config-if)#vrf forwarding A`	Associates the VRF with the Layer 3 interface.
Step 5	**ip address** *ip-address subnet-mask* **Example:** `Device(config-if)#ip address 1.1.1.1 255.255.255.0`	Enter the IP address for the interface.
Step 6	**no shutdown** **Example:** `Device(config-if)#no shutdown`	Enables the interface.
Step 7	**exit** **Example:** `Device(config-if)exit`	Exits the interface configuration mode.
Step 8	**ntp authentication-key** *number* **md5** *md5-number* **Example:** `Device(config)#ntp authentication-key 1 md5 cisco123`	Defines the authentication keys. The device does not synchronize to a time source unless the source has one of these authentication keys and the key number is specified by the **ntp trusted-key number** command. **Note** The authentication key *number* and the MD5 *passowrd* must be the same on both the client and server.

	Command or Action	Purpose
Step 9	**ntp authenticate** **Example:** `Device(config)#ntp authenticate`	Enables the NTP authentication feature. NTP authentication is disabled by default.
Step 10	**ntp trusted-key** *key-number* **Example:** `Device(config)#ntp trusted-key 1`	Specifies one or more keys that an NTP server must provide in its NTP packets in order for the NTP client to synchronize to it. The range for trusted keys is from 1 to 65535. This command provides protection against accidentally synchronizing the NTP client to an NTP server that is not trusted.
Step 11	**ntp server vrf** *vrf-name* **Example:** `Device(config)#ntp server vrf A 1.1.1.2 key 1`	Configures NTP Server in the specified VRF.

Configuring VRF-Aware Services for NTP on the NTP Server

Perform the following steps on the NTP server.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device>enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device#configure terminal`	Enters global configuration mode.
Step 3	**ntp authentication-key** *number* **md5** *passowrd* **Example:** `Device(config)#ntp authentication-key 1 md5 cisco123`	Defines the authentication keys. The device does not synchronize to a time source unless the source has one of these authentication keys and the key number is specified by the **ntp trusted-key number** command. **Note** The authentication key *number* and the MD5 *passowrd* must be the same on both the client and server.
Step 4	**ntp authenticate** **Example:** `Device(config)#ntp authenticate`	Enables the NTP authentication feature. NTP authentication is disabled by default.

	Command or Action	Purpose
Step 5	**ntp trusted-key** *key-number* **Example:** `Device(config)#ntp trusted-key 1`	Specifies one or more keys that an NTP server must provide in its NTP packets in order for the NTP client to synchronize to it. The range for trusted keys is from 1 to 65535. This command provides protection against accidentally synchronizing the NTP client to an NTP server that is not trusted.
Step 6	**interface** *interface-id* **Example:** `Device(config)#interface gigabitethernet 1/0/3`	Specifies the Layer 3 interface to be associated with the VRF, and enters the interface configuration mode.
Step 7	**vrf forwarding** *vrf-name* **Example:** `Device(config-if)#vrf forwarding A`	Associates the VRF with the Layer 3 interface.
Step 8	**ip address** *ip-address subnet-mask* **Example:** `Device(config-if)#ip address 1.1.1.2 255.255.255.0`	Enter the IP address for the interface.
Step 9	**exit** **Example:** `Device(config-if)exit`	Exits the interface configuration mode.

Configuring VRF-Aware Services for uRPF

uRPF can be configured on an interface assigned to a VRF, and source lookup is done in the VRF table.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device>enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device#configure terminal`	Enters global configuration mode.
Step 3	**interface** *interface-id* **Example:** `Device(config)#interface gigabitethernet 1/0/1`	Enters interface configuration mode, and specifies the Layer 3 interface to configure.

	Command or Action	Purpose
Step 4	**no switchport** **Example:** `Device(config-if)#no switchport`	Removes the interface from Layer 2 configuration mode if it is a physical interface.
Step 5	**ip vrf forwarding** *vrf-name* **Example:** `Device(config-if)#ip vrf forwarding vpn2`	Configures VRF on the interface.
Step 6	**ip address** *ip-address* **Example:** `Device(config-if)#ip address 10.1.5.1`	Enters the IP address for the interface.
Step 7	**ip verify unicast reverse-path** **Example:** `Device(config-if)#ip verify unicast reverse-path`	Enables uRPF on the interface.
Step 8	**end** **Example:** `Device(config-if)#end`	Returns to privileged EXEC mode.

Configuring VRF-Aware RADIUS

To configure VRF-Aware RADIUS, you must first enable AAA on a RADIUS server. The switch supports the **ip vrf forwarding** *vrf-name* server-group configuration and the **ip radius source-interface** global configuration commands, as described in the *Per VRF AAA Feature Guide*.

Configuring VRF-Aware Services for Syslog

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device>enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**configure terminal** **Example:**	Enters global configuration mode.

	Command or Action	Purpose
	`Device#configure terminal`	
Step 3	**logging on** **Example:** `Device(config)#logging on`	Enables or temporarily disables logging of storage router event message.
Step 4	**logging host** *ip-address* **vrf** *vrf-name* **Example:** `Device(config)#logging host 10.10.1.0 vrf vpn1`	Specifies the host address of the syslog server where logging messages are to be sent.
Step 5	**logging buffered** *logging buffered size* **debugging** **Example:** `Device(config)#logging buffered critical 6000 debugging`	Logs messages to an internal buffer.
Step 6	**logging trap debugging** **Example:** `Device(config)#logging trap debugging`	Limits the logging messages sent to the syslog server.
Step 7	**logging facility** *facility* **Example:** `Device(config)#logging facility user`	Sends system logging messages to a logging facility.
Step 8	**end** **Example:** `Device(config-if)#end`	Returns to privileged EXEC mode.

Configuring VRF-Aware Services for Traceroute

Procedure

	Command or Action	Purpose
Step 1	**traceroute vrf** *vrf-name ipaddress* **Example:** `Device(config)#traceroute vrf vpn2 10.10.1.1`	Specifies the name of a VPN VRF in which to find the destination address.

Configuring VRF-Aware Services for FTP and TFTP

So that FTP and TFTP are VRF-aware, you must configure some FTP/TFTP CLIs. For example, if you want to use a VRF table that is attached to an interface, say E1/0, you need to configure the **ip tftp source-interface E1/0** or the **ip ftp source-interface E1/0** command to inform TFTP or FTP server to use a specific routing table. In this example, the VRF table is used to look up the destination IP address. These changes are backward-compatible and do not affect existing behavior. That is, you can use the source-interface CLI to send packets out a particular interface even if no VRF is configured on that interface.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device>enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device#configure terminal`	Enters global configuration mode.
Step 3	**ip ftp source-interface** *interface-type interface-number* **Example:** `Device(config)#ip ftp source-interface gigabitethernet 1/0/2`	Specifies the source IP address for FTP connections.
Step 4	**end** **Example:** `Device(config)#end`	Returns to privileged EXEC mode.
Step 5	**configure terminal** **Example:** `Device#configure terminal`	Enters global configuration mode.
Step 6	**ip tftp source-interface** *interface-type interface-number* **Example:** `Device(config)#ip tftp source-interface gigabitethernet 1/0/2`	Specifies the source IP address for TFTP connections.
Step 7	**end** **Example:**	Returns to privileged EXEC mode.

	Command or Action	Purpose
	`Device(config)#end`	

Monitoring VRF-Aware Services for ARP

Procedure

	Command or Action	Purpose
Step 1	**show ip arp vrf** *vrf-name* **Example:** `Device#show ip arp vrf vpn1`	Displays the ARP table in the specified VRF.

Configuring VRF-Aware Services for Ping

Procedure

	Command or Action	Purpose
Step 1	**ping vrf***vrf-name***ip-host** **Example:** `Device#ping vrf vpn1 ip-host`	Displays the ARP table in the specified VRF.

Monitoring Multi-VRF CE

Table 34: Commands for Displaying Multi-VRF CE Information

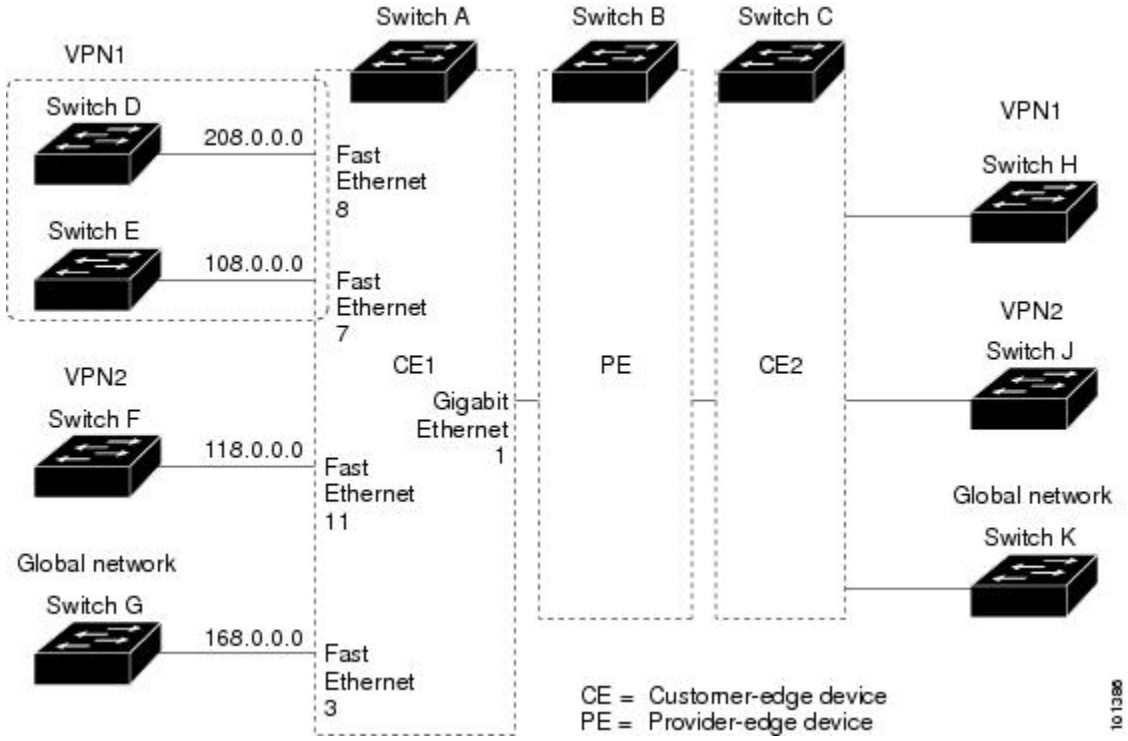
Command	Purpose
show ip protocols vrf *vrf-name*	Displays routing protocol information as a VRF.
show ip route vrf *vrf-name* [**connected**] [*protocol* [*as-number*]] [**list**] [**mobile**] [**odr**] [**profile**] [**static**] [**summary**] [**supernets-only**]	Displays IP routing table information as a VRF.
show ip vrf [**brief** | **detail** | **interfaces**] [*vrf-name*]	Displays information about the defined V

Configuration Example: Multi-VRF CE

OSPF is the protocol used in VPN1, VPN2, and the global network. BGP is used in the CE to PE connections. The examples following the illustration show how to configure a switch as CE Switch A, and the VRF

configuration for customer switches D and F. Commands for configuring CE Switch C and the other customer switches are not included but would be similar.

Figure 14: Establishing a Multi-VRF CE Configuration Example



On Switch A, enable routing and configure VRF.

```
Device#configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
Device(config)#ip routing
Device(config)#ip vrf v11
Device(config-vrf)#rd 800:1
Device(config-vrf)#route-target export 800:1
Device(config-vrf)#route-target import 800:1
Device(config-vrf)#exit
Device(config)#ip vrf v12
Device(config-vrf)#rd 800:2
Device(config-vrf)#route-target export 800:2
Device(config-vrf)#route-target import 800:2
Device(config-vrf)#exit
```

Configure the loopback and physical interfaces on Switch A. Gigabit Ethernet port 1 is a trunk connection to the PE. Gigabit Ethernet ports 8 and 11 connect to VPNs:

```
Device(config)#interface loopback1
Device(config-if)#ip vrf forwarding v11
Device(config-if)#ip address 8.8.1.8 255.255.255.0
Device(config-if)#exit

Device(config)#interface loopback2
Device(config-if)#ip vrf forwarding v12
Device(config-if)#ip address 8.8.2.8 255.255.255.0
Device(config-if)#exit
```

```
Device(config)#interface gigabitethernet1/0/5
Device(config-if)#switchport trunk encapsulation dot1q
Device(config-if)#switchport mode trunk
Device(config-if)#no ip address
Device(config-if)#exit
Device(config)#interface gigabitethernet1/0/8
Device(config-if)#switchport access vlan 208
Device(config-if)#no ip address
Device(config-if)#exit
Device(config)#interface gigabitethernet1/0/11
Device(config-if)#switchport trunk encapsulation dot1q
Device(config-if)#switchport mode trunk
Device(config-if)#no ip address
Device(config-if)#exit
```

Configure the VLANs used on Switch A. VLAN 10 is used by VRF 11 between the CE and the PE. VLAN 20 is used by VRF 12 between the CE and the PE. VLANs 118 and 208 are used for the VPNs that include Switch F and Switch D, respectively:

```
Device(config)#interface vlan10
Device(config-if)#ip vrf forwarding v11
Device(config-if)#ip address 38.0.0.8 255.255.255.0
Device(config-if)#exit
Device(config)#interface vlan20
Device(config-if)#ip vrf forwarding v12
Device(config-if)#ip address 83.0.0.8 255.255.255.0
Device(config-if)#exit
Device(config)#interface vlan118
Device(config-if)#ip vrf forwarding v12
Device(config-if)#ip address 118.0.0.8 255.255.255.0
Device(config-if)#exit
Device(config)#interface vlan208
Device(config-if)#ip vrf forwarding v11
Device(config-if)#ip address 208.0.0.8 255.255.255.0
Device(config-if)#exit
```

Configure OSPF routing in VPN1 and VPN2.

```
Device(config)#router ospf 1 vrf vl1
Device(config-router)#redistribute bgp 800 subnets
Device(config-router)#network 208.0.0.0 0.0.0.255 area 0
Device(config-router)#exit
Device(config)#router ospf 2 vrf vl2
Device(config-router)#redistribute bgp 800 subnets
Device(config-router)#network 118.0.0.0 0.0.0.255 area 0
Device(config-router)#exit
```

Configure BGP for CE to PE routing.

```
Device(config)#router bgp 800
Device(config-router)#address-family ipv4 vrf vl2
Device(config-router-af)#redistribute ospf 2 match internal
Device(config-router-af)#neighbor 83.0.0.3 remote-as 100
Device(config-router-af)#neighbor 83.0.0.3 activate
Device(config-router-af)#network 8.8.2.0 mask 255.255.255.0
Device(config-router-af)#exit
Device(config-router)#address-family ipv4 vrf vl1
Device(config-router-af)#redistribute ospf 1 match internal
Device(config-router-af)#neighbor 38.0.0.3 remote-as 100
Device(config-router-af)#neighbor 38.0.0.3 activate
```

```
Device(config-router-af)#network 8.8.1.0 mask 255.255.255.0
Device(config-router-af)#end
```

Switch D belongs to VPN 1. Configure the connection to Switch A by using these commands.

```
Device#configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
Device(config)#ip routing
Device(config)#interface gigabitethernet1/0/2
Device(config-if)#no switchport
Device(config-if)#ip address 208.0.0.20 255.255.255.0
Device(config-if)#exit

Device(config)#router ospf 101
Device(config-router)#network 208.0.0.0 0.0.0.255 area 0
Device(config-router)#end
```

Switch F belongs to VPN 2. Configure the connection to Switch A by using these commands.

```
Device#configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
Device(config)#ip routing
Device(config)#interface gigabitethernet1/0/1
Device(config-if)#switchport trunk encapsulation dot1q
Device(config-if)#switchport mode trunk
Device(config-if)#no ip address
Device(config-if)#exit

Device(config)#interface vlan118
Device(config-if)#ip address 118.0.0.11 255.255.255.0
Device(config-if)#exit

Device(config)#router ospf 101
Device(config-router)#network 118.0.0.0 0.0.0.255 area 0
Device(config-router)#end
```

When used on switch B (the PE router), these commands configure only the connections to the CE device, Switch A.

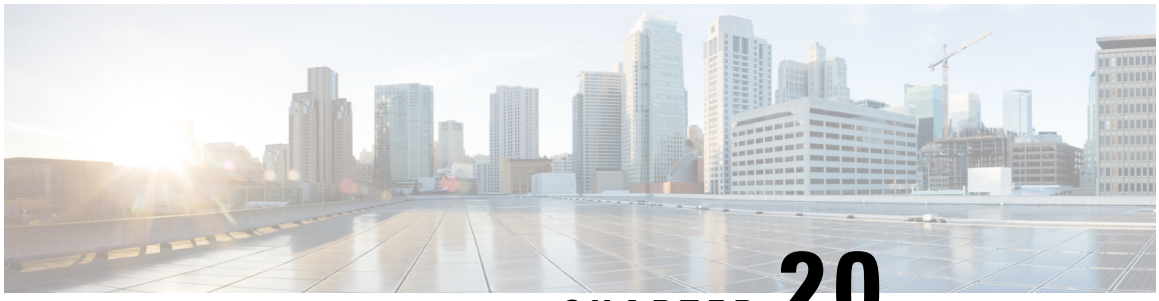
```
Device#configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
Device(config)#ip vrf v1
Device(config-vrf)#rd 100:1
Device(config-vrf)#route-target export 100:1
Device(config-vrf)#route-target import 100:1
Device(config-vrf)#exit

Device(config)#ip vrf v2
Device(config-vrf)#rd 100:2
Device(config-vrf)#route-target export 100:2
Device(config-vrf)#route-target import 100:2
Device(config-vrf)#exit
Device(config)#ip cef
Device(config)#interface Loopback1
Device(config-if)#ip vrf forwarding v1
Device(config-if)#ip address 3.3.1.3 255.255.255.0
Device(config-if)#exit

Device(config)#interface Loopback2
Device(config-if)#ip vrf forwarding v2
Device(config-if)#ip address 3.3.2.3 255.255.255.0
Device(config-if)#exit
```

```
Device(config)#interface gigabitethernet1/1/0.10
Device(config-if)#encapsulation dot1q 10
Device(config-if)#ip vrf forwarding v1
Device(config-if)#ip address 38.0.0.3 255.255.255.0
Device(config-if)#exit

Device(config)#interface gigabitethernet1/1/0.20
Device(config-if)#encapsulation dot1q 20
Device(config-if)#ip vrf forwarding v2
Device(config-if)#ip address 83.0.0.3 255.255.255.0
Device(config-if)#exit

Device(config)#router bgp 100
Device(config-router)#address-family ipv4 vrf v2
Device(config-router-af)#neighbor 83.0.0.8 remote-as 800
Device(config-router-af)#neighbor 83.0.0.8 activate
Device(config-router-af)#network 3.3.2.0 mask 255.255.255.0
Device(config-router-af)#exit
Device(config-router)#address-family ipv4 vrf vl
Device(config-router-af)#neighbor 38.0.0.8 remote-as 800
Device(config-router-af)#neighbor 38.0.0.8 activate
Device(config-router-af)#network 3.3.1.0 mask 255.255.255.0
Device(config-router-af)#end
```

Feature Information for Multi-VRF CE

Table 35: Feature Information for Multi-VRF CE

Feature Name	Release	Feature Information
Multi-VRF CE	Cisco IOS XE Everest 16.5.1a	This feature was introduced



CHAPTER 20

Configuring Unicast Reverse Path Forwarding

- Configuring Unicast Reverse Path Forwarding, on page 363
- Configuring IPv6 Unicast Reverse Path Forwarding, on page 363

Configuring Unicast Reverse Path Forwarding

The unicast reverse path forwarding (unicast RPF) feature 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. For example, a number of common types of denial-of-service (DoS) attacks, including Smurf and Tribal Flood Network (TFN), can take advantage of forged or rapidly changing source IP addresses to allow attackers to thwart efforts to locate or filter the attacks. For Internet service providers (ISPs) that provide public access, Unicast RPF deflects such attacks by forwarding only packets that have source addresses that are valid and consistent with the IP routing table. This action protects the network of the ISP, its customer, and the rest of the Internet.

Note

- Unicast RPF is supported in Network Essentials.
- 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*.

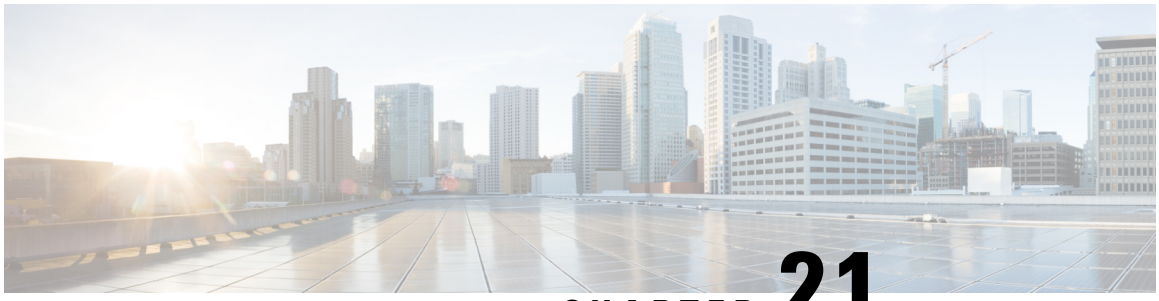
Configuring IPv6 Unicast Reverse Path Forwarding

The unicast Reverse Path Forwarding (unicast RPF) feature 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. For example, a number of common types of denial-of-service (DoS) attacks, including Smurf and Tribal Flood Network (TFN), can take advantage of forged or rapidly changing source IP addresses to allow attackers to thwart efforts to locate or filter the attacks. For Internet service providers (ISPs) that provide public access, Unicast RPF deflects such attacks by forwarding only packets that have source addresses that are valid and consistent with the IP routing table. This action protects the network of the ISP, its customer, and the rest of the Internet.

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*.



CHAPTER 21

Configuring Generic Routing Encapsulation(GRE) Tunnel IP Source and Destination VRF Membership

- Restrictions for GRE Tunnel IP Source and Destination VRF Membership, on page 365
- Information About GRE Tunnel IP Source and Destination VRF Membership, on page 366
- How to Configure GRE Tunnel IP Source and Destination VRF Membership, on page 366
- Configuration Example for GRE Tunnel IP Source and Destination VRF Membership, on page 367
- Additional References, on page 368
- Feature History for Generic Routing Encapsulation Tunnel IP Source and Destination VRF Membership, on page 368

Restrictions for GRE Tunnel IP Source and Destination VRF Membership

- Both ends of the tunnel must reside within the same VRF.
- The VRF associated with the tunnel vrf command is the same as the VRF associated with the physical interface over which the tunnel sends packets (outer IP packet routing).
- The VRF associated with the tunnel by using the ip vrf forwarding command is the VRF that the packets are to be forwarded in as the packets exit the tunnel (inner IP packet routing).
- The feature does not support the fragmentation of multicast packets passing through a multicast tunnel.
- The feature does not support the ISIS (Intermediate System to intermediate system) protocol.
- IPv6 ICMP response packets are not supported over IPv4 GRE tunnels.
- Keepalive is not supported on VRF aware GRE tunnels.

Information About GRE Tunnel IP Source and Destination VRF Membership

This feature allows you to configure the source and destination of a tunnel to belong to any Virtual Private Network (VPN) routing and forwarding (VRF) table. A VRF table stores routing data for each VPN. The VRF table defines the VPN membership of a customer site attached to the network access server (NAS). Each VRF table comprises an IP routing table, a derived Cisco Express Forwarding (CEF) table, and guidelines and routing protocol parameters that control the information that is included in the routing table.

Previously, GRE IP tunnels required the IP tunnel destination to be in the global routing table. The implementation of this feature allows you to configure a tunnel source and destination to belong to any VRF. As with existing GRE tunnels, the tunnel becomes disabled if no route to the tunnel destination is defined.

How to Configure GRE Tunnel IP Source and Destination VRF Membership

Follow these steps to configure GRE Tunnel IP Source and Destination VRF Membership:

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface tunnel***number*
4. **ip vrf forwarding***vrf-name*
5. **ip address***ip-address subnet-mask*
6. **tunnel source** { *ip-address* | *type number* }
7. **tunnel destination** { *hostname* | *ip-address* }
8. **tunnel vrf***vrf-name*

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device>enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device#configure terminal`	Enters global configuration mode.
Step 3	**interface tunnel***number* **Example:** `Device(config)#interface tunnel 0`	Enters interface configuration mode for the specified interface. • *number* is the number associated with the tunnel interface.

	Command or Action	Purpose
Step 4	**ip vrf forwarding***vrf-name* **Example:** `Device(config-if)#ip vrf forwarding green`	Associates a virtual private network (VPN) routing and forwarding (VRF) instance with an interface or subinterface. • *vrf-name* is the name assigned to a VRF.
Step 5	**ip address***ip-address subnet-mask* **Example:** `Device(config-if)#ip address 10.7.7.7 255.255.255.255`	Specifies the interface IP address and subnet mask. • *ip-address* specifies the IP address of the interface. • *subnet-mask* specifies the subnet mask of the interface.
Step 6	**tunnel source** {*ip-address* | *type number*} **Example:** `Device(config-if)#tunnel source loop 0`	Specifies the source of the tunnel interface. • *ip-address* specifies the IP address to use as the source address for packets in the tunnel. • *type* specifies the interface type (for example, serial). • *number* specifies the port, connector, or interface card number. The numbers are assigned at the factory at the time of installation or when added to a system, and can be displayed using the show interfaces command.
Step 7	**tunnel destination** {*hostname* | *ip-address*} **Example:** `Device(config-if)#tunnel destination 10.5.5.5`	Defines the tunnel destination. • *hostname* specifies the name of the host destination. • *ip-address* specifies the IP address of the host destination.
Step 8	**tunnel vrf***vrf-name* **Example:** `Device(config-if)#tunnel vrf finance1`	Associates a VPN routing and forwarding (VRF) instance with a specific tunnel destination. • *vrf-name* is the name assigned to a VRF.

Configuration Example for GRE Tunnel IP Source and Destination VRF Membership

In this example, packets received on interface e0 using VRF green are forwarded out of the tunnel through interface e1 using VRF blue.

```
ip vrf blue rd 1:1

ip vrf green rd 1:2

interface loop0
ip vrf forwarding blue
ip address 10.7.7.7 255.255.255.255

interface tunnel0
ip vrf forwarding green
```

```
ip address 10.3.3.3 255.255.255.0 tunnel source loop 0
tunnel destination 10.5.5.5 tunnel vrf blue

interface ethernet0
ip vrf forwarding green
ip address 10.1.1.1 255.255.255.0

interface ethernet1
ip vrf forwarding blue
ip address 10.2.2.2 255.255.255.0

ip route vrf blue 10.5.5.5 255.255.255.0 ethernet 1
```

Additional References

Table 36: Related Documents

Related Topic	Document Title
VRF tables	"Configuring Multiprotocol Label Switching" chapter of the Cisco IOS Switching Services Configuration Guide, Release 12.2
Tunnels	Cisco IOS Interface Configuration Guide, Release 12.2

Feature History for Generic Routing Encapsulation Tunnel IP Source and Destination VRF Membership

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Table 37: Feature History for Generic Routing Encapsulation Tunnel IP Source and Destination VRF Membership

Feature Name	Releases	Feature Information
Generic Routing Encapsulation Tunnel IP Source and Destination VRF Membership	Cisco IOS 16.6.1	The Generic Routing Encapsulation Tunnel IP Source and Destination VRF Membership feature allows you to configure the source and destination of a tunnel to belong to any virtual private network (VPN) routing and forwarding (VRF) table.