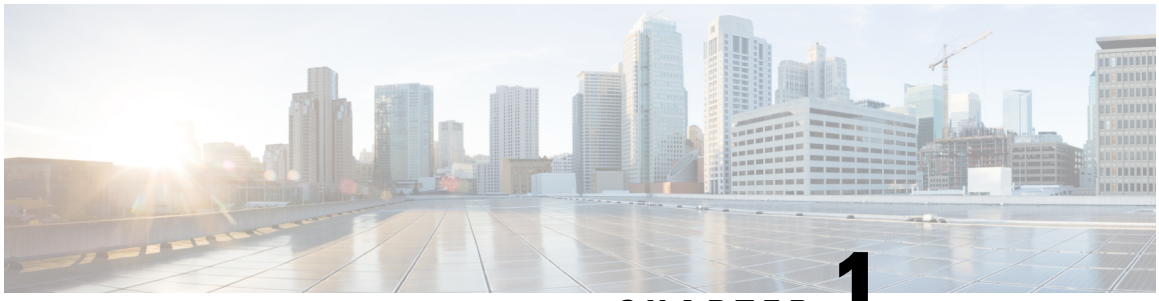
IP Routing Configuration Guide, Cisco IOS XE Dublin 17.12.x (Catalyst 9200 Switches)

First Published: 2023-07-28

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



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 4
- Feature History for Configuring Bidirectional Forwarding Detection , on page 19

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.
- When you use YANG operational models to delete individual BFD interval values, the whole BFD interval configuration gets deleted.

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.

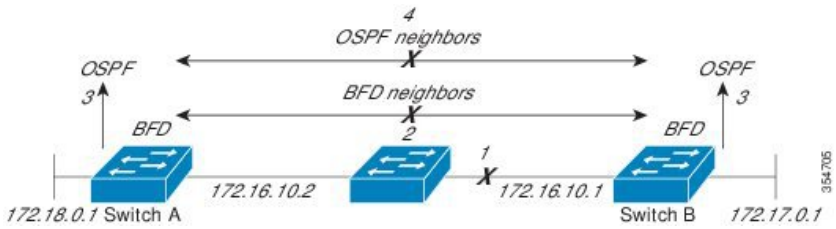
Neighbor Relationships

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

Figure 1: BFD Process on a Network Configured with OSPF

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

Figure 2: BFD Process During a Network Failure



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

OSPF registers with BFD when:

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

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

BFD Detection of Failures

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

The maximum number of BFD sessions that can be created is 128.

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.

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 IS-IS and OSPF deployments, is the use of modified failure detection mechanisms for IS-IS and OSPF routing protocols. If you use fast hellos for either IS-IS or OSPF, these Interior Gateway Protocol (IGP) protocols 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 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 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 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.
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.

	Command or Action	Purpose
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 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.

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 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.
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:**	Exits interface configuration mode and returns the device to privileged EXEC mode.

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

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 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.

	Command or Action	Purpose
		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.
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.
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.

	Command or Action	Purpose
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:** `Device#show ip ospf`	(Optional) Displays information that can help verify if BFD for OSPF has been enabled.

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.

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 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.

	Command or Action	Purpose
Step 5	**end** **Example:** `Device(config-if)#end`	Exits interface configuration mode and returns the device to privileged EXEC mode.
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.

	Command or Action	Purpose
Step 2	**configure terminal** **Example:** `Device#configure terminal`	Enters global configuration mode.
Step 3	**ip cef** [**distributed**] **Example:** `Device(config)#ip cef`	Enables Cisco Express Forwarding or distributed Cisco Express Forwarding.
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:**	Exits global configuration mode.

	Command or Action	Purpose
	`Device(config)#exit`	
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*	Enables BFD on the interface.

	Command or Action	Purpose
	Example: `Device(config-if)#bfd interval 500 min_rx 500 multiplier 5`	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.
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.

	Command or Action	Purpose
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 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.
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**	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	**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:** `Device(bfd-config)#end`	Exits BFD configuration mode and returns the device to privileged EXEC mode.

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.

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	**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.

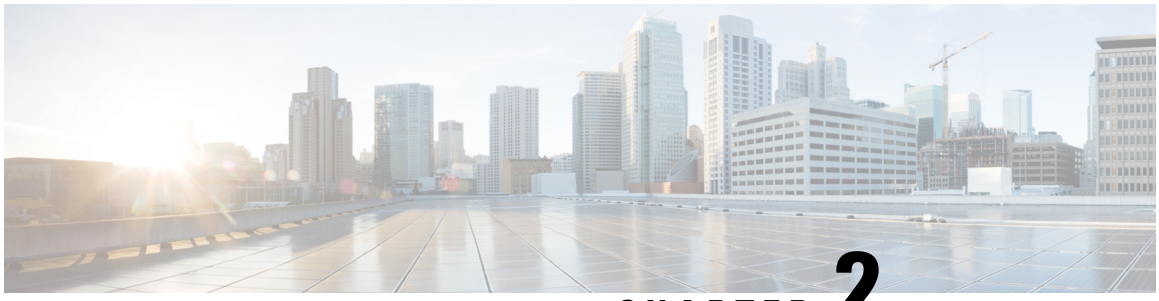
Feature History for Configuring Bidirectional Forwarding Detection

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 Fuji 16.9.2	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.
Cisco IOS XE Cupertino 17.9.1	Bidirectional Forwarding Detection	This feature was implemented on C9200CX-12P-2X2G, C9200CX-8P-2X2G, and C9200CX-12T-2X2G models of the Cisco Catalyst 9200CX Series Switches, which were introduced in this release.

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-EIGRP Support

- Prerequisites for BFD-EIGRP Support, on page 21
- Information About BFD-EIGRP Support, on page 21
- How to Configure BFD - EIGRP Support, on page 21
- Configuration Example for BFD in an EIGRP Network with Echo Mode Enabled by Default, on page 23
- Feature History for Configuring BFD-EIGRP Support, on page 28

Prerequisites for BFD-EIGRP Support

- Enhanced Interior Gateway Routing Protocol (EIGRP) must be running on all participating routers.
- The baseline parameters for Bidirectional Forwarding Detection (BFD) sessions on the interfaces over which you want to run BFD sessions to BFD neighbors must be configured using the **bfd** command.

Information About BFD-EIGRP Support

The BFD-EIGRP Support feature configures Bidirectional Forwarding Detection (BFD) feature for Enhanced Interior Gateway Routing Protocol (EIGRP) so that EIGRP registers with the BFD sessions on the routing interfaces, and receives forwarding path detection failure messages from BFD.

Use **bfd interval** *milliseconds* **min_rx** *milliseconds* **multiplier** *interval-multiplier* command to enable BFD on any interface. Use the **bfd all-interfaces** command in router configuration mode to enable BFD for all of the interfaces where EIGRP routing is enabled. Use the **bfd interface** *type number* command in router configuration mode to enable BFD for a subset of the interfaces where EIGRP routing is enabled.

How to Configure BFD - EIGRP Support

To configure BFD-EIGRP support, 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 eigrp** *as-number* **Example:** `Device(config)#router eigrp 123`	Configures the EIGRP routing process and enters router configuration mode.

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 FastEthernet 6/0`
		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 device 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 has been enabled.

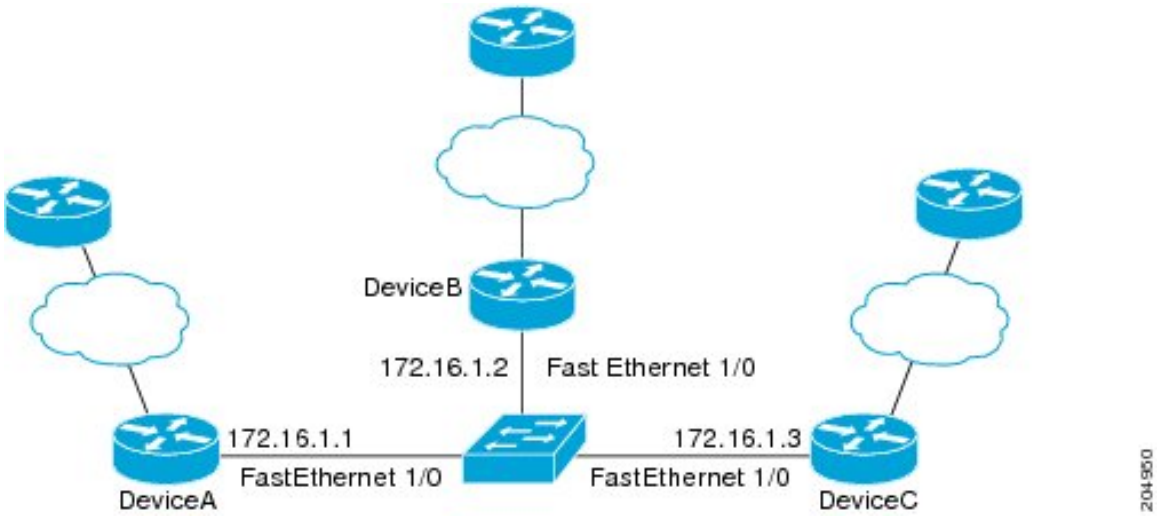
Configuration Example for BFD in an EIGRP Network with Echo Mode Enabled by Default

In the following example, the EIGRP network contains DeviceA, DeviceB, and DeviceC. Fast Ethernet interface 1/0 on DeviceA is connected to the same network as Fast Ethernet interface 1/0 on DeviceB. Fast Ethernet interface 1/0 on DeviceB is connected to the same network as Fast Ethernet interface 1/0 on DeviceC.

DeviceA and DeviceB are running BFD Version 1, which supports echo mode, and DeviceC is running BFD Version 0, which does not support echo mode. The BFD sessions between DeviceC and its BFD neighbors are said to be running echo mode with asymmetry because echo mode will run on the forwarding path for DeviceA and DeviceB, and their echo packets will return along the same path for BFD sessions and failure detections, while their BFD neighbor DeviceC runs BFD Version 0 and uses BFD controls packets for BFD sessions and failure detections.

The figure below shows a large EIGRP network with several devices, three of which are BFD neighbors that are running EIGRP as their routing protocol.

Figure 3: BFD Process on a Network Configured with EIGRP



The example, starting in global configuration mode, shows the configuration of BFD.

Configuration for DeviceA

```
interface Fast Ethernet0/0
no shutdown
ip address 10.4.9.14 255.255.255.0
duplex auto
speed auto
!
interface Fast Ethernet1/0
ip address 172.16.1.1 255.255.255.0
bfd interval 50 min_rx 50 multiplier 3
no shutdown
duplex auto
speed auto
!
```

```
router eigrp 11
network 172.16.0.0
bfd all-interfaces
auto-summary
!
ip default-gateway 10.4.9.1
ip default-network 0.0.0.0
ip route 0.0.0.0 0.0.0.0 10.4.9.1
ip route 172.16.1.129 255.255.255.255 10.4.9.1
!
no ip http server
!
logging alarm informational
!
control-plane
!
line con 0
exec-timeout 30 0
stopbits 1
line aux 0
stopbits 1
line vty 0 4
login
!
!
end
```

Configuration for DeviceB

```
!
interface Fast Ethernet0/0
no shutdown
ip address 10.4.9.34 255.255.255.0
duplex auto
speed auto
!
interface Fast Ethernet1/0
ip address 172.16.1.2 255.255.255.0
bfd interval 50 min_rx 50 multiplier 3
no shtdown
duplex auto
speed auto
!
router eigrp 11
network 172.16.0.0
bfd all-interfaces
auto-summary
!
ip default-gateway 10.4.9.1
ip default-network 0.0.0.0
ip route 0.0.0.0 0.0.0.0 10.4.9.1
ip route 172.16.1.129 255.255.255.255 10.4.9.1
!
no ip http server
!
logging alarm informational
!
control-plane
!
line con 0
exec-timeout 30 0
stopbits 1
line aux 0
```

```
stopbits 1
line vty 0 4
login
!
!
end
```

Configuration for DeviceC

```
!
!
interface Fast Ethernet0/0
no shutdown
ip address 10.4.9.34 255.255.255.0
duplex auto
speed auto
!
interface Fast Ethernet1/0
ip address 172.16.1.2 255.255.255.0
bfd interval 50 min_rx 50 multiplier 3
no shutdown
duplex auto
speed auto
!
router eigrp 11
network 172.16.0.0
bfd all-interfaces
auto-summary
!
ip default-gateway 10.4.9.1
ip default-network 0.0.0.0
ip route 0.0.0.0 0.0.0.0 10.4.9.1
ip route 172.16.1.129 255.255.255.255 10.4.9.1
!
no ip http server
!
logging alarm informational
!
control-plane
!
line con 0
exec-timeout 30 0
stopbits 1
line aux 0
stopbits 1
line vty 0 4
login
!
!
end
```

The output from the **show bfd neighbors details** command from DeviceA verifies that BFD sessions are created among all three devices and that EIGRP is registered for BFD support. The first group of output shows that DeviceC with the IP address 172.16.1.3 runs BFD Version 0 and therefore does not use the echo mode. The second group of output shows that DeviceB with the IP address 172.16.1.2 runs BFD Version 1, and the 50 millisecond BFD interval parameter had been adopted. The relevant command output is shown in bold in the output.

```
DeviceA# show bfd neighbors details
```

```
OurAddr
       NeighAddr
      LD/RD  RH/RS    Holdown(mult)  State     Int
172.16.1.1    172.16.1.3
     5/3    1(RH)    150 (3 )       Up   Fa1/0
Session state is UP and not using echo function.
Local Diag: 0, Demand mode: 0, Poll bit: 0
MinTxInt: 50000, MinRxInt: 50000, Multiplier: 3
Received MinRxInt: 50000, Received Multiplier: 3
Holdown (hits): 150(0), Hello (hits): 50(1364284)
Rx Count: 1351813, Rx Interval (ms) min/max/avg: 28/64/49 last: 4 ms ago
Tx Count: 1364289, Tx Interval (ms) min/max/avg: 40/68/49 last: 32 ms ago
Registered protocols: EIGRP
Uptime: 18:42:45
Last packet: Version: 0
            - Diagnostic: 0
             I Hear You bit: 1     - Demand bit: 0
             Poll bit: 0           - Final bit: 0
             Multiplier: 3         - Length: 24
             My Discr.: 3          - Your Discr.: 5
             Min tx interval: 50000    - Min rx interval: 50000
             Min Echo interval: 0
OurAddr       NeighAddr
     LD/RD  RH/RS   Holdown(mult)  State     Int
172.16.1.1    172.16.1.2

    6/1    Up        0    (3 )   Up        Fa1/0
Session state is UP and using echo function with 50 ms interval.
Local Diag: 0, Demand mode: 0, Poll bit: 0
MinTxInt: 1000000, MinRxInt: 1000000, Multiplier: 3
Received MinRxInt: 1000000, Received Multiplier: 3
Holdown (hits): 3000(0), Hello (hits): 1000(317)
Rx Count: 305, Rx Interval (ms) min/max/avg: 1/1016/887 last: 448 ms ago
Tx Count: 319, Tx Interval (ms) min/max/avg: 1/1008/880 last: 532 ms ago
Registered protocols: EIGRP
Uptime: 00:04:30
Last packet: Version: 1

         - Diagnostic: 0
             State bit: Up         - Demand bit: 0
             Poll bit: 0           - Final bit: 0
             Multiplier: 3         - Length: 24
             My Discr.: 1          - Your Discr.: 6
             Min tx interval: 1000000    - Min rx interval: 1000000
             Min Echo interval: 50000
```

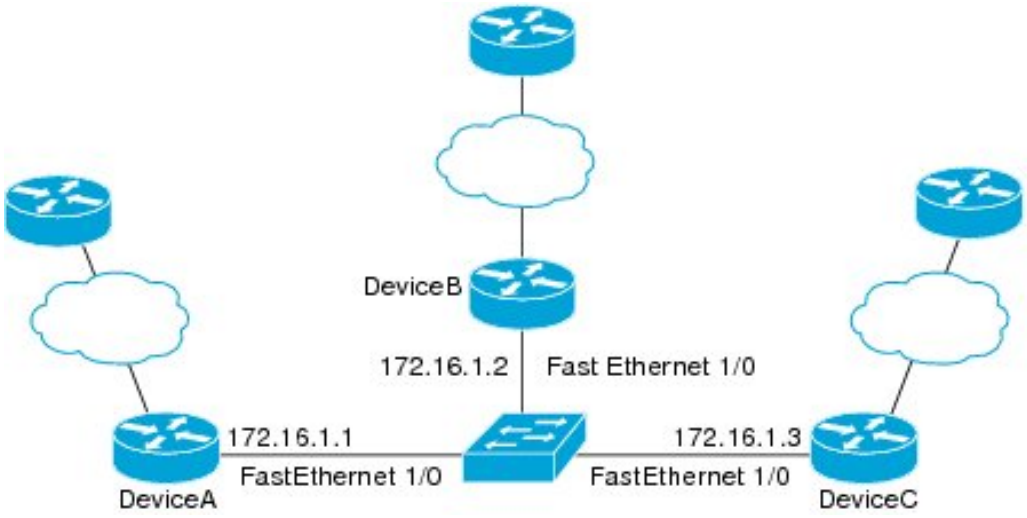
The output from the **show bfd neighbors details** command on DeviceB verifies that BFD sessions have been created and that EIGRP is registered for BFD support. As previously noted, DeviceA runs BFD Version 1, therefore echo mode is running, and DeviceC runs BFD Version 0, so echo mode does not run. The relevant command output is shown in bold in the output.

```
DeviceB# show bfd neighbors details

OurAddr       NeighAddr
     LD/RD  RH/RS   Holdown(mult)  State     Int
172.16.1.2    172.16.1.1
    1/6    Up        0    (3 )   Up        Fa1/0
Session state is UP and using echo function with 50 ms interval.
Local Diag: 0, Demand mode: 0, Poll bit: 0
MinTxInt: 1000000, MinRxInt: 1000000, Multiplier: 3
Received MinRxInt: 1000000, Received Multiplier: 3
Holdown (hits): 3000(0), Hello (hits): 1000(337)
```

```
Rx Count: 341, Rx Interval (ms) min/max/avg: 1/1008/882 last: 364 ms ago
Tx Count: 339, Tx Interval (ms) min/max/avg: 1/1016/886 last: 632 ms ago
Registered protocols: EIGRP
Uptime: 00:05:00
Last packet: Version: 1
           - Diagnostic: 0
             State bit: Up         - Demand bit: 0
             Poll bit: 0           - Final bit: 0
             Multiplier: 3         - Length: 24
             My Discr.: 6          - Your Discr.: 1
             Min tx interval: 1000000    - Min rx interval: 1000000
             Min Echo interval: 50000
OurAddr       NeighAddr

 LD/RD  RH/RS   Holdown(mult)  State     Int
172.16.1.2    172.16.1.3
     3/6    1(RH)     118  (3 )   Up        Fa1/0
Session state is UP and not using echo function.
Local Diag: 0, Demand mode: 0, Poll bit: 0
MinTxInt: 50000, MinRxInt: 50000, Multiplier: 3
Received MinRxInt: 50000, Received Multiplier: 3
Holdown (hits): 150(0), Hello (hits): 50(5735)
Rx Count: 5731, Rx Interval (ms) min/max/avg: 32/72/49 last: 32 ms ago
Tx Count: 5740, Tx Interval (ms) min/max/avg: 40/64/50 last: 44 ms ago
Registered protocols: EIGRP
Uptime: 00:04:45
Last packet: Version: 0
            - Diagnostic: 0
              I Hear You bit: 1     - Demand bit: 0
              Poll bit: 0           - Final bit: 0
              Multiplier: 3         - Length: 24
              My Discr.: 6          - Your Discr.: 3
              Min tx interval: 50000    - Min rx interval: 50000
              Min Echo interval: 0
```

The figure below shows that Fast Ethernet interface 1/0 on DeviceB has failed. When Fast Ethernet interface 1/0 on DeviceB is shut down, the BFD statistics of the corresponding BFD sessions on DeviceA and DeviceB are reduced.

Figure 4: BFD Process on Fast Ethernet Interfaces



When Fast Ethernet interface 1/0 on DeviceB fails, BFD will no longer detect DeviceB as a BFD neighbor for DeviceA or for DeviceC. In this example, Fast Ethernet interface 1/0 has been administratively shut down on DeviceB.

The following output from the **show bfd neighbors** command on DeviceA now shows only one BFD neighbor for DeviceA in the EIGRP network. The relevant command output is shown in bold in the output.

```
DeviceA# show bfd neighbors
OurAddr       NeighAddr

    LD/RD  RH/RS    Holdown(mult)  State     Int
172.16.1.1    172.16.1.3

    5/3    1(RH)      134   (3 )   Up        Fa1/0
```

The following output from the **show bfd neighbors** command on DeviceC also now shows only one BFD neighbor for DeviceC in the EIGRP network. The relevant command output is shown in bold in the output.

```
DeviceC# show bfd neighbors

OurAddr       NeighAddr

   LD/RD RH   Holdown(mult)   State     Int
172.16.1.3    172.16.1.1

    3/5  1    114   (3 )      Up        Fa1/0
```

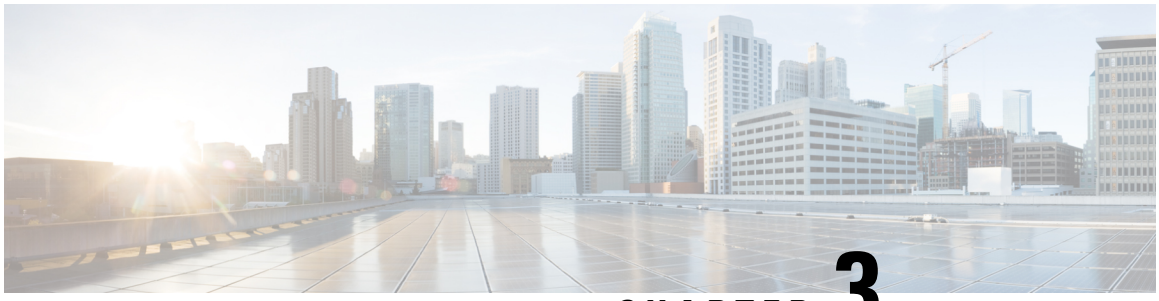
Feature History for Configuring BFD-EIGRP Support

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 16.9.2	BFD-EIGRP Support	The BFD-EIGRP Support feature configures the EIGRP with BFD so that EIGRP registers with BFD and receives all forwarding path detection failure messages from BFD.
Cisco IOS XE Cupertino 17.9.1	BFD-EIGRP Support	This feature was implemented on C9200CX-12P-2X2G, C9200CX-8P-2X2G, and C9200CX-12T-2X2G models of the Cisco Catalyst 9200CX Series Switches, which were introduced in this release.

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 BFD Support for EIGRP IPv6

- Prerequisites for BFD Support for EIGRP IPv6, on page 29
- Restrictions for BFD Support for EIGRP IPv6, on page 29
- Information About BFD Support for EIGRP IPv6, on page 29
- How to Configure BFD Support for EIGRP IPv6, on page 30
- Configuration Examples for BFD Support for EIGRP IPv6, on page 33
- Additional References, on page 34
- Feature History for BFD Support for EIGRP IPv6, on page 35

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 above mentioned 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:**	Specifies an EIGRP routing process and enters router configuration mode.

	Command or Action	Purpose
	`Device(config)# router eigrp name`	
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:

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	**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.
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:**	Configures interface-specific commands on an interface that belongs to an address family in

	Command or Action	Purpose
	`Device(config-router-af)# af-interface ethernet0/0`	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 9200 Series Switches)*
EIGRP commands: complete command syntax, command mode, command history, defaults, usage guidelines, and examples.	See the *IP Routing* section of the *Command Reference (Catalyst 9200 Series Switches)*
Configuring EIGRP	See the *Routing* section of the *Software Configuration Guide (Catalyst 9200 Switches)*

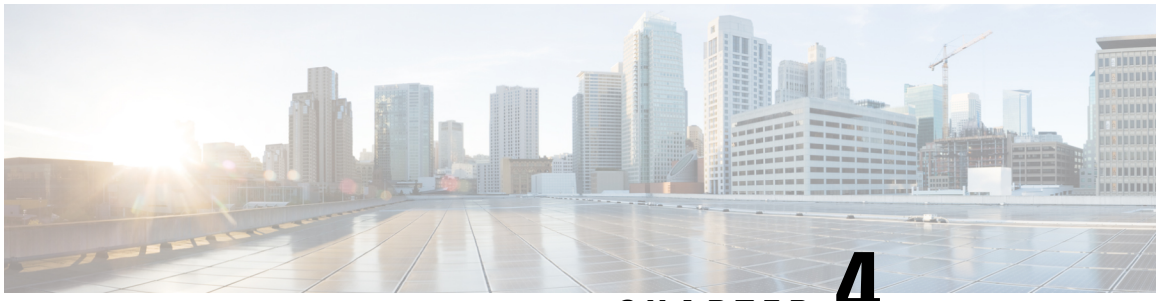
Feature History for BFD Support for EIGRP IPv6

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.
Cisco IOS XE Cupertino 17.9.1	BFD Support for EIGRP IPv6	This feature was implemented on C9200CX-12P-2X2G, C9200CX-8P-2X2G, and C9200CX-12T-2X2G models of the Cisco Catalyst 9200CX Series Switches, which were introduced in this release.

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 37
- Information About IP Unicast Routing, on page 37
- Information About IP Routing, on page 37
- Configuration Guidelines for IP Routing, on page 43
- How to Configure IP Addressing, on page 44
- How to Configure IP Unicast Routing, on page 62
- Configuration Example for Enabling IP Routing, on page 63
- Monitoring and Maintaining IP Addressing, on page 63
- Monitoring and Maintaining the IP Network, on page 64
- Feature History for IP Unicast Routing, on page 65

Restrictions for IP Unicast Routing

- The switch does not support tunnel interfaces for unicast routed traffic.
- Subnetwork Access Protocol (SNAP) address resolution is not supported on this device.

Information About 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 .

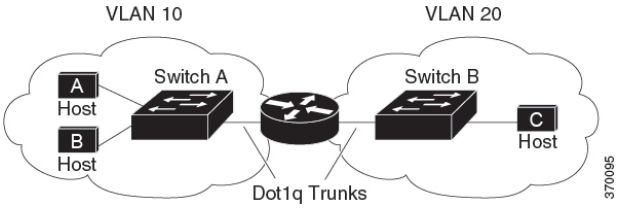
Information About IP Routing

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 5: 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 that is 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 that is 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.

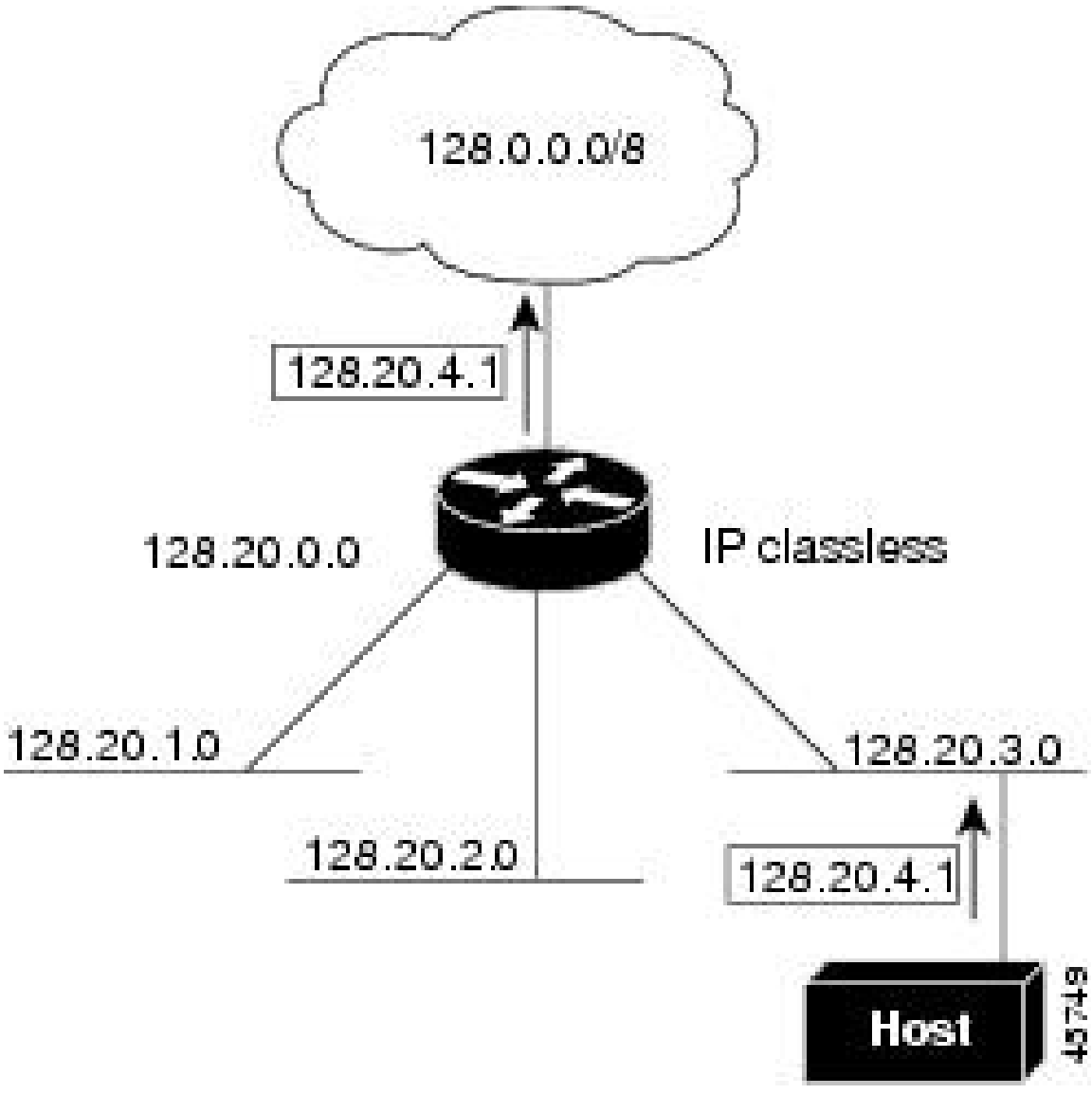
Distance-vector protocols that are supported by the switch are Routing Information Protocol (RIP), which uses a single distance metric (cost) to determine the best path. The switch also supports the Open Shortest Path First (OSPF) link-state protocol and Enhanced IGRP (EIGRP), which adds some link-state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

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 that are used to simulate a single, larger address space and is designed to relieve the pressure on the rapidly depleting Class B address space.

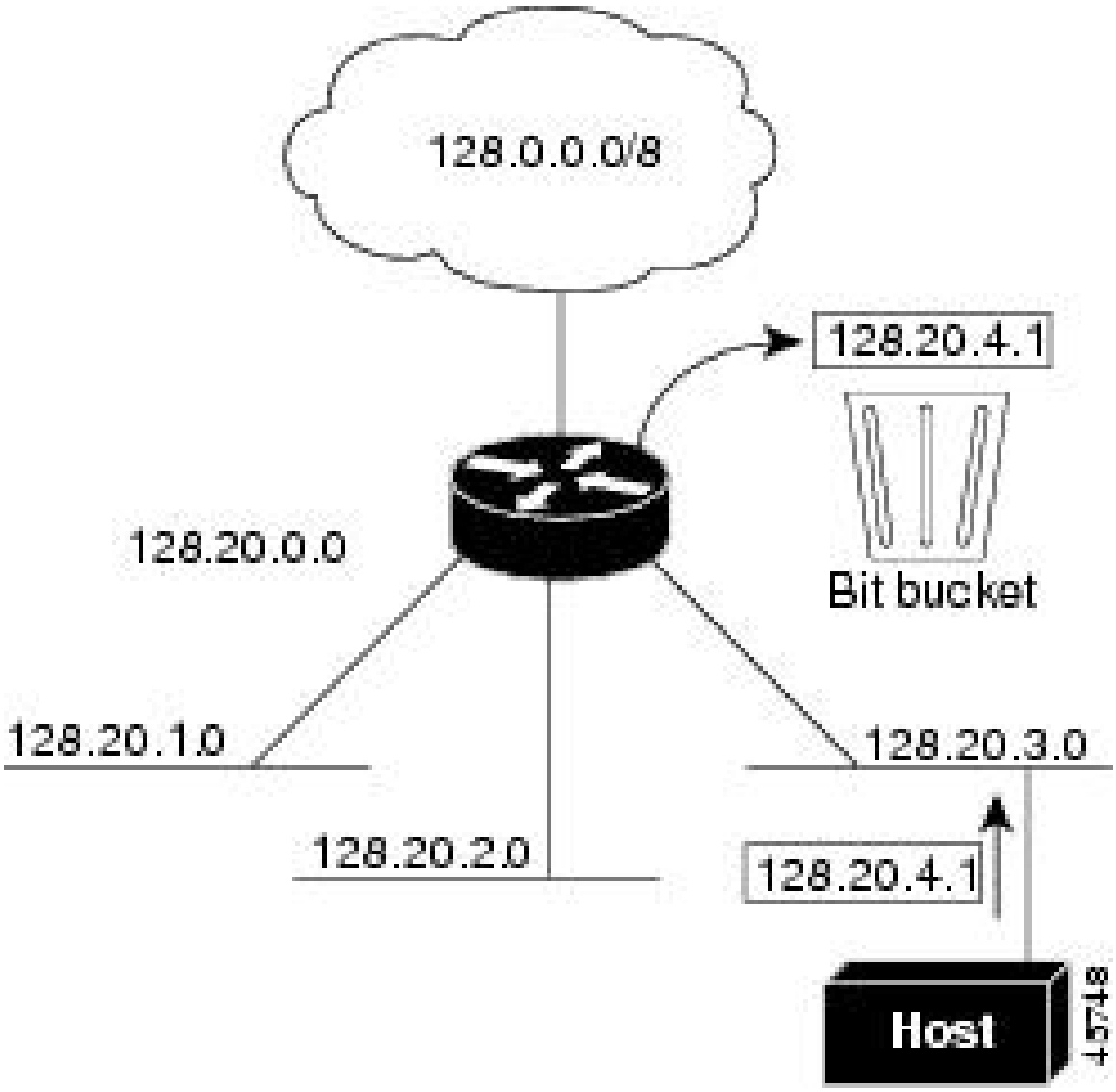
In the following figure, classless routing is enabled. When the host sends a packet to 120.20.4.1, instead of discarding the packet, the router forwards it to the best supernet route. If you disable classless routing and a router receives packets that are destined for a subnet of a network with no network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

Figure 6: IP Classless Routing



In the following 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 7: No IP Classless Routing



To prevent the device from forwarding packets that are 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.

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 that are 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 that is 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 does 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 that the device responds to network broadcasts. A broadcast is a data packet that is 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 that is 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 that is 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 that is 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 1: Default Addressing Configuration

Feature	Default Setting
IP address	None defined.
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.

Feature	Default Setting
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.

To assign IP addresses to network interfaces, 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`	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 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:** `Device#show ip route`	Verifies your entries.
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:** `Device(config)#ip subnet-zero`	Enables the use of subnet zero for interface addresses and routing updates.
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 that are destined for unrecognized subnets to the best supernet route possible, you can disable classless routing behavior.

To disable classless 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	**no ip classless** **Example:** `Device(config)#no ip classless`	Disables classless routing behavior.
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.

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.
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:**	(Optional) Sets the length of time an ARP cache entry stays in the cache. The recommended value of ARP timeout is 4 hours

	Command or Action	Purpose
	`Device(config-if)#arp timeout 20000`	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.
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.

To setting ARP encapsulation, perform this procedure:

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 1/0/2`	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.
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.

To enable proxy ARP, 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 1/0/2`	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.
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 that is 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.

Configuring 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.

To configure default gateway, 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 default-gateway** *ip-address* **Example:** `Device(config)#ip default gateway 10.1.5.1`	Sets up a default gateway (router).
Step 4	**end** **Example:** `Device(config)#end`	Returns to privileged EXEC mode.
Step 5	**show ip redirects** **Example:**	Displays the address of the default gateway router to verify the setting.

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

Configuring ICMP Router Discovery Protocol

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.

To configure ICMP router discovery protocol, 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`	Enters interface configuration mode, and specifies the Layer 3 interface to configure.
Step 4	**ip irdp** **Example:** `Device(config-if)#ip irdp`	Enables IRDP processing on the interface.

	Command or Action	Purpose
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:**	Returns to privileged EXEC mode.

	Command or Action	Purpose
	`Device(config)#end`	
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*.

Note The **ip network-broadcast** command must be configured at the ingress interface before configuring the **ip directed-broadcast** command at the egress interface. This ensures that the IP-directed broadcasts work correctly and prevents an outage from occurring after an upgrade.

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 interface to configure.
Step 4	**ip network-broadcast** **Example:** `Device(config-if)#ip network-broadcast`	Enables the ingress interface to receive and accept the network-prefix-directed broadcast packets.
Step 5	**exit** **Example:** `Device(config-if)#exit`	Returns to global configuration mode.
Step 6	**interface** *interface-id* **Example:** `Device(config)#interface gigabitethernet 1/0/3`	Enters interface configuration mode, and specifies the interface to configure.
Step 7	**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 8	**exit** **Example:** `Device(config-if)#exit`	Returns to global configuration mode.

	Command or Action	Purpose
Step 9	**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 10	**end** **Example:** `Device(config)#end`	Returns to privileged EXEC mode.
Step 11	**show ip interface** [*interface-id*] **Example:** `Device#show ip interface`	Verifies the configuration on the interface or all interfaces
Step 12	**show running-config** **Example:** `Device#show running-config`	Verifies your entries.
Step 13	**copy running-config startup-config** **Example:** `Device#copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

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:**	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 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:** `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 the configuration on the interface or all interfaces.
Step 9	**show running-config** **Example:** `Device#show running-config`	Verifies your entries.

	Command or Action	Purpose
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.

To establish an IP broadcast address, 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`	Enters interface configuration mode, and specifies the interface to configure.
Step 4	**ip broadcast-address** *ip-address* **Example:** `Device(config-if)#ip broadcast-address 128.1.255.255`	Enters a broadcast address different from the default, for example 128.1.255.255.
Step 5	**end** **Example:** `Device(config)#end`	Returns to privileged EXEC mode.

	Command or Action	Purpose
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:** `Device(config)#ip forward-protocol turbo-flood`	Uses the spanning-tree database to speed up flooding of UDP datagrams.
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.

How to Configure IP Unicast Routing

The following sections provide configuration information 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.
Step 3	**ip routing** **Example:** `Device(config)#ip routing`	Enables IP routing.
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.

What to Do Next

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

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

Configuration Example for Enabling IP Routing

This example shows how to enable IP routing:

```
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 2: 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 3: 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 4: Command to Clear IP Routes or Display Route Status

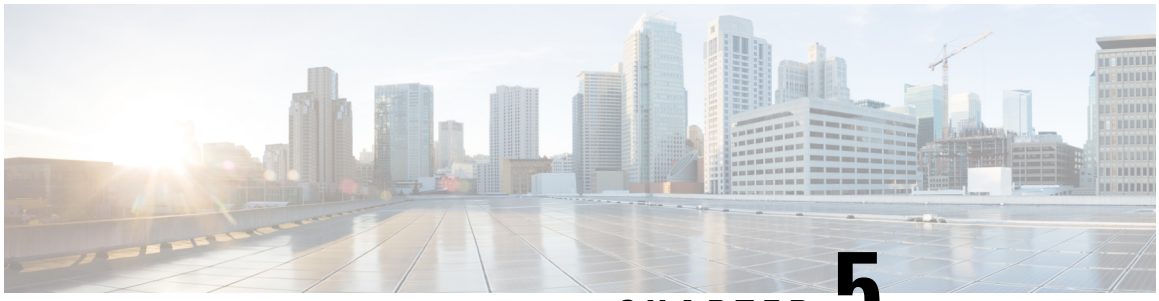
Command	Purpose
show ip route summary	Displays the current state of the routing table in summary for

Feature History for IP 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 Fuji 16.9.2	IP Unicast Routing	IP Unicast Routing is a routing process that forwards traffic to an unicast address. Layer 3 switches route packets either hrough preprogrammed static routes or through default routes.
Cisco IOS XE Amsterdam 17.3.1	New command **ip network-broadcast**	**ip network-broadcast** command was introduced to receive and accept network-prefix-directed broadcast packets.
Cisco IOS XE Cupertino 17.9.1	IP Unicast Routing	This feature was implemented on C9200CX-12P-2X2G, C9200CX-8P-2X2G, and C9200CX-12T-2X2G models of the Cisco Catalyst 9200CX Series Switches, which were introduced in this release.



CHAPTER 5

Configuring IPv6 Unicast Routing

- Information About IPv6 Unicast Routing, on page 67
- How to Configure IPv6 Unicast Routing, on page 71
- Configuration Examples for IPv6 Unicast Routing, on page 84
- Additional References, on page 86
- Feature History for IPv6 Unicast Routing, on page 86

Information About 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 and EIGRP 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 Networking Software (IOS & NX-OS)

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.

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.

Destination Guard

The IPv6 Destination Guard feature works with IPv6 neighbor discovery to ensure that the device performs address resolution only for those addresses that are known to be active on the link. It relies on the address glean functionality to populate all destinations active on the link into the binding table and then blocks resolutions before they happen when the destination is not found in the binding table.

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

MTU Path Discovery

IPv6 MTU Path Discovery allows a host to dynamically discover and adjust to differences in the maximum transmission unit (MTU) size of every link along a given data path.

As in IPv4, path MTU discovery in IPv6 allows a host to dynamically discover and adjust to differences in the MTU size of every link along a given data path. In IPv6, however, fragmentation is handled by the source of a packet when the path MTU of one link along a given data path is not large enough to accommodate the size of the packets. Having IPv6 hosts handle packet fragmentation saves IPv6 device processing resources and helps IPv6 networks run more efficiently.

For more information, see *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 that are derived from routing protocols. Therefore, PBR gives you more control over routing by extending and complementing the existing mechanisms that are 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 that are destined to site-local addresses.
- Tunneling protocols, such as IPv4-to-IPv6 or IPv6-to-IPv4.
- The switch as a tunnel endpoint supporting IPv4-to-IPv6 or IPv6-to-IPv4 tunneling protocols.
- IPv6 Web Cache Communication Protocol (WCCP).

IPv6 Feature Limitations

Because IPv6 is implemented in switch hardware, some limitations occurs 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 change 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 5: 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 show 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 that are 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 that is 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** {**advanced** |**vlan**} **Example:** `Device(config)# sdm prefer vlan`	Selects an SDM template. • **advanced**—Sets the switch to the advanced template. • **vlan**—Maximizes VLAN configuration on the switch with no routing that is supported in hardware.
Step 4	**end** **Example:**	Returns to privileged EXEC mode.

	Command or Action	Purpose
	`Device(config)# end`	
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`
		• 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

	Command or Action	Purpose
	`Device(config-if)# ipv6 enable`	to communicate with nodes on the same link.
Step 10	**exit** **Example:** `Device(config-if)# exit`	Returns to global configuration mode.
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** **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 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*	• 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

	Command or Action	Purpose
	• **ipv6 address***autoconfig* • **ipv6 address***dhcp*	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:** `Device(config)# end`	Returns to privileged EXEC mode.
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 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**	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 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.
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.

	Command or Action	Purpose
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. 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*] **Example:** `Device(config)# ipv6 route 2001:0DB8::/32 gigabitethernet2/0/1 130`	Configures a static IPv6 route. • *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 that is documented in RFC 2373, specified in hexadecimal using 16-bit values between colons.

	Command or Action	Purpose
		• *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*]
	Example:
	`Device# show ipv6 static 2001:0DB8::/32 interface gigabitethernet2/0/1`
	or
	`Device# show ipv6 route static`
		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.
		• **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:

	Command or Action	Purpose
		• 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 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.
Step 4	Do one of the following: • **match length** *minimum-length maximum-length*	Specifies the match criteria. You can specify any or all of the following: • Matches the Level 3 length of the packet.

	Command or Action	Purpose

	• **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`
		• 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 ipv6 default next-hop** *global-ipv6-address* [*global-ipv6-address...*]
	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`
		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 maps 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 that are 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 6: Command for Monitoring IPv6

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

Command	Purpose
show ipv6 traffic	Displays IPv6 traffic statistics.

Configuration Examples for IPv6 Unicast Routing

The following sections show 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 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 that is 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*

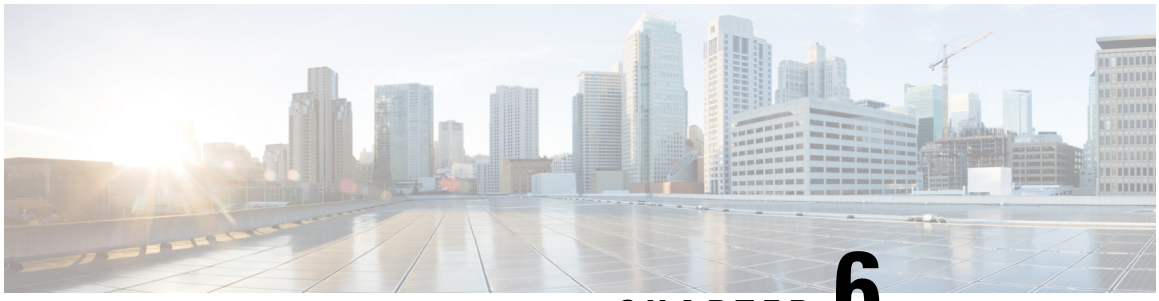
Feature History for IPv6 Unicast Routing

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 Fuji 16.9.2	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.
Cisco IOS XE Cupertino 17.9.1	IPv6 Unicast Routing	This feature was implemented on C9200CX-12P-2X2G, C9200CX-8P-2X2G, and C9200CX-12T-2X2G models of the Cisco Catalyst 9200CX Series Switches, which were introduced in this release.

Use the Cisco Feature Navigator to find information about platform and software image support.



CHAPTER 6

Configuring RIP

- Information About RIP, on page 87
- How to Configure Routing Information Protocol, on page 88
- Configuration Examples for Routing Information Protocol, on page 97
- Feature History for Routing Information Protocol, on page 98

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 7: 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:**	(Optional) Defines a neighboring router with which to exchange routing information. This step allows routing updates from RIP

	Command or Action	Purpose
	`Device(config-router)# neighbor 10.2.5.1`	(normally a broadcast protocol) to reach nonbroadcast networks.
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

	Command or Action	Purpose
		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.
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:**	Enters interface configuration mode, and specifies the interface to configure.

	Command or Action	Purpose
	`Device(config)# interface gigabitethernet 1/0/1`	
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.
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.
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.

	Command or Action	Purpose
		• **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.

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:**	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 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.
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:**	(Optional) Saves your entries in the configuration file.

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

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:** `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 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:**	Disables split horizon on the interface.

	Command or Action	Purpose
	`Device(config-if)# no ip split-horizon`	
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.

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
```

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
```

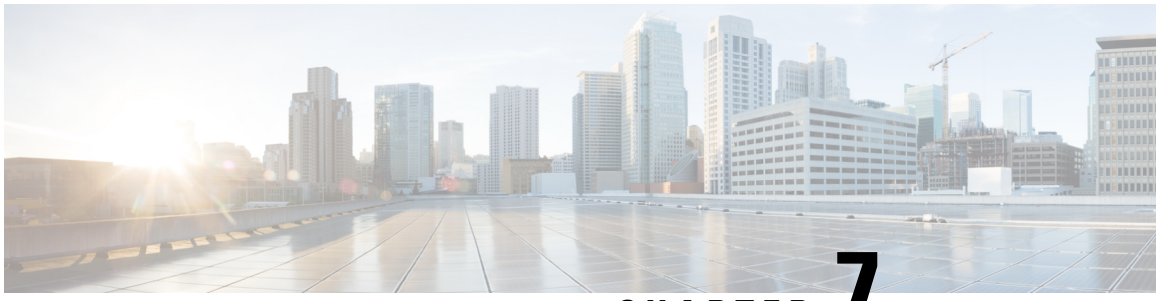
Feature History for Routing Information Protocol

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 Fuji 16.9.2	Routing Information Protocol	The Routing Information Protocol is an interior gateway protocol (IGP) created for use in small and homogeneous networks.
Cisco IOS XE Cupertino 17.9.1	Routing Information Protocol	This feature was implemented on C9200CX-12P-2X2G, C9200CX-8P-2X2G, and C9200CX-12T-2X2G models of the Cisco Catalyst 9200CX Series Switches, which were introduced in this release.

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 99
- How to Configure OSPF, on page 102
- Monitoring OSPF, on page 115
- Configuration Examples for OSPF, on page 116
- Configuration Examples for OSPF, on page 116
- Example: Configuring Basic OSPF Parameters, on page 116
- Feature History for Open Shortest Path First, on page 116

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.

Note The Network Essentials license allows configuration of 1000 routes only. To configure more than 1000 routes, Network Advantage license is required.

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 100
- OSPF NSF Capability, on page 100

OSPF NSF Awareness

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

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 8: 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 route ty is Type 2.
Default metric	Built-in, automatic metric translation, as appropriate for each routing protocol.

Feature	Default Setting
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.
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.

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 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. **Note** OSPF for Routed Access supports only one OSPFv2 and one OSPFv3 instance with a maximum number of 1000 dynamically learned routes.
Step 4	**network** address wildcard-mask **area** area-id **Example:** `Device(config-router)#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 5	**end** **Example:** `Device(config-router)#end`	Returns to privileged EXEC mode.
Step 6	**show ip protocols** **Example:** `Device#show ip protocols`	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 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	• Displays information about OSPF interfaces. • Displays general information about OSPF routing processes.

	Command or Action	Purpose
	`Device# show ipv6 ospf 21`	
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 **Example:** `Device(config-if)#ip ospf 8`	(Optional) Explicitly specifies the cost of sending a packet on the interface.
Step 5	**ip ospf retransmit-interval** seconds **Example:**	(Optional) Specifies the number of seconds between link state advertisement transmissions.

	Command or Action	Purpose
	`Device(config-if)#ip ospf transmit-interval 10`	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.
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:**	Returns to privileged EXEC mode.

	Command or Action	Purpose
	`Device(config)#end`	
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:** `Device#show ip ospf neighbor detail`	Displays NSF awareness status of neighbor switch. The output matches one of these examples: • *Options is 0x52* *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:**	Enters global configuration mode.

	Command or Action	Purpose
	`Device#configure terminal`	
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.
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.

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

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.

	Command or Action	Purpose
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.
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*	(Optional) Configures route calculation timers.

	Command or Action	Purpose
	Example: `Device(config)#timers throttle spf 200 100 100`	• *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.
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:**	Enters global configuration mode.

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

	Command or Action	Purpose
	`Device#configure terminal`	
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.
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 9: Show IP OSPF Statistics Commands

Command	Purpose
show ip ospf [*process-id*]	Displays general inf processes.

Command	Purpose
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
```

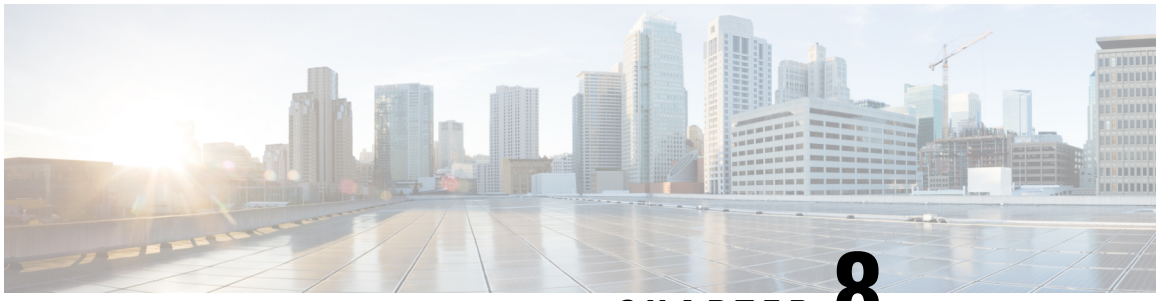
Feature History for Open Shortest Path First

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 Fuji 16.9.2	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.
Cisco IOS XE Cupertino 17.9.1	Open Shortest Path First	This feature was implemented on C9200CX-12P-2X2G, C9200CX-8P-2X2G, and C9200CX-12T-2X2G models of the Cisco Catalyst 9200CX Series Switches, which were introduced in this release.

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 OSPF Link-State Database Overload Protection

- Information About OSPF Link-State Database Overload Protection, on page 119
- How to Configure OSPF Link-State Database Overload Protection, on page 120
- Configuration Examples for OSPF Link-State Database Overload Protection, on page 122
- Feature History for OSPF Link-State Database Overload Protection, on page 124

Information About OSPF Link-State Database Overload Protection

The OSPF Link-State Database Overload Protection feature allows you to limit the number of nonself-generated link-state advertisements (LSAs) for a given Open Shortest Path First (OSPF) process or OSPFv3 process. Excessive LSAs generated by other devices in the OSPF domain can substantially drain the CPU and memory resources of the device.

The OSPF Link-State Database Overload Protection feature is applicable to OSPF, OSPFv2 and OSPFv3.

Benefits of Using OSPF Link-State Database Overload Protection

The OSPF Link-State Database Overload Protection feature provides a mechanism at the OSPF level to limit the number of nonself-generated LSAs for a given OSPF process. When other devices in the network have been misconfigured, they may generate a high volume of LSAs, for instance, to redistribute large numbers of prefixes. This protection mechanism prevents devices from receiving a large number of LSAs and therefore experiencing CPU and memory shortages.

Overview of OSPF Link-State Database Overload Protection

When the OSPF Link-State Database Overload Protection feature is enabled, the device keeps a count of the number of nonself-generated LSAs that it has received. When the configured threshold number of LSAs is reached, an error message is logged. When the configured maximum number of LSAs is exceeded, the device sends a notification. If the count of received LSAs is still higher than the configured maximum after one minute, the OSPF process takes down all adjacencies and clears the OSPF database. In this ignore state, all OSPF packets received on any interface that belong to this OSPF process are ignored and no OSPF packets are generated on any of these interfaces. The OSPF process remains in the ignore state for the time configured

by the **ignore-time** keyword of the **max-lsa** command. Each time the OSPF process gets into an ignore state a counter is incremented. If this counter exceeds the number of times configured by the **ignore-count** keyword, the OSPF process stays permanently in the same ignore state and manual intervention is required to get the OSPF process out of the ignore state. You can get the OSPF process out of the permanent ignore state by restarting the OSPF process. The ignore state counter is reset to 0 when the OSPF process remains in the normal state of operation for the amount of time that was specified by the **reset-time** keyword. If the **warning-only** keyword of the **max-lsa** command is configured, the OSPFprocess will send only a warning that the LSA maximum has been exceeded.

How to Configure OSPF Link-State Database Overload Protection

Limiting the Number of Non Self-Generated LSAs for an OSPF Process

To configure a limit for the number of non self-generated LSAs for an OPSF process, 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 ospf***process-id* **Example:** `Device(config)# router ospf 1`	Enables OSPF routing. The *process-id* argument identifies the OSPF process.
Step 4	**router-id** *ip-address* **Example:** `Device(config-router)# router-id 10.0.0.1`	Specifies a fixed router ID for an OSPF process.
Step 5	**log-adjacency-changes** [**detail**] **Example:** `Device(config-router)# log-adjacency-changes`	Configures the device to send a syslog message when an OSPF neighbor goes up or down.
Step 6	**max-lsa** *maximum number* [*threshold-percentage*] [**warning-only**] [**ignore-time** *minutes*] [**ignore-count***count-number*] [**reset-time***minutes*] **Example:** `Device(config-router)# max-lsa 12000`	Limits the number of non self-generated LSAs that an OSPF routing process can keep in the OSPF link-state database (LSDB). • The default limit for the number of non self-generated LSAs is 50,000 LSAs.

	Command or Action	Purpose
		• The default value for the *threshold* argument is 75 percent. • The default value for the **ignore-time** argument is 5 minutes. • The default value for the **reset-time** argument is 10 minutes. • The default value for the **ignore-count** argument is 5 counts.
Step 7	**network** *ip-address wildcard-mask* **area** *area-id* **Example:** `Device(config-router)# network 209.165.201.1 255.255.255.255 area 0`	Defines the interfaces on which OSPF runs and defines the area ID for those interfaces.
Step 8	**end** **Example:** `Device(config-router)# end`	
Step 9	**show ip ospf** [*process-id area-id*] **database** [**database-summary**] **Example:** `Device# show ip ospf 2000 database database-summary`	Displays lists of information related to the OSPF database for a specific device. Use this command to verify the number of non self-generated LSAs on a device.

Limiting the Number of Non Self-Generated LSAs for an OSPFv3 Process

To configure a limit for the number of non self-generated LSAs for an OPSFv3 process, 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`	Enables OSPFv3 routing. The *process-id* argument identifies the OSPFv3 process.

	Command or Action	Purpose
Step 4	**router-id** *ip-address* **Example:** `Device(config-router)# router-id 10.0.0.1`	Specifies a fixed router ID for an OSPF process.
Step 5	**log-adjacency-changes** [**detail**] **Example:** `Device(config-router)# log-adjacency-changes`	Configures the device to send a syslog message when an OSPF neighbor goes up or down.
Step 6	**max-lsa** *maximum number* [*threshold-percentage*] [**warning-only**] [**ignore-time** *minutes*] [**ignore-count***count-number*] [**reset-time***minutes*] **Example:** `Device(config-router)# max-lsa 12000`	Limits the number of non self-generated LSAs that an OSPF routing process can keep in the OSPF link-state database (LSDB). • The default limit for the number of non self-generated LSAs is 50,000 LSAs. • The default value for the *threshold* argument is 75 percent. • The default value for the **ignore-time** argument is 5 minutes. • The default value for the **reset-time** argument is 10 minutes. • The default value for the **ignore-count** argument is 5 counts.
Step 7	**end** **Example:** `Device(config-router)# end`	
Step 8	**show ospfv3** [*process-id area-id*] **database** [**database-summary**] **Example:** `Device# show ospfv3 2000 database database-summary`	Displays lists of information related to the OSPF database for a specific device. Use this command to verify the number of non self-generated LSAs on a device.

Configuration Examples for OSPF Link-State Database Overload Protection

Example: Setting a Limit for LSA Generation

In the following example, the device is configured to not accept any more non self-generated LSAs once a maximum of 14,000 has been exceeded:

```
Device(config)# router ospf 1
Device(config-router)# router-id 192.168.0.1
Device(config-router)# log-adjacency-changes
Device(config-router)# max-lsa 14000
Device(config-router)# area 33 nssa
Device(config-router)# network 192.168.0.10.0.0.0 area 1
Device(config-router)# network 192.168.5.10.0.0.0 area 1
Device(config-router)# network 192.168.2.10.0.0.0 area 0
```

In the following example, the device is configured to not accept any more non self-generated LSAs once a maximum of 12,000 has been exceeded for an OPSFv3 process:

```
Device> enable
Device# configure terminal
Device(config)# router ospfv3 1
Device(config-router)# router-id 10.0.0.1
Device(config-router)# log-adjacency-changes
Device(config-router)# max-lsa 12000
```

In the following example, the **show ip ospf** command is entered to confirm the configuration:

```
Device# show ip ospf 1
Routing Process "ospf1" with ID 192.168.0.1
Supports only single TOS(TOS0)routes
Supports opaque LSA
Supports Link-local Signaling(LLS)
Supports area transit capability
Maximum number of nonself-generated LSA allowed 14000
Threshold for warning message75%
Ignore-time 5minutes,reset-time 10minutes
Ignore-countallowed 5,currentignore-count 0
```

In the following example, the output is displayed when the **show ip ospf** command is entered when the device is in the ignore state:

```
Device# show ip ospf 1
Routing Process "ospf1" with ID 192.168.0.1
Supports only single TOS(TOS0)routes
Supports opaque LSA
Supports Link-local Signaling(LLS)
Supports area transit capability
Maximum number of nonself-generated LSA allowed 14000
Threshold for warning message 75%
Ignore-time 5minutes, reset-time 10minutes
Ignore-count allowed 5,current ignore-count 1
Ignoring all neighbors due to max-lsa limit, time remaining: 00:04:52
```

The following output is displayed when the **show ip ospf** command is entered after the device left the ignore state:

```
Device# show ip ospf 1
Routing Process "ospf 1" with ID 192.168.0.1
Supports only single TOS(TOS0) routes
Supports opaque LSA Supports Link-local Signaling (LLS)
Supports area transit capability
Maximum number of non self-generated LSA allowed 14000
Threshold for warning message 75%
```

```
Ignore-time 5 minutes, reset-time 10 minutes
Ignore-count allowed 5, current ignore-count 1- time remaining: 00:09:51
```

The following output is displayed when the **show ip ospf** command is entered for a device that is permanently in the ignore state:

```
Device# show ip ospf 1
Routing Process "ospf 1" with ID 192.168.0.1
Supports only single TOS(TOS0) routes
Supports opaque LSA Supports Link-local Signaling (LLS)
Supports area transit capability
Maximum number of non self-generated LSA allowed 14000
Threshold for warning message 75%
Ignore-time 5 minutes, reset-time 10 minutes
Ignore-count allowed 5, current ignore-count 6
Permanently ignoring all neighbors due to max-lsa limit
```

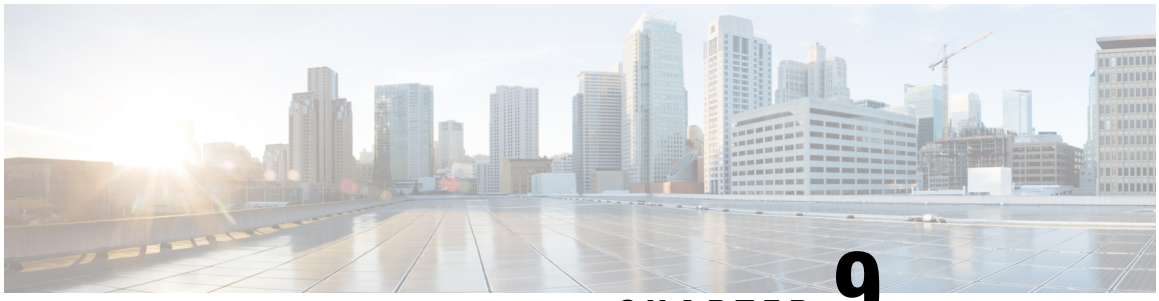
Feature History for OSPF Link-State Database Overload Protection

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

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

Release	Feature	Feature Information
Cisco IOS XE Dublin 17.11.1	OSPF Link-State Database Overload Protection	The OSPF Link-State Database Overload Protection feature allows yo of non self-generated link-state advertisements (LSAs) for a given OSP LSAs generated by other routers in the OSPF domain can substantiall memory resources of the device. The default limit for the number of non self-generated LSAs is 50,000

Use 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 OSPF Limit on Number of Redistributed Routes

-
- Restrictions for OSPF Limit on Number of Redistributed Routes, on page 125
- Prerequisites for OSPF Limit on Number of Redistributed Routes, on page 125
- Information About OSPF Limit on Number of Redistributed Routes, on page 125
- How to Configure an OSPF Limit on the Number of Redistributed Routes, on page 126
- Configuration Examples for OSPF Limit on Number of Redistributed Routes, on page 130
- Feature History for OSPF Limit on Number of Redistributed Routes, on page 131

Restrictions for OSPF Limit on Number of Redistributed Routes

OSPFv3 Limit on Number of Redistributed Routes is supported only for the IPv6 address family.

Prerequisites for OSPF Limit on Number of Redistributed Routes

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

Information About OSPF Limit on Number of Redistributed Routes

OSPF supports a user-defined maximum number of prefixes (routes) that can be redistributed into OSPF from other protocols or other OSPF 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 OSPF for a network that allows redistribution of Border Gateway Protocol (BGP) into OSPF, the network can get severely flooded. Limiting the number of redistributed routes prevents this potential problem.

From Cisco IOS XE Dublin 17.11.1, the command **redistribute maximum-prefix** *maximum[threshold]* is enabled with the default number of routes set at 10240 routes. The default number of routes is to protect the OSPF processes from being flooded with routes. You can still configure the number of routes using the **redistribute maximum-prefix** command.

The OSPF Limit on Number of Redistributed Routes feature is applicable to OSPF, OSPFv2 and OSPFv3.

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

The following sections provide information on configuring an OSPF 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 OSPF.

Limiting the Number of OSPF Redistributed Routes

This task describes how to limit the number of OSPF 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 ospf** *process-id* **Example:** `Device(config)# router ospf 1`	Configures an OSPF routing process.
Step 4	**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 5	**redistribute maximum-prefix** *maximum* [*threshold*] **Example:**	Sets a maximum number of IP prefixes that are allowed to be redistributed into OSPF.

	Command or Action	Purpose
	`Device(config-router-af)# redistribute maximum-prefix 100 80`	• The default value for the *maximum* argument is set at 10240 routes. • 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 6	**end** **Example:** `Device(config-router)# end`	Exits router configuration mode.

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:**	Redistributes routes from one routing domain into another routing domain.

	Command or Action	Purpose
	`Device(config-router-af)# redistribute eigrp 10`	
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. • The default value for the *maximum* argument is set at 10240 routes. • 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 OSPF

To request a warning message when the number of routes redistributed into OSPF 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 ospf** *process-id* **Example:** `Device(config)# router ospf 1`	Configures an OSPF routing process.

	Command or Action	Purpose
Step 4	**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 5	**redistribute maximum-prefix** *maximum* [*threshold*] [**warning-only**] **Example:** `Device(config-router-af)# redistribute maximum-prefix 1000 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 OSPF. • 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 6	**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.

	Command or Action	Purpose
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.
Step 6	**redistribute maximum-prefix** *maximum* [*threshold*] [**warning-only**] **Example:** `Device(config-router-af)# redistribute maximum-prefix 1000 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. • 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 OSPF Limit on Number of Redistributed Routes

The following sections provide configuration examples for OSPF Limit on Number of Redistributed Routes.

Example: OSPF Limit on Number of Redistributed Routes

This example shows how to set a maximum of 1200 prefixes that can be redistributed into the OSPF 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 ospf 1
Device(config-router-af)# redistribute static subnets
Device(config-router-af)# redistribute maximum-prefix 1200 80
```

This example shows how to set a maximum of 1200 prefixes that can be redistributed into the OSPFv3 process 1.

```
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 ospf 11
Device(config-router-af)# redistribute eigrp 10 subnets
Device(config-router-af)# redistribute maximum-prefix 600 85 warning-only
```

This example shows how to enable two warnings to be logged for an OSSPv3 process.

```
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
```

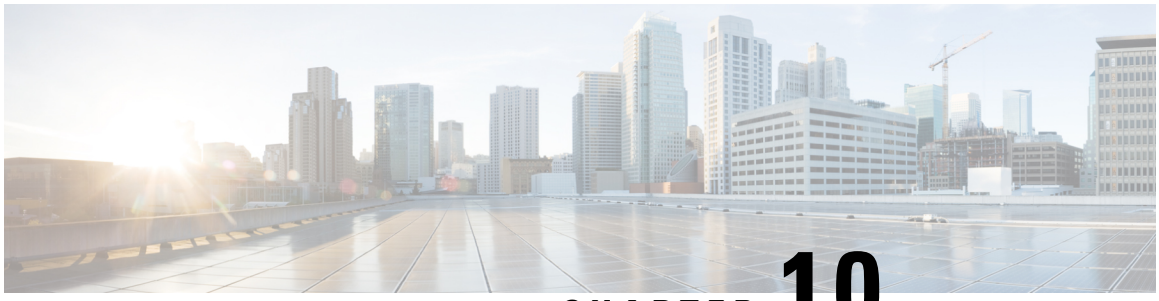
Feature History for OSPF Limit on Number of Redistributed Routes

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	OSPF Limit on Number of Redistributed Routes	OSPF supports a user-defined maximum number of prefixes (routes) that can be redistributed into OSPFv3 from other protocols or other OSPFv3 processes.
Cisco IOS XE Cupertino 17.9.1	OSPF Limit on Number of Redistributed Routes	This feature was implemented on C9200CX-12P-2X2G, C9200CX-8P-2X2G, and C9200CX-12T-2X2G models of the Cisco Catalyst 9200CX Series Switches, which were introduced in this release.
Cisco IOS XE Dublin 17.11.1	Default Value for Number of Redistributed Routes	Sets a default value of 10240 routes for the **redistribute maximum-prefix** command. The default is to protect the device from being flooded with routes.

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 EIGRP

- Information About EIGRP, on page 133
- How to Configure EIGRP, on page 138
- Monitoring and Maintaining EIGRP, on page 145
- Feature History for EIGRP, on page 146

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 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 that are 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 that 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 that are 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 that is 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 that are learned by other IP routing protocols.

EIGRP Nonstop Forwarding

The device stack supports two levels of EIGRP nonstop forwarding:

- EIGRP NSF Awareness
- EIGRP NSF Capability

EIGRP NSF Awareness

When the neighboring device is NSF-capable, the Layer 3 device continues to forward packets from the neighboring device during the interval between the primary Route Processor (RP) in a device 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

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 that is directed toward the device stack. EIGRP peer routers maintain the routes that are 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

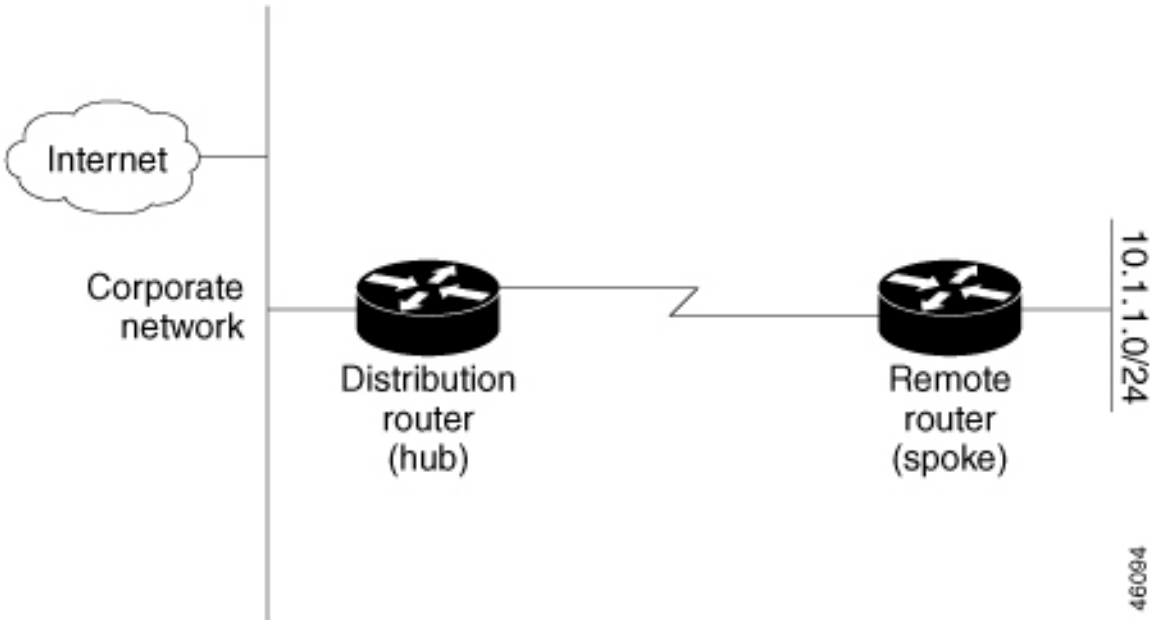
for the remote device to have a complete routing table. Generally, the distribution device need not send anything more than a default route to the remote device.

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 8: 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 that is 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.

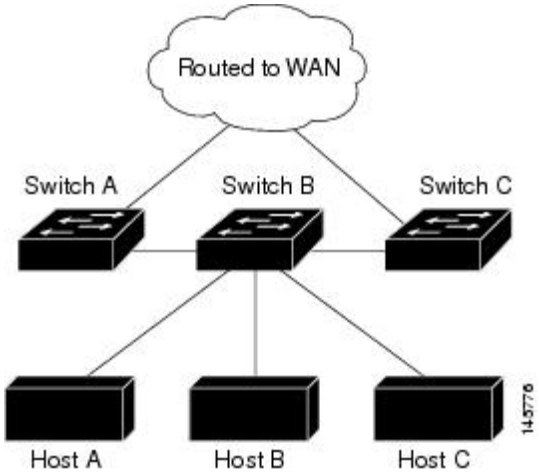
In a network using EIGRPv6 stub routing, the only allowable route for IPv6 traffic to the user is through a switch that is configured with EIGRPv6 stub routing. The switch sends the routed traffic to interfaces that are configured as user interfaces or are connected to other devices.

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 9: 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 devices on your network that are configured for IGRP, and you want to change to EIGRP, you must designate transition devices 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 10: 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.
Default metric	Only connected routes and interface static routes can be redistributed default metric. The metric includes: • Bandwidth: 0 or greater kb/s. • Delay (tens of microseconds): 0 or any positive number that is a n 39.1 nanoseconds. • Reliability: any number between 0 and 255 (255 means 100 perc reliability). • Loading: effective bandwidth as a number between 0 and 255 (2 percent loading). • MTU: maximum transmission unit size of the route in bytes. 0 or ar 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.

Feature	Default Setting
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 Allows Layer 3 switche forwarding packets from a neighboring NSF-capable router during 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.
Traffic-share	Distributed proportionately to the ratios of the metrics.
Variance	1 (equal-cost load-balancing).

Configuring Basic EIGRP Parameters

To configure basic EIGRP 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.

	Command or Action	Purpose
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 devices 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. **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:**	Enters interface configuration mode, and specifies the Layer 3 interface to configure.

	Command or Action	Purpose
	`Device(config-router)#interface gigabitethernet 1/0/1`	
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.

To configure EIGRP interfaces, 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	**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.
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.

	Command or Action	Purpose
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** **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	**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.
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

	Command or Action	Purpose
		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 11: 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
show ip eigrp topology [*autonomous-system-number*] | [[*ip-address*] *mask*]]	Displays the EIGRP topology tab

Command	Purpose
show ip eigrp traffic [*autonomous-system-number*]	Displays the number of packets sent process.

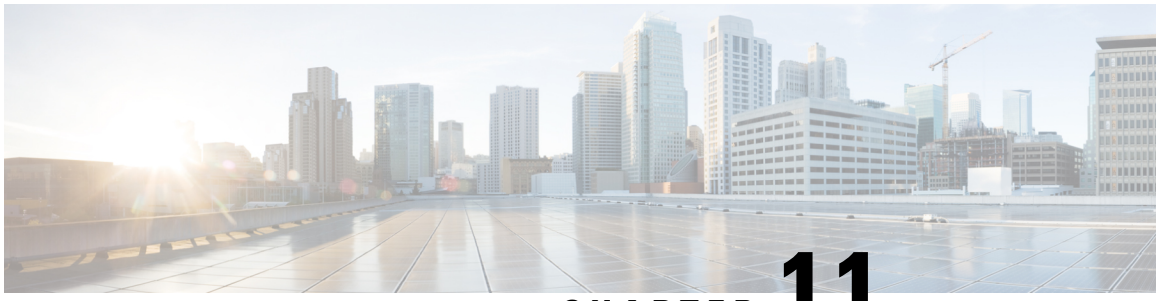
Feature History for EIGRP

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 16.9.2	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.
Cisco IOS XE Cupertino 17.9.1	EIGRP	This feature was implemented on C9200CX-12P-2X2G, C9200CX-8P-2X2G, and C9200CX-12T-2X2G models of the Cisco Catalyst 9200CX Series Switches, which were introduced in this release.

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



CHAPTER 11

Configuring IS-IS Routing

- Information About IS-IS Routing, on page 147
- How to Configure IS-IS, on page 149
- Monitoring and Maintaining IS-IS, on page 158
- Feature History for IS-IS, on page 158

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 network reorganizes itself into a backbone area that is made up of all the connected set of Level 2 devices that are 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.

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 that are 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 12: 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.

Feature	Default Setting
LSP generation throttling timers	Maximum interval between two consecutive occurrences—5000 millise Initial LSP generation delay—50 milliseconds. Hold time between the first and second LSP generation—200 milliseco
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 a n Nonstop Forwarding-capable router during hardware or software chang
Partial route computation (PRC) throttling timers	Maximum PRC wait interval—5000 milliseconds. Initial PRC calculation delay after a topology change—50 milliseconds Hold time between the first and second PRC calculation—200 millisec
Partition avoidance	Disabled.
Password	No area or domain password is defined, and authentication is disabled.
Set-overload-bit	Disabled. When enabled, if no arguments are entered, the overload bit i immediately and remains set until you enter the **no set-overload-bit** co
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:**	Enters global configuration mode.

	Command or Action	Purpose
	`Device#configure terminal`	
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.
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.

	Command or Action	Purpose
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*]	(Optional) Forces a default route into the IS-IS routing domain. When you enter the **route-map** *map-name* command, the routing

	Command or Action	Purpose
	Example: `Device(config-router)#default-information originate route-map map1`	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.
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

	Command or Action	Purpose
		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.
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).

	Command or Action	Purpose
		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 that are 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:**	(Optional) Specifies the number of IS-IS hello packets that a neighbor must miss before the device declares the adjacency as down. The range is from 3 to 1000; default is 3.

	Command or Action	Purpose
	`Device(config-if)#isis hello-multiplier 5`	**Note** Using a smaller hello multiplier causes fast convergence, but might result in routing instability.
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 that is 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.

	Command or Action	Purpose
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:** `Device(config)#end`	Returns to privileged EXEC mode.

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 13: 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*

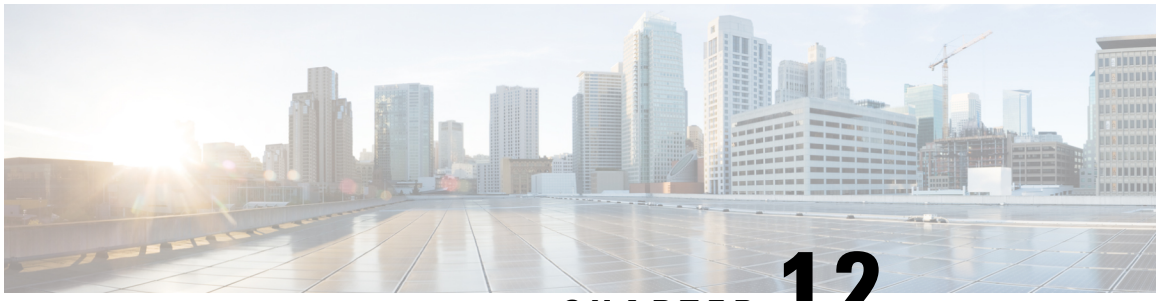
Feature History for IS-IS

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 16.9.2	IS-IS Routing	Integrated Intermediate System-to-Intermediate System (IS-IS) is an ISO dynamic routing protocol (described in ISO 105890).
Cisco IOS XE Cupertino 17.9.1	IS-IS Routing	This feature was implemented on C9200CX-12P-2X2G, C9200CX-8P-2X2G, and C9200CX-12T-2X2G models of the Cisco Catalyst 9200CX Series Switches, which were introduced in this release.

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



CHAPTER 12

Configuring VRF-lite

- Information About VRF-lite, on page 161
- Guidelines for Configuring VRF-lite, on page 162
- How to Configure VRF-lite, on page 163
- Additional Information for VRF-lite, on page 177
- Verifying VRF-lite Configuration, on page 178
- Configuration Examples for VRF-lite, on page 179
- Additional References for VRF-Lite, on page 183
- Feature History for Multicast VRF-lite, on page 183

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 edge (PE) routers exchange routing information with CE devices by using static routing or a routing protocol such as BGP, RIPv1, or RIPv2.

  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 (iBGP).

- 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. 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 allows a CE device 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.

To configure VRF, create a VRF table and specify the Layer 3 interface associated with the VRF.

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.
- VRF-lite lets multiple customers share the same physical link between the PE and the CE.
- 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

- The Cisco Catalyst switch supports PIM-SM and PIM-SSM protocols.

IPv6 specific

- VRF-aware OSPFv3, 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.

	Command or Action	Purpose
Step 11	**server-private** {*ip-address* | *name*} [**nat**] [**single-connection**] [**port** *port-number*] [**timeout** *seconds*] [**key** [**0** | **7**] *string*] **Example:** `Device(config-sg-tacacs+)# server-private 10.1.1.1 port 19 key cisco`	Configures the IP address of the private TACACS+ server for the group server.
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

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.

	Command or Action	Purpose
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:** `Device(config-if)# vrf forwarding vrf-name`	Associates the VRF with the Layer 3 interface.
Step 10	**ip address** *ip-address* **mask** **Example:** `Device(config-if)# ip address ip-address mask`	Configures IP address for the Layer 3 interface.

	Command or Action	Purpose
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 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:**	Names the VRF and enters VRF configuration mode.

	Command or Action	Purpose
	`Device(config)# vrf definition vrf-name`	
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).
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.
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

Procedure

	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.
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

Procedure

	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

Procedure

	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

Procedure

	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

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 2	**vrf definition** *vrf-name* **Example:**	Names the VRF and enters VRF configuration mode.

	Command or Action	Purpose
	`Device(config)# vrf definition vrf-name`	
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

Procedure

	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:** `Device# show ipv6 vrf [brief | detail | interfaces] [vrf-name]`	Verifies the configuration. Displays information about the configured VRFs.

	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.

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

Procedure

	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

Procedure

	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

Procedure

	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

Procedure

	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.

	Command or Action	Purpose
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
```

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.
`Device# show vrf definition [brief | detail | interfaces] [vrf-name]`	Displays information about the defined VRF instances.

Command	Purpose
`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
```

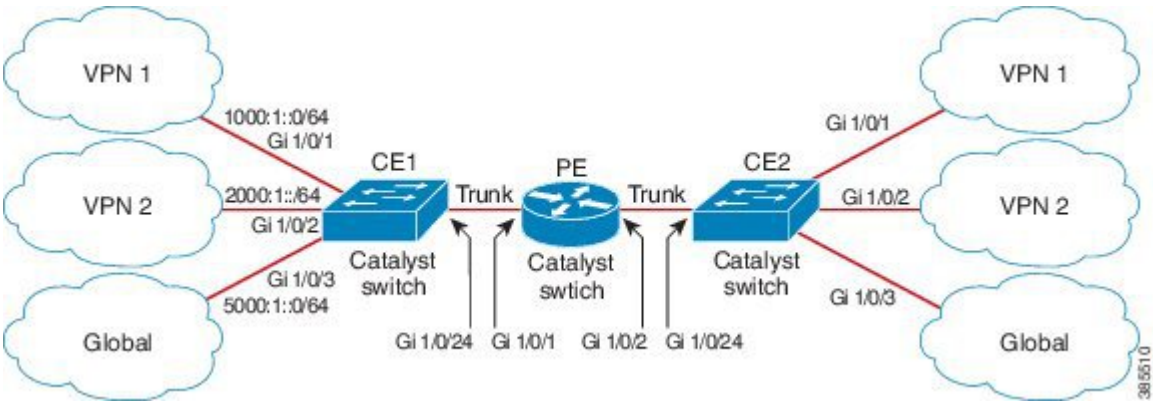
Configuration Examples for VRF-lite

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 10: 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 9200 Series Switches)*

Standards and RFCs

Standard/RFC	Title
RFC 6763	*DNS-Based Service Discovery*
Multicast DNS Internet-Draft	Multicast

Feature History for Multicast VRF-lite

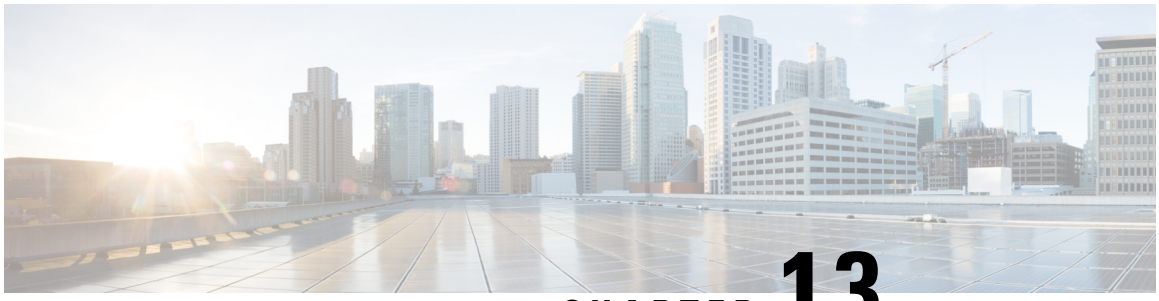
This table provides release and related information for the features explained in this module.

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 16.9.2	IPv6 Multicast support with VRF-Lite	IPv6 VRF-Lite allows a service provider to support two or more VPNs with overlapping IP addresses using one interface.

Release	Feature	Feature Information
Cisco IOS XE Cupertino 17.9.1	IPv6 Multicast support with VRF-Lite	This feature was implemented on C9200CX-12P-2X2G, C9200CX-8P-2X2G, and C9200CX-12T-2X2G models of the Cisco Catalyst 9200CX Series Switches, which were introduced in this release.

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



CHAPTER 13

Configuring Multi-VRF CE

- Information About Multi-VRF CE, on page 185
- How to Configure Multi-VRF CE, on page 189
- Monitoring Multi-VRF CE, on page 203
- Configuration Example: Multi-VRF CE, on page 203
- Feature History for Multi-VRF CE, on page 206

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 . 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:

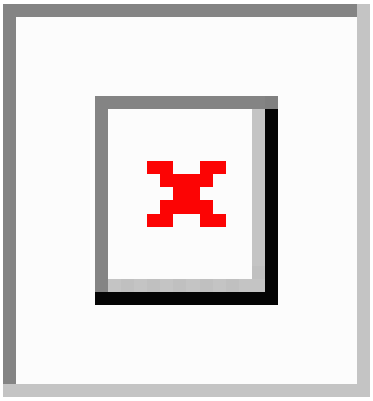
- Customer edge (CE) devices provide customers access to the service-provider network over a data link to one or more provider edge routers. The CE device advertises the site's local routes to the router and learns the remote VPN routes from it. A switch can be a CE.
- Provider routers or core routers 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 11: 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. 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.
- 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

This section provides guidelines for configuring multi-VRF CE:

- 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 number of VRFs supported on the Cisco Catalyst 9200 Series Switche models is as follows:

Switch Model	Number of VRFs supported
C9200L-24T-4G	1
C9200L-24P-4G	
C9200L-48T-4G	
C9200L-48P-4G	
C9200L-24T-4X	
C9200L-24P-4X	
C9200L-48T-4X	
C9200L-48P-4X	
C9200-24T	4
C9200-24P	
C9200-48T	
C9200-48P	
C9200-24PB	32
C9200-48PB	

- 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 14: 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.

	Command or Action	Purpose
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)
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`	(Optional) Saves your entries in the configuration file.

	Command or Action	Purpose
	`startup-config`	

Configuring Multicast VRFs

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 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:**	(Optional) Associates a route map with the VRF.

	Command or Action	Purpose
	`Device(config-vrf)#import map importmap1`	
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.
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`	(Optional) Saves your entries in the configuration file.

	Command or Action	Purpose
	`startup-config`	

Configuring a VPN Routing Session

Routing within the VPN can be configured with any supported routing protocol (RIP, OSPF, EIGRP, or) 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 isis subnets** **Example:** `Device(config-router)#redistribute isis 10 subnets`	Sets the switch to redistribute information from the ISIS network to the OSPF network.

	Command or Action	Purpose
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 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.

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

	Command or Action	Purpose
	`Device(config-if)exit`	
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.
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.

	Command or Action	Purpose
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.
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.
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:** `Device#configure terminal`	Enters global configuration mode.
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:**	Sends system logging messages to a logging facility.

	Command or Action	Purpose
	`Device(config)#logging facility user`	
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:**	Specifies the source IP address for FTP connections.

	Command or Action	Purpose
	`Device(config)#ip ftp source-interface gigabitethernet 1/0/2`	
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:** `Device(config)#end`	Returns to privileged EXEC mode.

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.

Monitoring 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

This section provides information on commands for monitoring multi-VRF CE:

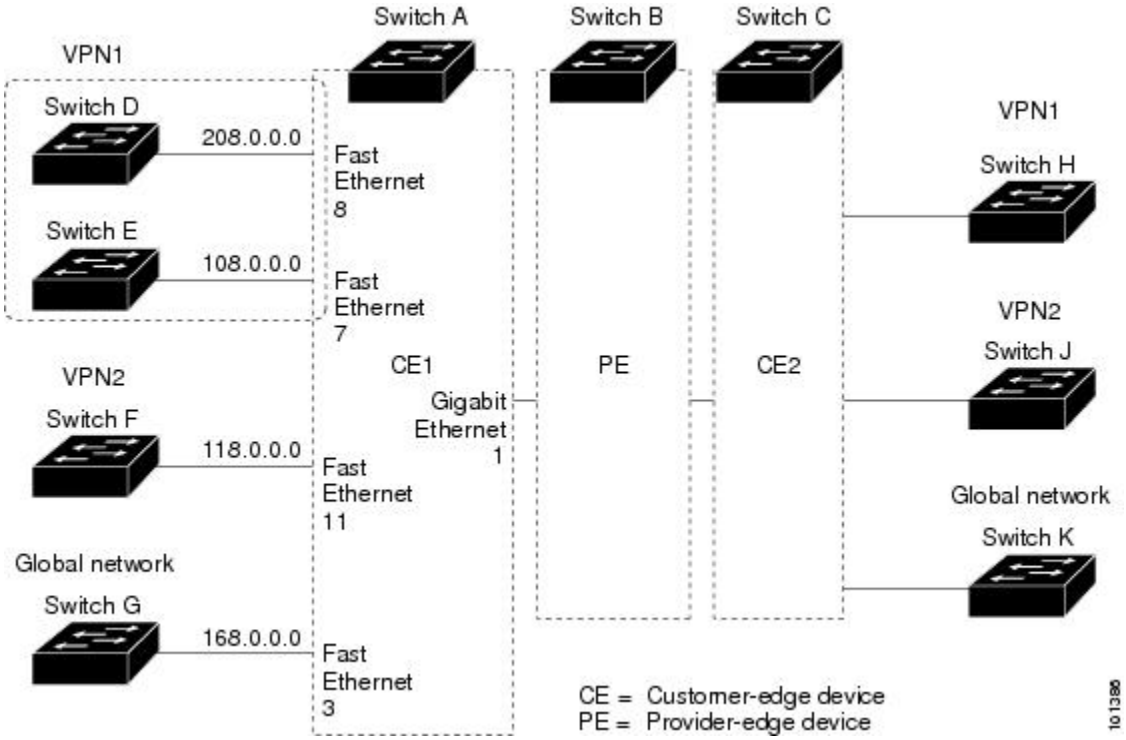
Table 15: Commands for Displaying Multi-VRF CE Information

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. 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 12: 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 isis 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 isis subnets
Device(config-router)#network 118.0.0.0 0.0.0.255 area 0
Device(config-router)#exit
```

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 History for Multi-VRF CE

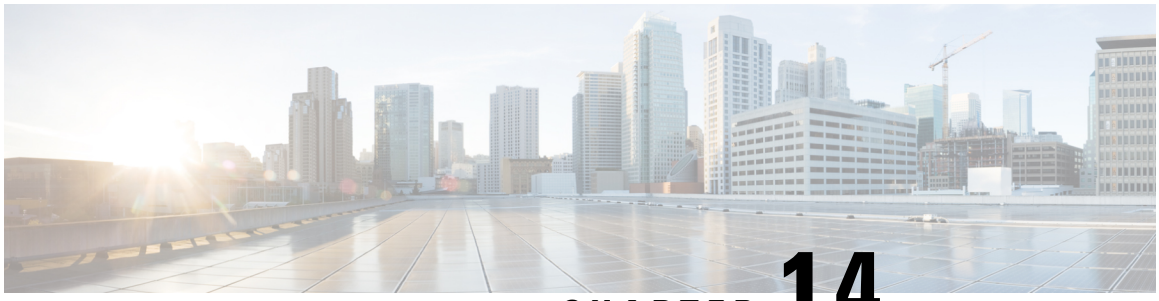
This table provides release and related information for the features explained in this module.

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 16.9.2	Multi-VRF CE	The switch supports multiple VPN routing/forwarding (multi-VRF) instances in customer edge (CE) devices (multi-VRF CE).
Cisco IOS XE Amsterdam 17.2.1	Additional VRF Support	On the C9200-24PB and C9200-48PB models of the Cisco Catalyst 9200 Series Switches, the switch supports 32 VRFs.

Release	Feature	Feature Information
Cisco IOS XE Cupertino 17.9.1	Multi-VRF CE	This feature was implemented on C9200CX-12P-2X2G, C9200CX-8P-2X2G, and C9200CX-12T-2X2G models of the Cisco Catalyst 9200CX Series Switches, which were introduced in this release.

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



CHAPTER 14

Configuring Unicast Reverse Path Forwarding

- Prerequisites for Unicast Reverse Path Forwarding, on page 209
- Restrictions for Unicast Reverse Path Forwarding, on page 209
- Information About Unicast Reverse Path Forwarding, on page 210
- How to Configure Unicast Reverse Path Forwarding, on page 215
- Monitoring and Maintaining Unicast Reverse Path Forwarding, on page 217
- Example: Configuring Unicast RPF, on page 218
- Feature History for Unicast Reverse Path Forwarding, on page 219

Prerequisites for Unicast Reverse Path Forwarding

- Unicast Reverse Path Forwarding (RPF) requires Cisco Express Forwarding to function properly on a device.
- Prior to configuring Unicast RPF, you must configure the following access control lists (ACLs):
  - Configure standard or extended ACL to mitigate the transmission of invalid IP addresses (by performing egress filtering). Configuring standard or extended ACLs permit only valid source addresses to leave your network and enter the Internet.
  - Configure standard or extended ACL entries to drop (deny) packets that have invalid source IP addresses (by performing ingress filtering). Invalid source IP addresses include the following types:
    - Broadcast addresses (including multicast addresses)
    - Loopback addresses
    - Private addresses (RFC 1918, *Address Allocation for Private Internets*)
    - Reserved addresses
    - Source addresses that fall outside the range of valid addresses that are associated with the protected network

Restrictions for Unicast Reverse Path Forwarding

The following basic restrictions apply to multihomed clients:

- Clients should not be multihomed on the same device because multihoming defeats the purpose of creating a redundant service for a client.
- Ensure that packets that flow up the link (out to the Internet) match the route advertised out of the link. Otherwise, Unicast RPF filters these packets as malformed packets.

Information About Unicast Reverse Path Forwarding

The Unicast Reverse Path Forwarding 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 Enabling IPv4 unicast RPF also enables IPv6 unicast RPF. This is applicable only for the .

Unicast RPF Operation

When Unicast RPF is enabled on an interface of a device, the device examines all packets received as input on that interface to ensure that the source address and source interface information appears in the routing table and matches the interface on which packets are received. This ability to “look backwards” is available only when Cisco Express Forwarding is enabled on a device because the lookup relies on the presence of a Forwarding Information Base (FIB). Cisco Express Forwarding generates a FIB as part of its operation.

Note Unicast RPF is an input function and is applied only on the input interface of a device at the upstream end of a connection.

Unicast RPF does a reverse lookup in the Cisco Express Forwarding table to check if any packet received at the interface of a device arrives on the best return path (or return route) to the source of the packet. If the packet was received from one of the best reverse path routes, the packet is forwarded as normal. No reverse path route on the interface from which the packet was received can mean that the source address was modified. If Unicast RPF cannot find a reverse path for the packet, the packet is dropped or forwarded, depending on whether an access control list (ACL) is specified by using the **ip verify unicast reverse-path** command in interface configuration mode.

Note With Unicast RPF, all equal-cost “best” return paths are considered valid. Unicast RPF supports multiple return paths, provided that each path is equal to the others in terms of the routing cost (such as number of hops, weights, and so on) and the route is available in the FIB. Unicast RPF also functions where Enhanced Interior Gateway Routing Protocol (EIGRP) variants are used.

Before forwarding a packet that is received at the interface on which Unicast RPF and ACLs have been configured, Unicast RPF does the following checks:

1. If input ACLs are configured on the inbound interface.
2. If the packet has arrived on the best return path to the source by doing a reverse lookup in the FIB table.
3. Does a lookup of the Cisco Express Forwarding table for packet forwarding.
4. Checks output ACLs on the outbound interface.
5. Forwards the packet.

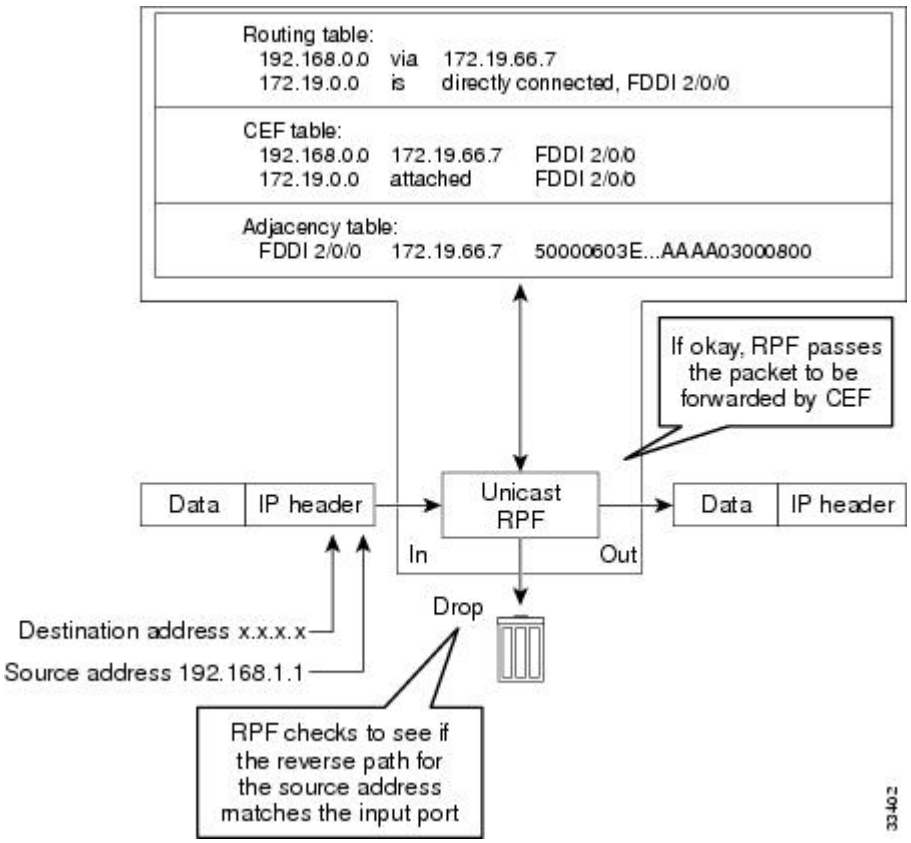
Per-Interface Statistics

Each time a packet is dropped or forwarded at an interface, that information is counted two ways: globally on the device and at each interface where you have applied Unicast RPF. Global statistics on dropped packets provide information about potential attacks on the network; however, these global statistics do not help to specify which interface is the source of the attack.

Per-interface statistics allow network administrators to track two types of information about malformed packets: Unicast RPF drops and Unicast RPF suppressed drops. Statistics on the number of packets that Unicast RPF drops help to identify the interface that is the entry point of the attack. The Unicast RPF drop count tracks the number of drops at the interface. The Unicast RPF suppressed drop count tracks the number of packets that failed the Unicast RPF check but were forwarded because of the permit permission set up in the ACL. Using the drop count and suppressed drop count statistics, a network administrator can takes steps to isolate the attack at a specific interface.

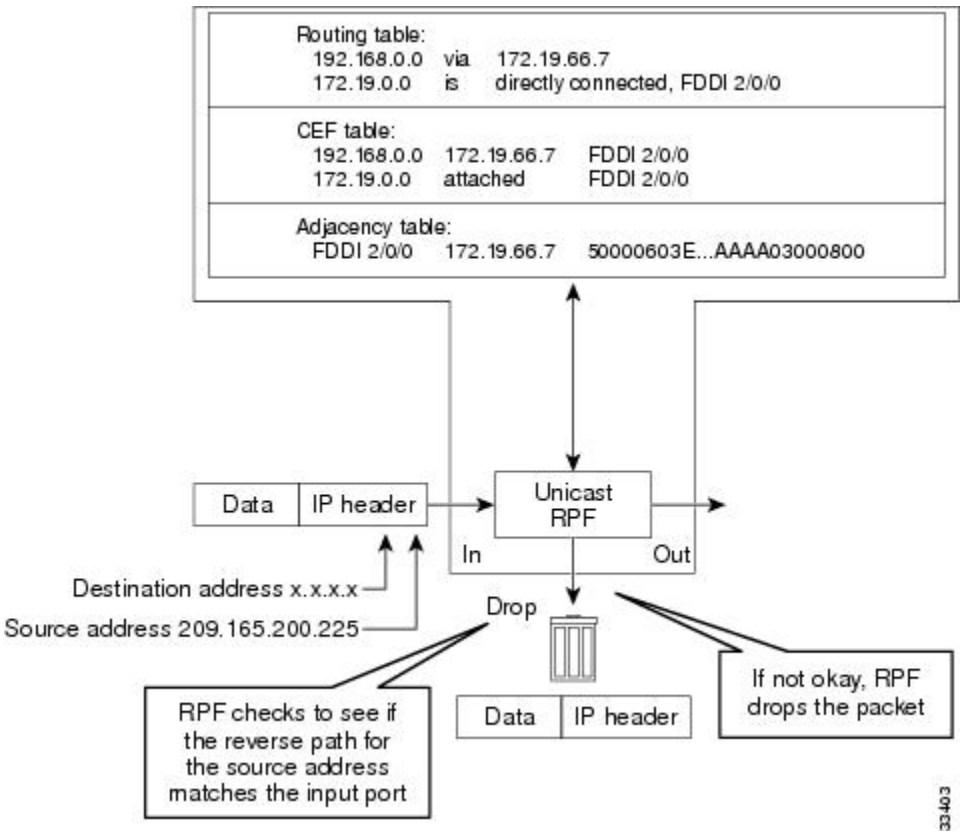
The figure below illustrates how Unicast RPF and CEF work together to validate IP source addresses by verifying packet return paths. In this example, a customer has sent a packet having a source address of 192.168.1.1 from interface FDDI 2/0/0. Unicast RPF checks the FIB to see if 192.168.1.1 has a path to FDDI 2/0/0. If there is a matching path, the packet is forwarded. If there is no matching path, the packet is dropped.

Figure 13: Unicast RPF Validating IP Source Addresses



The figure below illustrates how Unicast RPF drops packets that fail validation. In this example, a customer has sent a packet having a source address of 209.165.200.225, which is received at interface FDDI 2/0/0. Unicast RPF checks the FIB to see if 209.165.200.225 has a return path to FDDI 2/0/0. If there is a matching path, the packet is forwarded. In this case, there is no reverse entry in the routing table that routes the customer packet back to source address 209.165.200.225 on interface FDDI 2/0/0, and so the packet is dropped.

Figure 14: Unicast RPF Dropping Packets That Fail Verification



Implementation of Unicast Reverse Path Forwarding Notification

Unicast RPF is a security feature that verifies the validity of the source IP of an incoming packet. When a packet arrives at an interface and its source IP is unknown in the routing table or is a known bad source address, Unicast RPF drops the packet. IP verification of the source is done to prevent the DoS attacks by detecting problems with the incoming packets on an interface. However, deploying Unicast RPF without some automated monitoring capability is a challenge.

The CISCO-IP-URPF-MIB lets you specify a Unicast RPF drop-rate threshold on interfaces of a managed device that will send an SNMP notification when the threshold is exceeded. The MIB includes objects for specifying global and per-interface drop counts and drop rates and a method to generate SNMP traps when the drop rate exceeds a configurable per-interface threshold.

Although you can configure some parameters globally, you must configure the CISCO-IP-URPF-MIB on individual interfaces.

Security Policy and Unicast RPF

When determining how to deploy Unicast Reverse Path Forwarding (RPF), consider the following points:

- Apply Unicast RPF at the downstream interface, away from the larger portion of the network, preferably at the edges of your network. The further you apply Unicast RPF, the finer the granularity you have in mitigating address spoofing and in identifying sources of spoofed addresses. For example, applying Unicast RPF on an aggregation device helps to mitigate attacks from many downstream networks or

clients and is simple to administer, but Unicast RPF does not help in identifying the source of the attack. Applying Unicast RPF at the network access server helps to limit the scope of the attack and trace the source of the attack. However, deploying Unicast RPF across many sites adds to the administration cost of operating a network.

- When you deploy Unicast RPF on many entities on a network (for example, across the Internet, intranet, and extranet resources), you have better chances of mitigating large-scale network disruptions throughout the Internet community, and of tracing the source of an attack.
- Unicast RPF does not inspect IP packets that are encapsulated in tunnels, such as the generic routing encapsulation (GRE), Layer 2 Tunneling Protocol (L2TP), or Point-to-Point Tunneling Protocol (PPTP). Configure Unicast RPF on a home gateway so that Unicast RPF processes network traffic only after tunneling and encryption layers are stripped off from the packets.

Ingress and Egress Filtering Policy for Unicast RPF

Note Unicast RPF with access control lists (ACLs) is not supported on the

Unicast RPF can be more effective at mitigating spoofing attacks when combined with a policy of ingress and egress filtering by using ACLs.

Ingress filtering applies filters to traffic that is received at a network interface from either internal or external networks. With ingress filtering, packets that arrive from other networks or the Internet and that have a source address that matches a local network or private or broadcast addresses are dropped. For example, in ISP environments, ingress filtering can be applied to traffic that is received at a device from either a client (customer) or the Internet.

Egress filtering applies filters to the traffic that exits a network interface (the sending interface). By filtering packets on devices that connect your network to the Internet or to other networks, you can permit only packets with valid source IP addresses to leave your network.

For more information on network filtering, refer to RFC 2267, *Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing*.

Where to Use Unicast Reverse Path Forwarding

Unicast RPF can be used in any “single-homed” environment where there is essentially only one access point out of the network, which means that there is only one upstream connection to the network. Networks having one access point offer the best example of symmetric routing, which means that the interface where a packet enters the network is also the best return path to the source of the IP packet. Unicast RPF is best used at the network perimeter for Internet, intranet, or extranet environments, or in ISP environments for customer network terminations.

Routing Table Requirements

Unicast Reverse Path Forwarding (RPF) uses the routing information in Cisco Express Forwarding tables for routing traffic. The amount of routing information that must be available in Cisco Express Forwarding tables depends on the device where Unicast RPF is configured and the functions the device performs in the network. For example, in an ISP environment where a device is a leased-line aggregation device for customers, the

information about static routes that are redistributed into the Interior Gateway Protocol (IGP) or Internal Border Gateway Protocol (IBGP) (depending on which technique is used in the network) is required in the routing table. Because Unicast RPF is configured on customer interfaces, only minimal routing information is required. If a single-homed ISP configures Unicast RPF on the gateway to the Internet, the full Internet routing table information is required by Unicast RPF to help protect the ISP from external denial of service (DoS) attacks that use addresses that are not in the Internet routing table.

Where Not to Use Unicast Reverse Path Forwarding

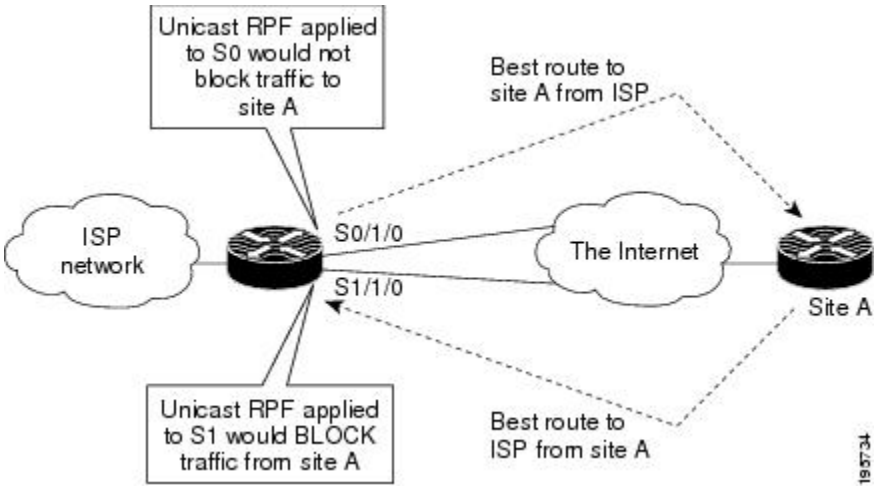
Do not use unicast RPF on interfaces that are internal to a network. Internal interfaces are likely to have routing asymmetry (see the figure below), which means that there can be multiple routes to the source of a packet. Unicast RPF is applied only where there is a natural or configured symmetry.

For example, devices at the edge of an ISP network are more likely to have symmetrical reverse paths than devices that are in the core of an ISP network. The best forwarding path to forward packets from devices that are at the core of an ISP network may not be the best forwarding path that is selected for packets that are returned to the device.

We recommend that you do not apply Unicast RPF where there is a chance of asymmetric routing, unless you configure access control lists (ACLs) to allow the device to accept incoming packets. ACLs permit the use of Unicast RPF when packets arrive through specific, less-optimal asymmetric input paths.

The figure below illustrates how Unicast RPF can block legitimate traffic in an asymmetric routing environment.

Figure 15: Unicast RPF Blocking Legitimate Traffic in an Asymmetric Routing Environment



Unicast Reverse Path Forwarding with BOOTP and DHCP

Unicast RPF allows packets with 0.0.0.0 as the source IP address and 255.255.255.255 as the destination IP address to pass through a network to enable Bootstrap Protocol (BOOTP) and DHCP functions to work properly when Unicast RPF is configured.

How to Configure Unicast Reverse Path Forwarding

The following section provide configuration information about unicast reverse path forwarding.

Configuring Unicast Reverse Path Forwarding

Before you begin

To use Unicast Reverse Path Forwarding, you must configure a device for Cisco Express Forwarding switching or distributed Cisco Express Forwarding switching. If Cisco Express Forwarding is not enabled globally on a device, Unicast RPF will not work on that device. If Cisco Express Forwarding is running on a device, individual interfaces on the device can be configured with other switching modes. Unicast RPF is an input-side function that is enabled on an interface or subinterface that supports any type of encapsulation, and Unicast RPF operates on IP packets that are received by the 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	**ip cef distributed** **Example:** `Device(config)# ip cef distributed`	Enables Cisco Express Forwarding or distributed Cisco Express Forwarding on a device.
Step 4	**interface** *slot/subslot/port* **Example:** `Device(config)# interface GigabitEthernet 0/0`	Selects the input interface on which you want to apply Unicast Reverse Path Forwarding and enters interface configuration mode. The interface that is configured is the receiving interface, which allows Unicast RPF to verify the best return path before forwarding a packet to the next destination.
Step 5	**ip verify unicast reverse-path** *list* **Example:** `Device(config-if)# ip verify unicast reverse-path 197`	Enables Unicast RPF on the interface. • Use the *list* argument to identify an access list. If the access list denies network access, spoofed packets are dropped at the interface. If the access list permits network access, spoofed packets are forwarded to the destination address. Forwarded packets are counted in the interface statistics. If the access list includes the logging option, information about the spoofed packets is logged to the log server. • Repeat this step for each access list that you want specify

	Command or Action	Purpose
Step 6	**exit** **Example:** `Device(config-if)# exit`	Exits interface configuration mode.

Troubleshooting Tips

HSRP Failure

The failure to disable Unicast RPF before disabling Cisco Express Forwarding can cause a Hot Standby Router Protocol (HSRP) failure. If you want to disable Cisco Express Forwarding on a device, you must first disable Unicast RPF.

Monitoring and Maintaining Unicast Reverse Path Forwarding

This section describes commands used to monitor and maintain unicast RPF.

Command	Purpose
`Device# show ip traffic`	Displays global router statistics about Unicast RPF drops and suppressed drops.
`Device# show ip interface type`	Displays per-interface statistics about Unicast RPF drops and suppressed drops.
`Device# show access-lists`	Displays the number of matches to a specific ACL.
`Device(config-if)# no ip verify unicast reverse-path list`	Disables Unicast RPF at the interface. Use the *list* option to disable Unicast RPF for a specific ACL at the interface.

Caution To disable CEF, you must first disable Unicast RPF. Failure to disable Unicast RPF before disabling CEF can cause HSRP failure. If you want to disable CEF on the router, you must first disable Unicast RPF.

Unicast RPF counts the number of packets dropped or suppressed because of malformed or forged source addresses. Unicast RPF counts dropped or forwarded packets that include the following global and per-interface information:

- Global Unicast RPF drops
- Per-interface Unicast RPF drops
- Per-interface Unicast RPF suppressed drops

The **show ip traffic** command shows the total number (global count) of dropped or suppressed packets for all interfaces on the router. The Unicast RPF drop count is included in the IP statistics section.

```
Device# show ip traffic

IP statistics:
  Rcvd:  1471590 total, 887368 local destination
         0 format errors, 0 checksum errors, 301274 bad hop count
         0 unknown protocol, 0 not a gateway
         0 security failures, 0 bad options, 0 with options
  Opts:  0 end, 0 nop, 0 basic security, 0 loose source route
         0 timestamp, 0 extended security, 0 record route
         0 stream ID, 0 strict source route, 0 alert, 0 other
  Frags: 0 reassembled, 0 timeouts, 0 couldn't reassemble
         0 fragmented, 0 couldn't fragment
  Bcast: 205233 received, 0 sent
  Mcast: 463292 received, 462118 sent
  Sent:  990158 generated, 282938 forwarded
  ! The second line below ("0 unicast RPF") displays Unicast RPF packet dropping
information.
  Drop:  3 encapsulation failed, 0 unresolved, 0 no adjacency
         0 no route, 0 unicast RPF, 0 forced drop
```

A nonzero value for the count of dropped or suppressed packets can mean one of two things:

- Unicast RPF is dropping or suppressing packets that have a bad source address (normal operation).
- Unicast RPF is dropping or suppressing legitimate packets because the route is misconfigured to use Unicast RPF in environments where asymmetric routing exists; that is, where multiple paths can exist as the best return path for a source address.

The **show ip interface** command shows the total of dropped or suppressed packets at a specific interface. If Unicast RPF is configured to use a specific ACL, that ACL information is displayed along with the drop statistics.

```
Device> show ip interface ethernet0/1/1

   Unicast RPF ACL 197
   1 unicast RPF drop
   1 unicast RPF suppressed drop
```

The **show access-lists** command displays the number of matches found for a specific entry in a specific access list.

```
Device> show access-lists

Extended IP access list 197
     deny ip 192.168.201.0 0.0.0.63 any log-input (1 match)
     permit ip 192.168.201.64 0.0.0.63 any log-input (1 match)
     deny ip 192.168.201.128 0.0.0.63 any log-input
     permit ip 192.168.201.192 0.0.0.63 any log-input
```

Example: Configuring Unicast RPF

```
Device# configure terminal
Device(config)# ip cef distributed
Device(config)# interface GigabitEthernet 1/0/2
Device(config-if)# description Connection to Upstream ISP
Device(config-if)# ip address 209.165.200.225 255.255.255.252
Device(config-if)# no ip redirects
Device(config-if)# no ip directed-broadcast
```

```
Device(config-if)# no ip proxy-arp
Device(config-if)# ip verify unicast reverse-path
```

```
Device# configure terminal
Device(config)# ip cef distributed
Device(config)# interface GigabitEthernet 1/0/2
Device(config-if)# description Connection to Upstream ISP
Device(config-if)# ip address 209.165.200.225 255.255.255.252
Device(config-if)# no ip redirects
Device(config-if)# no ip directed-broadcast
Device(config-if)# no ip proxy-arp
Device(config-if)# ip verify unicast source reachable-via rx
```

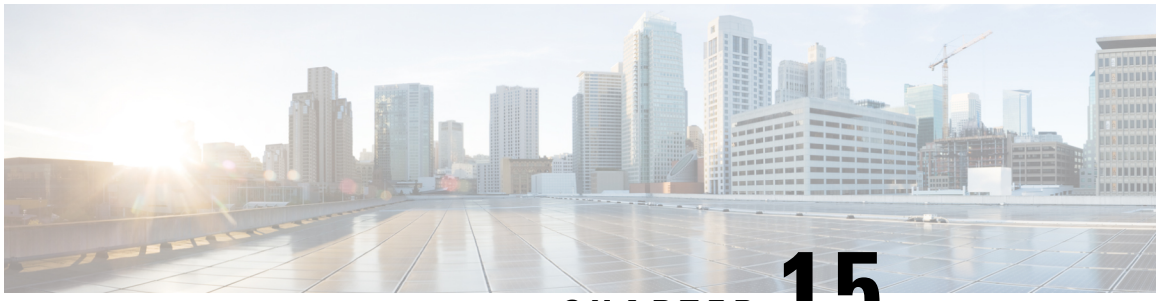
Feature History for Unicast Reverse Path Forwarding

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 16.9.2	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.
Cisco IOS XE Cupertino 17.9.1	Unicast Reverse Path Forwarding	This feature was implemented on C9200CX-12P-2X2G, C9200CX-8P-2X2G, and C9200CX-12T-2X2G models of the Cisco Catalyst 9200CX Series Switches, which were introduced in this release.

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



CHAPTER 15

Protocol-Independent Features

- Distributed Cisco Express Forwarding and Load-Balancing Scheme for CEF Traffic , on page 221
- Number of Equal-Cost Routing Paths, on page 226
- Static Unicast Routes, on page 227
- Default Routes and Networks, on page 229
- Route Maps to Redistribute Routing Information, on page 231
- Policy-Based Routing, on page 237
- Filtering Routing Information, on page 241
- Managing Authentication Keys, on page 245
- Feature History for Protocol-Independent Features, on page 247

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.

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).

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.

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:**	(Optional) Displays CEF-related interface information on a non-stacking switch.

	Command or Action	Purpose
	`Device# show cef linecard detail`	
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:

Procedure

	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	**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:

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device# enable`	Enters global configuration mode.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
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.

	Command or Action	Purpose
		• 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 16: 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
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**	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 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.

	Command or Action	Purpose
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:** `Device(config-route-map)# match ip next-hop 8 45`	Matches a next-hop router address passed by one of the access lists specified (numbered from 1 to 199).
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.

	Command or Action	Purpose
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.
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).

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

	Command or Action	Purpose
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 Policy-based Routing

- 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).
- 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

- 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.
- When configuring match criteria in a route map, follow these guidelines:
  - Do not match ACLs that permit packets destined for a local address.

- VRF and PBR are mutually exclusive on a switch interface. You cannot enable VRF when PBR is enabled on an interface. The reverse is also true, you cannot enable PBR when VRF is enabled on an interface.
- Web Cache Communication Protocol (WCCP) and PBR are mutually exclusive on a switch interface. You cannot enable WCCP when PBR is enabled on an interface. 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.

	Command or Action	Purpose
		• (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:** `Device(config-route-map)# match length 64 1500`	Matches the length of the packet.
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.

	Command or Action	Purpose
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.
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

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:** `Device(config-router)# passive-interface gigabitethernet 1/0/1`	Suppresses sending routing updates through the specified Layer 3 interface.
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.

	Command or Action	Purpose
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.
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:**	Suppresses processing in routes listed in updates.

	Command or Action	Purpose
	`Device(config-router)# distribute-list 125 in`	
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*]	Defines an administrative distance.

	Command or Action	Purpose
	Example: `Device(config-router)# distance 50 10.1.5.1`	*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*} **Example:** `Device(config-keychain)# 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 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.

	Command or Action	Purpose
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.

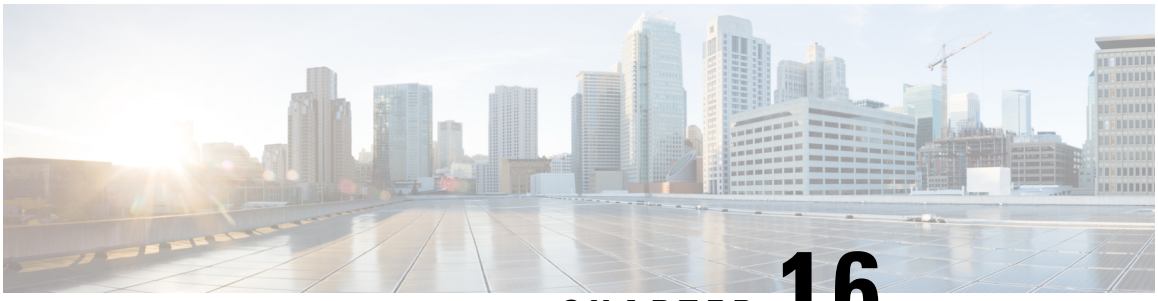
Feature History for Protocol-Independent Features

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 16.9.2	Protocol-Independent Features-Distributed Cisco Express Forwarding	Cisco Express Forwarding (CEF) is a Layer 3 IP switching technology used to optimize network performance.
	Protocol-Independent Features-Policy-Based Routing	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.
	Protocol-Independent Features-Managing Authentication Keys	Key management is a method of controlling authentication keys used by routing protocols. Authentication keys are available for EIGRP and RIP Version 2.
Cisco IOS XE Cupertino 17.9.1	Protocol-Independent Features-Distributed Cisco Express Forwarding	This feature was implemented on C9200CX-12P-2X2G, C9200CX-8P-2X2G, and C9200CX-12T-2X2G models of the Cisco Catalyst 9200CX Series Switches, which were introduced in this release.

Use the Cisco Feature Navigator to find information about platform and software image support.



CHAPTER 16

Configuring Generic Routing Encapsulation(GRE) Tunnel IP Source and Destination VRF Membership

- Restrictions for GRE Tunnel IP Source and Destination VRF Membership, on page 249
- Information About GRE Tunnel IP Source and Destination VRF Membership, on page 250
- How to Configure GRE Tunnel IP Source and Destination VRF Membership, on page 250
- Configuration Example for GRE Tunnel IP Source and Destination VRF Membership, on page 251
- Additional References, on page 252
- Feature History for Generic Routing Encapsulation Tunnel IP Source and Destination VRF Membership, on page 252

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.
- 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:

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 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.
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.

	Command or Action	Purpose
		• *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 17: 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

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 16.9.2	Generic Routing Encapsulation(GRE) Tunnel IP Source and Destination VRF Membership	GRE Tunnel IP Source and Destination VRF Membership feature allows you to configure the source and destination of a tunnel to belong to any VPN VRF table.
Cisco IOS XE Cupertino 17.9.1	Generic Routing Encapsulation(GRE) Tunnel IP Source and Destination VRF Membership	This feature was implemented on C9200CX-12P-2X2G, C9200CX-8P-2X2G, and C9200CX-12T-2X2G models of the Cisco Catalyst 9200CX Series Switches, which were introduced in this release.

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