MPLS Layer 3 VPN Configuration Guide for Cisco ASR 9000 Series Routers, IOS XR 24.1.x, 24.2.x, 24.3.x

First Published: 2024-03-14

Last Modified: 2024-09-04

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

CONTENTS

Preface

From Release 6.1.2 onwards, Cisco introduces support for the 64-bit Linux-based IOS XR operating system. Extensive feature parity is maintained between the 32-bit and 64-bit environments. Unless explicitly marked otherwise, the contents of this document are applicable for both the environments. For more details on Cisco IOS XR 64 bit, refer to the Release Notes for Cisco ASR 9000 Series Routers, Release 6.1.2 document.

This guide describes the Cisco ASR 9000 Series Router configurations. The preface for the *L2VPN and Ethernet Services Configuration Guide for Cisco ASR 9000 Series Routers* contains these sections:

Changes to This Document

The following table lists the technical changes made to this document since it was first published.

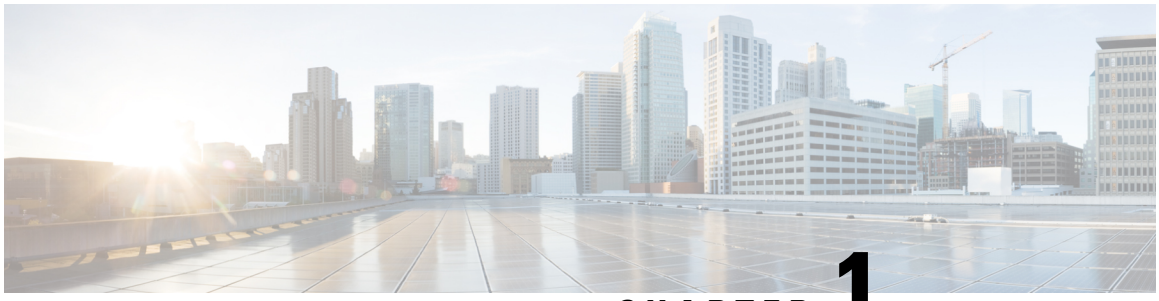
Date	Summary
September 2024	Republished for Release 24.3.1.
June 2024	Republished for Release 24.2.1.
March 2024	Initial release of this document.

Obtaining Documentation and Submitting a Service Request

For information on obtaining documentation, submitting a service request, and gathering additional information, see the monthly What's New in Cisco Product Documentation, which also lists all new and revised Cisco technical documentation, at:

http://www.cisco.com/en/US/docs/general/whatsnew/whatsnew.html

Subscribe to the What's New in Cisco Product Documentation as a Really Simple Syndication (RSS) feed and set content to be delivered directly to your desktop using a reader application. The RSS feeds are a free service and Cisco currently supports RSS version 2.0.



CHAPTER 1

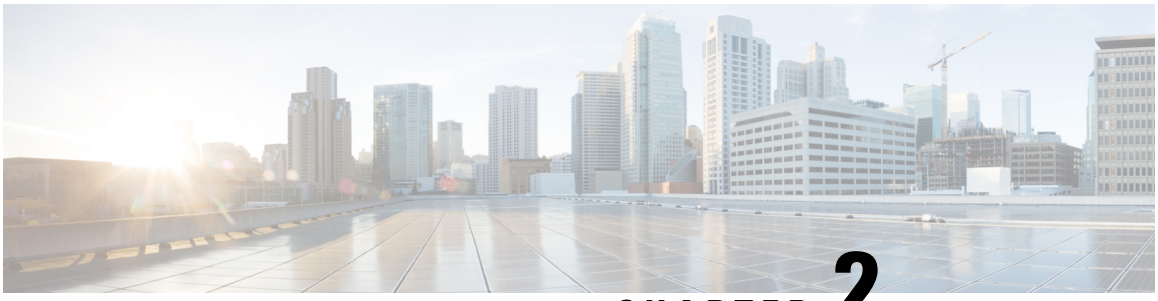
New and Changed L3VPN Features

New and Changed L3VPN Features

Table 1: L3VPN Features Added or Modified in IOS XR Release 24.x.x

Feature	Description	Changed in Release	Where Documented
None	No new features introduced	Not applicable	Not applicable

New and Changed L3VPN Features



CHAPTER 2

YANG Data Models for L3VPN Features

This chapter provides information about the YANG data models for L3VPN Features.

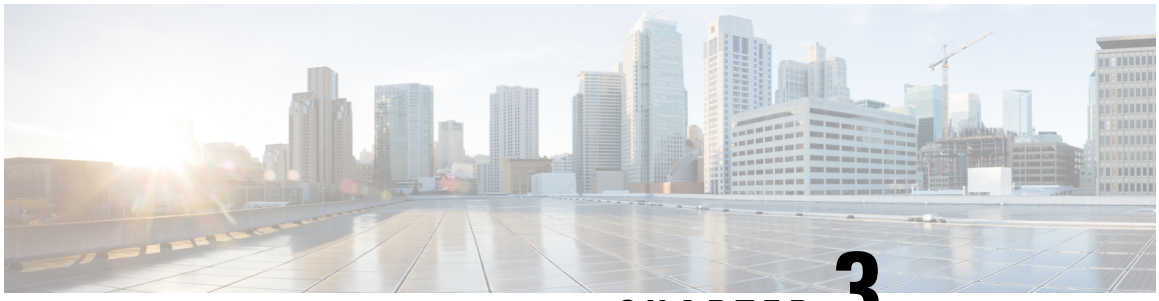
Using YANG Data Models

Cisco IOS XR supports a programmatic way of configuring and collecting operational data of a network device using YANG data models. Although configurations using CLIs are easier and human-readable, automating the configuration using model-driven programmability results in scalability.

The data models are available in the release image, and are also published in the Github repository. Navigate to the release folder of interest to view the list of supported data models and their definitions. Each data model defines a complete and cohesive model, or augments an existing data model with additional XPaths. To view a comprehensive list of the data models supported in a release, navigate to the **Available-Content.md** file in the repository.

You can also view the data model definitions using the YANG Data Models Navigator tool. This GUI-based and easy-to-use tool helps you explore the nuances of the data model and view the dependencies between various containers in the model. You can view the list of models supported across Cisco IOS XR releases and platforms, locate a specific model, view the containers and their respective lists, leaves, and leaf lists presented visually in a tree structure. This visual tree form helps you get insights into nodes that can help you automate your network.

To get started with using the data models, see the *Programmability Configuration Guide*.



CHAPTER 3

Implementing MPLS Layer 3 VPNs

A Multiprotocol Label Switching (MPLS) Layer 3 Virtual Private Network (VPN) consists of a set of sites that are interconnected by means of an MPLS provider core network. At each customer site, one or more customer edge (CE) routers attach to one or more provider edge (PE) routers.

This module provides the conceptual and configuration information for MPLS Layer 3 VPNs on Cisco IOS XR software.

Note You must acquire an evaluation or permanent license in order to use MPLS Layer 3 VPN functionality. However, if you are upgrading from a previous version of the software, MPLS Layer 3 VPN functionality will continue to work using an implicit license for 90 days (during which time, you can purchase a permanent license). For more information about licenses, see the *Software Entitlement on the Cisco ASR 9000 Series Router* module in the *System Management Configuration Guide for Cisco ASR 9000 Series Routers*.

Prerequisites for Implementing MPLS L3VPN

The following prerequisites are required to configure MPLS Layer 3 VPN:

- To perform these configuration tasks, your Cisco IOS XR software system administrator must assign you to a user group associated with a task group that includes the corresponding command task IDs. All command task IDs are listed in individual command references and in the *Cisco IOS XR Task ID Reference Guide*.
- If you suspect user group assignment is preventing you from using a command, contact your AAA administrator for assistance.

- You must be in a user group associated with a task group that includes the proper task IDs for:
  - BGP commands
  - MPLS commands (generally)
  - MPLS Layer 3 VPN commands
- To configure MPLS Layer 3 VPNs, routers must support MPLS forwarding and Forwarding Information Base (FIB).

The following prerequisites are required for configuring MPLS VPN Inter-AS with autonomous system boundary routers (ASBRs) exchanging VPN-IPV4 addresses or IPv4 routes and MPLS labels:

- Before configuring external Border Gateway Protocol (eBGP) routing between autonomous systems or subautonomous systems in an MPLS VPN, ensure that all MPLS VPN routing instances and sessions are properly configured (see the How to Implement MPLS Layer 3 VPNs, for procedures)
- These following tasks must be performed:
  - Define VPN routing instances
  - Configure BGP routing sessions in the MPLS core
  - Configure PE-to-PE routing sessions in the MPLS core
  - Configure BGP PE-to-CE routing sessions
  - Configure a VPN-IPv4 eBGP session between directly connected ASBRs

MPLS L3VPN Restrictions

The following restrictions apply when configuring MPLS VPN Inter-AS with ASBRs exchanging IPv4 routes and MPLS labels:

- For networks configured with eBGP multihop, a label switched path (LSP) must be configured between non adjacent routers.
- Inter-AS supports IPv4 routes only. IPv6 is not supported.

Note The physical interfaces that connect the BGP speakers must support FIB and MPLS.

Information About MPLS Layer 3 VPNs

To implement MPLS Layer 3 VPNs, you need to understand the following concepts:

MPLS L3VPN Overview

Before defining an MPLS VPN, VPN in general must be defined. A VPN is:

- An IP-based network delivering private network services over a public infrastructure
- A set of sites that are allowed to communicate with each other privately over the Internet or other public or private networks

Conventional VPNs are created by configuring a full mesh of tunnels or permanent virtual circuits (PVCs) to all sites in a VPN. This type of VPN is not easy to maintain or expand, as adding a new site requires changing each edge device in the VPN.

MPLS-based VPNs are created in Layer 3 and are based on the peer model. The peer model enables the service provider and the customer to exchange Layer 3 routing information. The service provider relays the data between the customer sites without customer involvement.

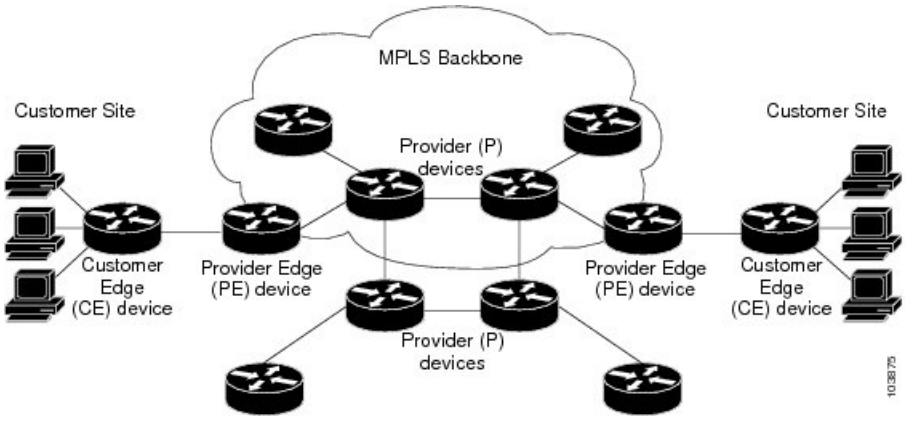
MPLS VPNs are easier to manage and expand than conventional VPNs. When a new site is added to an MPLS VPN, only the edge router of the service provider that provides services to the customer site needs to be updated.

The components of the MPLS VPN are described as follows:

- Provider (P) router—Router in the core of the provider network. P routers run MPLS switching and do not attach VPN labels to routed packets. VPN labels are used to direct data packets to the correct private network or customer edge router.
- PE router—Router that attaches the VPN label to incoming packets based on the interface or subinterface on which they are received, and also attaches the MPLS core labels. A PE router attaches directly to a CE router.
- Customer (C) router—Router in the Internet service provider (ISP) or enterprise network.
- Customer edge (CE) router—Edge router on the network of the ISP that connects to the PE router on the network. A CE router must interface with a PE router.

This following figures shows a basic MPLS VPN topology.

Figure 1: Basic MPLS VPN Topology



MPLS L3VPN Benefits

MPLS L3VPN provides the following benefits:

- Service providers can deploy scalable VPNs and deliver value-added services.

- Connectionless service guarantees that no prior action is necessary to establish communication between hosts.
- Centralized Service: Building VPNs in Layer 3 permits delivery of targeted services to a group of users represented by a VPN.
- Scalability: Create scalable VPNs using connection-oriented, point-to-point overlays, Frame Relay, or ATM virtual connections.
- Security: Security is provided at the edge of a provider network (ensuring that packets received from a customer are placed on the correct VPN) and in the backbone.
- Integrated Quality of Service (QoS) support: QoS provides the ability to address predictable performance and policy implementation and support for multiple levels of service in an MPLS VPN.
- Straightforward Migration: Service providers can deploy VPN services using a straightforward migration path.
- Migration for the end customer is simplified. There is no requirement to support MPLS on the CE router and no modifications are required for a customer intranet.

How MPLS L3VPN Works

MPLS VPN functionality is enabled at the edge of an MPLS network. The PE router performs the following tasks:

- Exchanges routing updates with the CE router
- Translates the CE routing information into VPN version 4 (VPNv4) routes.
- Exchanges VPNv4 and VPNv6 routes with other PE routers through the Multiprotocol Border Gateway Protocol (MP-BGP)

Virtual Routing and Forwarding Tables

Each VPN is associated with one or more VPN routing and forwarding (VRF) instances. A VRF defines the VPN membership of a customer site attached to a PE router. A VRF consists of the following components:

- An IP version 4 (IPv4) unicast routing table
- A derived FIB table
- A set of interfaces that use the forwarding table
- A set of rules and routing protocol parameters that control the information that is included in the routing table

These components are collectively called a VRF instance.

A one-to-one relationship does not necessarily exist between customer sites and VPNs. A site can be a member of multiple VPNs. However, a site can associate with only one VRF. A VRF contains all the routes available to the site from the VPNs of which it is a member.

Packet forwarding information is stored in the IP routing table and the FIB table for each VRF. A separate set of routing and FIB tables is maintained for each VRF. These tables prevent information from being

forwarded outside a VPN and also prevent packets that are outside a VPN from being forwarded to a router within the VPN.

VPN Routing Information: Distribution

The distribution of VPN routing information is controlled through the use of VPN route target communities, implemented by BGP extended communities. VPN routing information is distributed as follows:

- When a VPN route that is learned from a CE router is injected into a BGP, a list of VPN route target extended community attributes is associated with it. Typically, the list of route target community extended values is set from an export list of route targets associated with the VRF from which the route was learned.
- An import list of route target extended communities is associated with each VRF. The import list defines route target extended community attributes that a route must have for the route to be imported into the VRF. For example, if the import list for a particular VRF includes route target extended communities A, B, and C, then any VPN route that carries any of those route target extended communities—A, B, or C—is imported into the VRF.

BGP Distribution of VPN Routing Information

A PE router can learn an IP prefix from the following sources:

- A CE router by static configuration
- An eBGP session with the CE router
- A Routing Information Protocol (RIP) exchange with the CE router
- Open Shortest Path First (OSPF), Enhanced Interior Gateway Routing Protocol (EIGRP), and RIP as Interior Gateway Protocols (IGPs)

The IP prefix is a member of the IPv4 address family. After the PE router learns the IP prefix, the PE converts it into the VPN-IPv4 prefix by combining it with a 64-bit route distinguisher. The generated prefix is a member of the VPN-IPv4 address family. It uniquely identifies the customer address, even if the customer site is using globally nonunique (unregistered private) IP addresses. The route distinguisher used to generate the VPN-IPv4 prefix is specified by the **rd** command associated with the VRF on the PE router.

BGP distributes reachability information for VPN-IPv4 prefixes for each VPN. BGP communication takes place at two levels:

- Within the IP domain, known as an autonomous system.
- Between autonomous systems.

PE to PE or PE to route reflector (RR) sessions are iBGP sessions, and PE to CE sessions are eBGP sessions. PE to CE eBGP sessions can be directly or indirectly connected (eBGP multihop).

BGP propagates reachability information for VPN-IPv4 prefixes among PE routers by the BGP protocol extensions (see RFC 2283, Multiprotocol Extensions for BGP-4), which define support for address families other than IPv4. Using the extensions ensures that the routes for a given VPN are learned only by other members of that VPN, enabling members of the VPN to communicate with each other.

MPLS Forwarding

Based on routing information stored in the VRF IP routing table and the VRF FIB table, packets are forwarded to their destination using MPLS.

A PE router binds a label to each customer prefix learned from a CE router and includes the label in the network reachability information for the prefix that it advertises to other PE routers. When a PE router forwards a packet received from a CE router across the provider network, it labels the packet with the label learned from the destination PE router. When the destination PE router receives the labeled packet, it pops the label and uses it to direct the packet to the correct CE router. Label forwarding across the provider backbone is based on either dynamic label switching or traffic engineered paths. A customer data packet carries two levels of labels when traversing the backbone:

- The top label directs the packet to the correct PE router.
- The second label indicates how that PE router should forward the packet to the CE router.

More labels can be stacked if other features are enabled. For example, if traffic engineering (TE) tunnels with fast reroute (FRR) are enabled, the total number of labels imposed in the PE is four (Layer 3 VPN, Label Distribution Protocol (LDP), TE, and FRR).

Automatic Route Distinguisher Assignment

To take advantage of iBGP load balancing, every network VRF must be assigned a unique route distinguisher. VRF requires a route distinguisher for BGP to distinguish between potentially identical prefixes received from different VPNs.

With thousands of routers in a network each supporting multiple VRFs, configuration and management of route distinguishers across the network can present a problem. Cisco IOS XR software simplifies this process by assigning unique route distinguisher to VRFs using the **rd auto** command.

To assign a unique route distinguisher for each router, you must ensure that each router has a unique BGP router-id. If so, the **rd auto** command assigns a Type 1 route distinguisher to the VRF using the following format: *ip-address:number*. The IP address is specified by the BGP router-id statement and the number (which is derived as an unused index in the 0 to 65535 range) is unique across theVRFs.

Finally, route distinguisher values are checkpointed so that route distinguisher assignment to VRF is persistent across failover or process restart. If an route distinguisher is explicitely configured for a VRF, this value is not overridden by the autoroute distinguisher.

MPLS L3VPN Major Components

An MPLS-based VPN network has three major components:

- VPN route target communities—A VPN route target community is a list of all members of a VPN community. VPN route targets need to be configured for each VPN community member.
- Multiprotocol BGP (MP-BGP) peering of the VPN community PE routers—MP-BGP propagates VRF reachability information to all members of a VPN community. MP-BGP peering needs to be configured in all PE routers within a VPN community.
- MPLS forwarding—MPLS transports all traffic between all VPN community members across a VPN service-provider network.

A one-to-one relationship does not necessarily exist between customer sites and VPNs. A given site can be a member of multiple VPNs. However, a site can associate with only one VRF. A customer-site VRF contains all the routes available to the site from the VPNs of which it is a member

Inter-AS Support for L3VPN

This section contains the following topics:

Inter-AS Support: Overview

An autonomous system (AS) is a single network or group of networks that is controlled by a common system administration group and uses a single, clearly defined routing protocol.

As VPNs grow, their requirements expand. In some cases, VPNs need to reside on different autonomous systems in different geographic areas. In addition, some VPNs need to extend across multiple service providers (overlapping VPNs). Regardless of the complexity and location of the VPNs, the connection between autonomous systems must be seamless.

An MPLS VPN Inter-AS provides the following benefits:

- Allows a VPN to cross more than one service provider backbone.

  Service providers, running separate autonomous systems, can jointly offer MPLS VPN services to the same end customer. A VPN can begin at one customer site and traverse different VPN service provider backbones before arriving at another site of the same customer. Previously, MPLS VPN could traverse only a single BGP autonomous system service provider backbone. This feature lets multiple autonomous systems form a continuous, seamless network between customer sites of a service provider.

- Allows a VPN to exist in different areas.

  A service provider can create a VPN in different geographic areas. Having all VPN traffic flow through one point (between the areas) allows for better rate control of network traffic between the areas.

- Allows confederations to optimize iBGP meshing.

  Internal Border Gateway Protocol (iBGP) meshing in an autonomous system is more organized and manageable. You can divide an autonomous system into multiple, separate subautonomous systems and then classify them into a single confederation. This capability lets a service provider offer MPLS VPNs across the confederation, as it supports the exchange of labeled VPN-IPv4 Network Layer Reachability Information (NLRI) between the subautonomous systems that form the confederation.

Inter-AS and ASBRs

Separate autonomous systems from different service providers can communicate by exchanging IPv4 NLRI and IPv6 in the form of VPN-IPv4 addresses. The ASBRs use eBGP to exchange that information. Then an Interior Gateway Protocol (IGP) distributes the network layer information for VPN-IPV4 prefixes throughout each VPN and each autonomous system. The following protocols are used for sharing routing information:

- Within an autonomous system, routing information is shared using an IGP.
- Between autonomous systems, routing information is shared using an eBGP. An eBGP lets service providers set up an interdomain routing system that guarantees the loop-free exchange of routing information between separate autonomous systems.

  The primary function of an eBGP is to exchange network reachability information between autonomous systems, including information about the list of autonomous system routes. The autonomous systems

use EBGP border edge routers to distribute the routes, which include label switching information. Each border edge router rewrites the next-hop and MPLS labels.

Inter-AS configurations supported in an MPLS VPN can include:

- Interprovider VPN—MPLS VPNs that include two or more autonomous systems, connected by separate border edge routers. The autonomous systems exchange routes using eBGP. No IGP or routing information is exchanged between the autonomous systems.
- BGP Confederations—MPLS VPNs that divide a single autonomous system into multiple subautonomous systems and classify them as a single, designated confederation. The network recognizes the confederation as a single autonomous system. The peers in the different autonomous systems communicate over eBGP sessions; however, they can exchange route information as if they were iBGP peers.

Confederations

A confederation is multiple subautonomous systems grouped together. A confederation reduces the total number of peer devices in an autonomous system. A confederation divides an autonomous system into subautonomous systems and assigns a confederation identifier to the autonomous systems. A VPN can span service providers running in separate autonomous systems or multiple subautonomous systems that form a confederation.

In a confederation, each subautonomous system is fully meshed with other subautonomous systems. The subautonomous systems communicate using an IGP, such as Open Shortest Path First (OSPF) or Intermediate System-to-Intermediate System (IS-IS). Each subautonomous system also has an eBGP connection to the other subautonomous systems. The confederation eBGP (CEBGP) border edge routers forward next-hop-self addresses between the specified subautonomous systems. The next-hop-self address forces the BGP to use a specified address as the next hop rather than letting the protocol choose the next hop.

You can configure a confederation with separate subautonomous systems two ways:

- Configure a router to forward next-hop-self addresses between only the CEBGP border edge routers (both directions). The subautonomous systems (iBGP peers) at the subautonomous system border do not forward the next-hop-self address. Each subautonomous system runs as a single IGP domain. However, the CEBGP border edge router addresses are known in the IGP domains.
- Configure a router to forward next-hop-self addresses between the CEBGP border edge routers (both directions) and within the iBGP peers at the subautonomous system border. Each subautonomous system runs as a single IGP domain but also forwards next-hop-self addresses between the PE routers in the domain. The CEBGP border edge router addresses are known in the IGP domains.

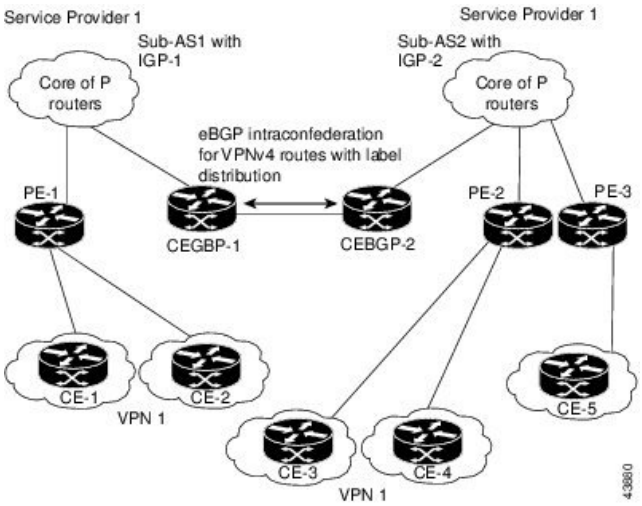
Note eBGP Connection Between Two Subautonomous Systems in a Confederation figure illustrates how two autonomous systems exchange routes and forward packets. Subautonomous systems in a confederation use a similar method of exchanging routes and forwarding packets.

The figure below illustrates a typical MPLS VPN confederation configuration. In this configuration:

- The two CEBGP border edge routers exchange VPN-IPv4 addresses with labels between the two autonomous systems.

- The distributing router changes the next-hop addresses and labels and uses a next-hop-self address.
- IGP-1 and IGP-2 know the addresses of CEBGP-1 and CEBGP-2.

Figure 2: eBGP Connection Between Two Subautonomous Systems in a Confederation



In this confederation configuration:

- CEBGP border edge routers function as neighboring peers between the subautonomous systems. The subautonomous systems use eBGP to exchange route information.
- Each CEBGP border edge router (CEBGP-1 and CEBGP-2) assigns a label for the router before distributing the route to the next subautonomous system. The CEBGP border edge router distributes the route as a VPN-IPv4 address by using the multiprotocol extensions of BGP. The label and the VPN identifier are encoded as part of the NLRI.
- Each PE and CEBGP border edge router assigns its own label to each VPN-IPv4 address prefix before redistributing the routes. The CEBGP border edge routers exchange IPV-IPv4 addresses with the labels. The next-hop-self address is included in the label (as the value of the eBGP next-hop attribute). Within the subautonomous systems, the CEBGP border edge router address is distributed throughout the iBGP neighbors, and the two CEBGP border edge routers are known to both confederations.
- For more information about how to configure confederations, see the Configuring MPLS Forwarding for ASBR Confederations, on page 61 .

MPLS VPN Inter-AS BGP Label Distribution

Note This section is not applicable to Inter-AS over IP tunnels.

You can set up the MPLS VPN Inter-AS network so that the ASBRs exchange IPv4 routes with MPLS labels of the provider edge (PE) routers. Route reflectors (RRs) exchange VPN-IPv4 routes by using multihop, multiprotocol external Border Gateway Protocol (eBGP). This method of configuring the Inter-AS system is often called MPLS VPN Inter-AS BGP Label Distribution.

Configuring the Inter-AS system so that the ASBRs exchange the IPv4 routes and MPLS labels has the following benefits:

- Saves the ASBRs from having to store all the VPN-IPv4 routes. Using the route reflectors to store the VPN-IPv4 routes and distributes them to the PE routers results in improved scalability compared with configurations in which the ASBR holds all the VPN-IPv4 routes and distributes the routes based on VPN-IPv4 labels.
- Having the route reflectors hold the VPN-IPv4 routes also simplifies the configuration at the border of the network.
- Enables a non-VPN core network to act as a transit network for VPN traffic. You can transport IPv4 routes with MPLS labels over a non-MPLS VPN service provider.
- Eliminates the need for any other label distribution protocol between adjacent label switch routers (LSRs). If two adjacent LSRs are also BGP peers, BGP can handle the distribution of the MPLS labels. No other label distribution protocol is needed between the two LSRs.

Exchanging IPv4 Routes with MPLS labels

Note This section is not applicable to Inter-AS over IP tunnels.

You can set up a VPN service provider network to exchange IPv4 routes with MPLS labels. You can configure the VPN service provider network as follows:

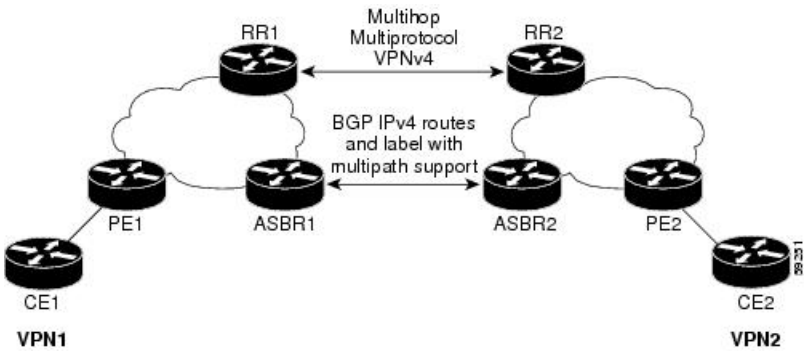
- Route reflectors exchange VPN-IPv4 routes by using multihop, multiprotocol eBGP. This configuration also preserves the next-hop information and the VPN labels across the autonomous systems.
- A local PE router (for example, PE1 in the figure below) needs to know the routes and label information for the remote PE router (PE2).

  This information can be exchanged between the PE routers and ASBRs in one of two ways:

  - Internal Gateway Protocol (IGP) and Label Distribution Protocol (LDP): The ASBR can redistribute the IPv4 routes and MPLS labels it learned from eBGP into IGP and LDP and from IGP and LDP into eBGP.
  - Internal Border Gateway Protocol (iBGP) IPv4 label distribution: The ASBR and PE router can use direct iBGP sessions to exchange VPN-IPv4 and IPv4 routes and MPLS labels.

Alternatively, the route reflector can reflect the IPv4 routes and MPLS labels learned from the ASBR to the PE routers in the VPN. This reflecting of learned IPv4 routes and MPLS labels is accomplished by enabling the ASBR to exchange IPv4 routes and MPLS labels with the route reflector. The route reflector also reflects the VPN-IPv4 routes to the PE routers in the VPN. For example, in VPN1, RR1 reflects to PE1 the VPN-IPv4 routes it learned and IPv4 routes and MPLS labels learned from ASBR1. Using the route reflectors to store the VPN-IPv4 routes and forward them through the PE routers and ASBRs allows for a scalable configuration.

Figure 3: VPNs Using eBGP and iBGP to Distribute Routes and MPLS Labels



BGP Routing Information

BGP routing information includes the following items:

- Network number (prefix), which is the IP address of the destination.
- Autonomous system (AS) path, which is a list of the other ASs through which a route passes on the way to the local router. The first AS in the list is closest to the local router; the last AS in the list is farthest from the local router and usually the AS where the route began.
- Path attributes, which provide other information about the AS path, for example, the next hop.

BGP Messages and MPLS Labels

MPLS labels are included in the update messages that a router sends. Routers exchange the following types of BGP messages:

- Open messages—After a router establishes a TCP connection with a neighboring router, the routers exchange open messages. This message contains the number of the autonomous system to which the router belongs and the IP address of the router that sent the message.
- Update messages—When a router has a new, changed, or broken route, it sends an update message to the neighboring router. This message contains the NLRI, which lists the IP addresses of the usable routes. The update message includes any routes that are no longer usable. The update message also includes path attributes and the lengths of both the usable and unusable paths. Labels for VPN-IPv4 routes are encoded in the update message, as specified in RFC 2858. The labels for the IPv4 routes are encoded in the update message, as specified in RFC 3107.
- Keepalive messages—Routers exchange keepalive messages to determine if a neighboring router is still available to exchange routing information. The router sends these messages at regular intervals. (Sixty seconds is the default for Cisco routers.) The keepalive message does not contain routing data; it contains only a message header.
- Notification messages—When a router detects an error, it sends a notification message.

Sending MPLS Labels with Routes

When BGP (eBGP and iBGP) distributes a route, it can also distribute an MPLS label that is mapped to that route. The MPLS label mapping information for the route is carried in the BGP update message that contains the information about the route. If the next hop is not changed, the label is preserved.

When you issue the **show bgp neighbors** *ip-address* command on both BGP routers, the routers advertise to each other that they can then send MPLS labels with the routes. If the routers successfully negotiate their ability to send MPLS labels, the routers add MPLS labels to all outgoing BGP updates.

Generic Routing Encapsulation Support for L3VPN

Generic Routing Encapsulation (GRE) is a tunneling protocol that can encapsulate many types of packets to enable data transmission using a tunnel. The GRE tunneling protocol enables:

- High assurance Internet Protocol encryptor (HAIPE) devices for encryption over the public Internet and nonsecure connections.
- Service providers (that do not run MPLS in their core network) to provide VPN services along with the security services.

GRE is used with IP to create a virtual point-to-point link to routers at remote points in a network. For detailed information about configuring GRE tunnel interfaces, see the *Implementing Generic Routing Encapsulation* module of the *MPLS Layer 3 VPN Configuration Guide for Cisco ASR 9000 Series Routers*.

Note GRE is used with IP to create a virtual point-to-point link to routers at remote points in a network. For detailed information about configuring GRE tunnel interfaces, refer to the *Cisco IOS XR Interfaces and Hardware Components Configuration Guide*. For a PE to PE (core) link, enable LDP (with implicit null) on the GRE interfaces for L3VPN.

GRE Restriction for L3VPN

The following restrictions are applicable to L3VPN forwarding over GRE:

- Carrier Supporting Carrier (CsC) or Inter-AS is not supported.
- GRE-based L3VPN does not interwork with MPLS or IP VPNs.
- GRE tunnel is supported only as a core link(PE-PE, PE-P, P-P, P-PE). A PE-CE (edge) link is not supported.
- VPNv6 forwarding using GRE tunnels is not supported.

VPNv4 Forwarding Using GRE Tunnels

This section describes the working of VPNv4 forwarding over GRE tunnels. The following description assumes that GRE is used only as a core link between the encapsulation and decapsulation provider edge (PE) routers that are connected to one or more customer edge (CE) routers.

Ingress of Encapsulation Router

On receiving prefixes from the CE routers, Border Gateway Protocol (BGP) assigns the VPN label to the prefixes that need to be exported. These VPN prefixes are then forwarded to the Forwarding Information Base (FIB) using the Route Information Base (RIB) or the label switched database (LSD). The FIB then populates the prefix in the appropriate VRF table. The FIB also populates the label in the global label table. Using BGP, the prefixes are then relayed to the remote PE router (decapsulation router).

Egress of Encapsulation Router

The forwarding behavior on egress of the encapsulation PE router is similar to the MPLS VPN label imposition. Regardless of whether the VPN label imposition is performed on the ingress or egress side, the GRE tunnel forwards a packet that has an associated label. This labeled packet is then encapsulated with a GRE header and forwarded based on the IP header.

Ingress of Decapsulation Router

The decapsulation PE router learns the VPN prefixes and label information from the remote encapsulation PE router using BGP. The next-hop information for the VPN prefix is the address of the GRE tunnel interface connecting the two PE routers. BGP downloads these prefixes to the RIB. The RIB downloads the routes to the FIB and the FIB installs the routes in the hardware.

Egress of Decapsulation Router

The egress forwarding behavior on the decapsulation PE router is similar to VPN disposition and forwarding, based on the protocol type of the inner payload.

Carrier Supporting Carrier Support for L3VPN

This section provides conceptual information about MPLS VPN Carrier Supporting Carrier (CSC) functionality and includes the following topics:

- CSC Prerequisites
- CSC Benefits
- Configuration Options for the Backbone and Customer Carriers

Throughout this document, the following terminology is used in the context of CSC:

backbone carrier—Service provider that provides the segment of the backbone network to the other provider. A backbone carrier offers BGP and MPLS VPN services.

customer carrier—Service provider that uses the segment of the backbone network. The customer carrier may be an Internet service provider (ISP) or a BGP/MPLS VPN service provider.

CE router—A customer edge router is part of a customer network and interfaces to a provider edge (PE) router. In this document, the CE router sits on the edge of the customer carrier network.

PE router—A provider edge router is part of a service provider's network connected to a customer edge (CE) router. In this document, the PE router sits on the edge of the backbone carrier network

ASBR—An autonomous system boundary router connects one autonomous system to another.

CSC Prerequisites

The following prerequisites are required to configure CSC:

- You must be able to configure MPLS VPNs with end-to-end (CE-to-CE router) pings working.
- You must be able to configure Interior Gateway Protocols (IGPs), MPLS Label Distribution Protocol (LDP), and Multiprotocol Border Gateway Protocol (MP-BGP).
- You must ensure that CSC-PE and CSC-CE routers support BGP label distribution.

Note BGP is the only supported label distribution protocol on the link between CE and PE.

CSC Benefits

This section describes the benefits of CSC to the backbone carrier and customer carriers.

Benefits to the Backbone Carrier

- The backbone carrier can accommodate many customer carriers and give them access to its backbone.
- The MPLS VPN carrier supporting carrier feature is scalable.
- The MPLS VPN carrier supporting carrier feature is a flexible solution.

Benefits to the Customer Carriers

- The MPLS VPN carrier supporting carrier feature removes from the customer carrier the burden of configuring, operating, and maintaining its own backbone.
- Customer carriers who use the VPN services provided by the backbone carrier receive the same level of security that Frame Relay or ATM-based VPNs provide.
- Customer carriers can use any link layer technology to connect the CE routers to the PE routers .
- The customer carrier can use any addressing scheme and still be supported by a backbone carrier.

Benefits of Implementing MPLS VPN CSC Using BGP

The benefits of using BGP to distribute IPv4 routes and MPLS label routes are:

- BGP takes the place of an IGP and LDP in a VPN forwarding and routing instance (VRF) table.
- BGP is the preferred routing protocol for connecting two ISPs.

Configuration Options for the Backbone and Customer Carriers

To enable CSC, the backbone and customer carriers must be configured accordingly:

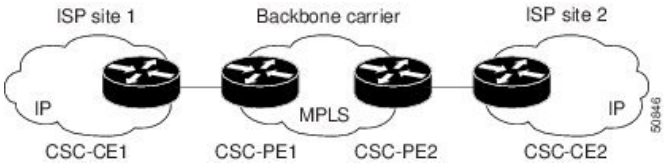
- The backbone carrier must offer BGP and MPLS VPN services.
- The customer carrier can take several networking forms. The customer carrier can be:
  - An ISP with an IP core (see the "Customer Carrier: ISP with IP Core").
  - An MPLS service provider with or without VPN services (see "Customer Carrier: MPLS Service Provider").

Note An IGP in the customer carrier network is used to distribute next hops and loopbacks to the CSC-CE. IBGP with label sessions are used in the customer carrier network to distribute next hops and loopbacks to the CSC-CE.

Customer Carrier: ISP with IP Core

The following figure shows a network configuration where the customer carrier is an ISP. The customer carrier has two sites, each of which is a point of presence (POP). The customer carrier connects these sites using a VPN service provided by the backbone carrier. The backbone carrier uses MPLS or IP tunnels to provide VPN services. The ISP sites use IP.

Figure 4: Network: Customer Carrier Is an ISP

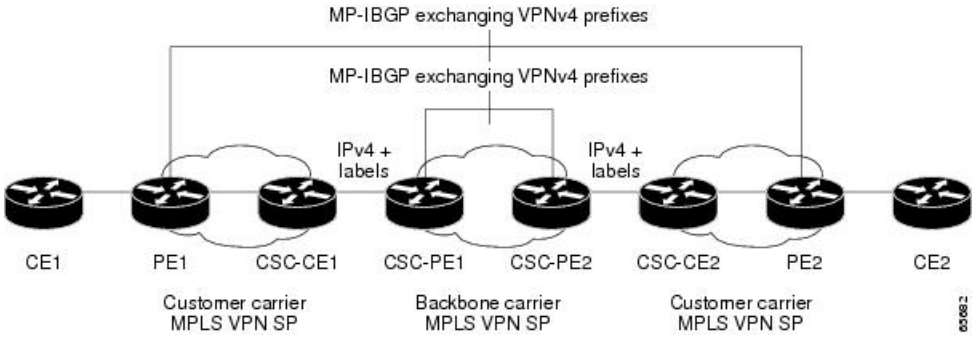


The links between the CE and PE routers use eBGP to distribute IPv4 routes and MPLS labels. Between the links, the PE routers use multiprotocol iBGP to distribute VPNv4 routes.

Customer Carrier: MPLS Service Provider

The following figure shows a network configuration where the backbone carrier and the customer carrier are BGP/MPLS VPN service providers. The customer carrier has two sites. The customer carrier uses MPLS in its network while the backbone carrier may use MPLS or IP tunnels in its network.

Figure 5: Network: Customer Carrier Is an MPLS VPN Service Provider



In Network: Customer Carrier Is an MPLS VPN Service Provider configuration, the customer carrier can configure its network in one of these ways:

- The customer carrier can run an IGP and LDP in its core network. In this case, the CSC-CE1 router in the customer carrier redistributes the eBGP routes it learns from the CSC-PE1 router of the backbone carrier to an IGP
- The CSC-CE1 router of the customer carrier system can run an IPv4 and labels iBGP session with the PE1 router.

LDP CSC IPv6

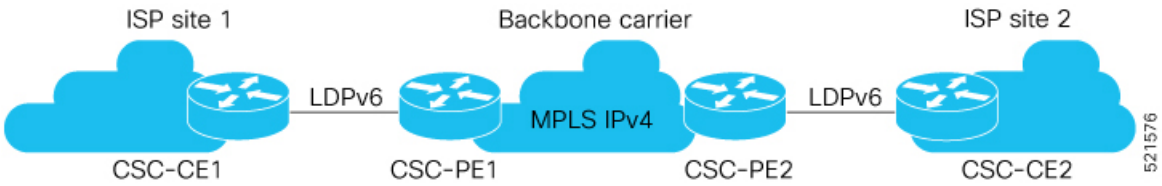
Table 2: Feature History Table

Feature Name	Release Information	Feature Description
LDP CSC IPv6	Release 7.4.1	The feature enables LDPv6 to run between CSC-PE and CSC-CE devices and carry IPv6 customer carrier traffic over the IPv4 backbone carrier network. LDP carries the labels that are exchanged in dedicated Virtual Private Network routing and forwarding (VRF) instances. IGP carries the routes between CSC-PE and CSC-CE routers.

IPv6 support for MPLS LDP enables the set up of label switched paths (LSPs) for IPv6 prefixes. This feature allows the IPv6 CSC-CE sites to communicate with each other over an MPLS IPv4 core network using MPLS LSPs. This feature enables the backbone carrier network running an MPLS IPv4 infrastructure to offer IPv6 services without any major changes in the infrastructure.

The topology shows two customers, CSC-CE1 and CSC-CE2, connecting their remote sites through the backbone carrier. The PE device of the backbone network connects with both customers through MPLS but under different VRFs according to interface-VRF mapping. The MPLS label distribution protocol for PE-CE connectivity is LDPv6 and requires them to run in a customer VRF context on the PE device.

Figure 6: LDP CSC IPv6



The links between the CE and PE routers use LDPv6 to distribute IPv6 routes and MPLS labels. The links between the PE routers use MPLS IPv4 network to distribute VPNv6 routes.

Configure LDP CSC IPv6

Perform the following tasks to configure LDP CSC IPv6.

- Configure LDPv6
- Configure BGP
- Configure ospfv3

Configuration Example

This section provides the configuration example.

```
/* CSC-PE1 Configuration */

/* Configure LDPv6 */
```

```
Router# configure
Router(config)# mpls ldp
Router(config-ldp)# vrf vpn2
Router(config-ldp-vrf)# router-id 10.0.0.1
Router(config-ldp-vrf)# address-family ipv6
Router(config-ldp-vrf-af)# discovery transport-address 10:10:10::1
Router(config-ldp-vrf-af)# exit
Router(config-ldp-vrf)# interface Bundle-Ether104.2
Router(config-ldp-vrf-if)# address-family ipv6
Router(config-ldp-vrf-if-af)# exit
Router(config-ldp-vrf-if)# exit
Router(config-ldp-vrf)# interface TenGigE0/0/0/12/3.2
Router(config-ldp-vrf-if)# address-family ipv6
Router(config-ldp-vrf-if-af)# commit

/* Configure BGP */
Router# configure
Router(config)# router bgp 100
Router(config-bgp)# nsr
Router(config-bgp)# bgp router-id 172.16.0.1
Router(config-bgp)# bgp redistribute-internal
Router(config-bgp)# bgp graceful-restart
Router(config-bgp)# bgp log neighbor changes detail
Router(config-bgp)# ibgp policy out enforce-modifications
Router(config-bgp)# address-family vpnv4 unicast
Router(config-bgp-af)# address-family vpnv6 unicast
Router(config-bgp-af)# neighbor-group vpn
Router(config-bgp-nbrgrp)# remote-as 100
Router(config-bgp-nbrgrp)# update-source Loopback0
Router(config-bgp-nbrgrp)# address-family vpnv4 unicast
Router(config-bgp-nbrgrp-af)# address-family vpnv6 unicast
Router(config-bgp-nbrgrp-af)# neighbor 192.168.0.1
Router(config-bgp-nbr)# use neighbor-group vpn
Router(config-bgp-nbr)# vrf vpn2
Router(config-bgp-vrf)# rd 1:2
Router(config-bgp-vrf)# address-family ipv4 unicast
Router(config-bgp-vrf-af)# exit
Router(config-bgp-vrf)# address-family ipv6 unicast
Router(config-bgp-vrf-af)# label mode per-prefix
Router(config-bgp-vrf-af)# maximum-paths ebgp 32
Router(config-bgp-vrf-af)# maximum-paths ibgp 32 unequal-cost
Router(config-bgp-vrf-af)# redistribute ospfv3 lab1 metric 19
Router(config-bgp-vrf-af)# commit

/* OSPF Configuration */
Router# configure
Router(config)# router ospfv3 LAB1
Router(config-ospfv3)# nsr
Router(config-ospfv3)# graceful-restart
Router(config-ospfv3)# vrf vpn2
Router(config-ospfv3-vrf)# mtu-ignore
Router(config-ospfv3-vrf)# graceful-restart
Router(config-ospfv3-vrf)# router-id 10.0.0.1
Router(config-ospfv3-vrf)# redistribute bgp 100 metric 19
Router(config-ospfv3-vrf)# area 0
Router(config-ospfv3-vrf-ar)# interface Loopback2
Router(config-ospfv3-vrf-ar-if)# passive
Router(config-ospfv3-vrf-ar-if)# interface Bundle-Ether104.2
Router(config-ospfv3-vrf-ar-if)# network point-to-point
Router(config-ospfv3-vrf-ar-if)# interface TenGigE0/0/0/12/3.2
Router(config-ospfv3-vrf-ar-if)# network point-to-point
Router(config-ospfv3-vrf-ar-if)# interface TenGigE0/1/0/7/3.2
```

```
Router(config-ospfv3-vrf-ar-if)# network point-to-point
Router(config-ospfv3-vrf-ar-if)# commit


/* CSC-CE1 Configuration */

/* Configure LDPv6 */
Router# configure
Router(config)# mpls ldp
Router(config-ldp)# vrf vpn2
Router(config-ldp-vrf)# router-id 209.165.201.1
Router(config-ldp-vrf)# address-family ipv6
Router(config-ldp-vrf-af)# discovery transport-address 209.165.201.1::1
Router(config-ldp-vrf-af)# exit
Router(config-ldp-vrf)# interface TenGigE0/3/0/29.2
Router(config-ldp-vrf-if)# address-family ipv6
Router(config-ldp-vrf-if-af)# exit
Router(config-ldp-vrf-if)# exit
Router(config-ldp-vrf)# interface Bundle-Ether104.2
Router(config-ldp-vrf-if)# address-family ipv6
Router(config-ldp-vrf-if-af)# commit

/* Configure BGP */
BGP configuration on CSC-CE is optional for LDP CSC IPv6 feature.
This configuration is required only if you want to run BGP between CSC-CEs.
Router# configure
Router(config)# router bgp 200
Router(config-bgp)# nsr
Router(config-bgp)# bgp router-id 209.165.201.2
Router(config-bgp)# bgp graceful-restart
Router(config-bgp)# bgp log neighbor changes detail
Router(config-bgp)# ibgp policy out enforce-modifications
Router(config-bgp)# address-family ipv6 unicast
Router(config-bgp-af)# address-family vpnv6 unicast
Router(config-bgp-af)# neighbor-group v6-csc
Router(config-bgp-nbrgrp)# remote-as 200
Router(config-bgp-nbrgrp)# address-family ipv6 unicast
Router(config-bgp-nbrgrp-af)# next-hop-self
Router(config-bgp-nbrgrp-af)# soft-reconfiguration inbound always
Router(config-bgp-nbrgrp-af)# neighbor-group v6-ixia
Router(config-bgp-nbrgrp)# remote-as 65001
Router(config-bgp-nbrgrp)# address-family ipv6 unicast
Router(config-bgp-nbrgrp-af)# route-policy pass in
Router(config-bgp-nbrgrp-af)# route-policy pass out
Router(config-bgp-nbrgrp-af)# as-override
Router(config-bgp-nbrgrp-af)# vrf vpn2
Router(config-bgp-vrf)# rd 4:2
Router(config-bgp-vrf)# address-family ipv6 unicast
Router(config-bgp-vrf-af)# label mode per-prefix
Router(config-bgp-vrf-af)# maximum-paths ibgp 8
Router(config-bgp-vrf-af)# neighbor 5:5:5::2
Router(config-bgp-vrf-nbr)# use neighbor-group v6-csc
Router(config-bgp-vrf-nbr)# update-source Loopback2
Router(config-bgp-vrf-nbr)# neighbor 104:1:1:2::2
Router(config-bgp-vrf-nbr)# use neighbor-group v6-ixia
Router(config-bgp-vrf-nbr)# commit


/* OSPF Configuration */
Router# configure
Router(config)# router ospfv3 LAB1
Router(config-ospfv3)# nsr
Router(config-ospfv3)# graceful-restart
```

```
Router(config-ospfv3)# vrf vpn2
Router(config-ospfv3-vrf)# mtu-ignore
Router(config-ospfv3-vrf)# graceful-restart
Router(config-ospfv3-vrf)# router-id 209.165.201.1
Router(config-ospfv3-vrf)# capability vrf-lite
Router(config-ospfv3-vrf)# area 0
Router(config-ospfv3-vrf-ar)# interface Loopback2
Router(config-ospfv3-vrf-ar-if)# passive
Router(config-ospfv3-vrf-ar-if)# interface Bundle-Ether104.2
Router(config-ospfv3-vrf-ar-if)# network point-to-point
Router(config-ospfv3-vrf-ar-if)# interface TenGigE0/3/0/29.2
Router(config-ospfv3-vrf-ar-if)# network point-to-point
Router(config-ospfv3-vrf-ar-if)# interface TenGigE0/3/0/18.2
Router(config-ospfv3-vrf-ar-if)# network point-to-point
Router(config-ospfv3-vrf-ar-if)# commit
```

Running Configuration

This section shows LDP CSC IPv6 running configuration

```
/* CSC-PE1 Configuration */

/* LDPv6 Configuration */
mpls ldp
 vrf vpn2
  router-id 10.0.0.1
  address-family ipv6
   discovery transport-address 10:10:10::1
  interface Bundle-Ether104.2
   address-family ipv6
interface TenGigE0/0/0/12/3.2
   address-family ipv6

/* BGP Configuration */
router bgp 100
 nsr
 bgp router-id 172.16.0.1
 bgp redistribute-internal
 bgp graceful-restart
 bgp log neighbor changes detail
 ibgp policy out enforce-modifications
 address-family vpnv4 unicast
 address-family vpnv6 unicast
neighbor-group vpn
  remote-as 100
  update-source Loopback0
  address-family vpnv4 unicast
  address-family vpnv6 unicast
neighbor 192.168.0.1
  use neighbor-group vpn
vrf vpn2
  rd 1:2
  address-family ipv4 unicast
  !
  address-family ipv6 unicast
   label mode per-prefix
   maximum-paths ebgp 32
   maximum-paths ibgp 32 unequal-cost
   redistribute ospfv3 lab1 metric 19

/* OSPF Configuration */
router ospfv3 lab1
```

```
 nsr
graceful-restart
vrf vpn2
  mtu-ignore
  graceful-restart
  router-id 1.1.1.2
  redistribute bgp 100 metric 19
  area 0
   interface Loopback2
    passive
  interface Bundle-Ether104.2
    network point-to-point
interface TenGigE0/0/0/12/3.2
    network point-to-point
interface TenGigE0/1/0/7/3.2
    network point-to-point


/* CSC-CE1 Configuration */

/* LDPv6 Configuration */
mpls ldp
 vrf vpn2
  router-id 209.165.201.1
  address-family ipv6
   discovery transport-address 209.165.201.1::2
  !
  interface TenGigE0/3/0/29.2
address-family ipv6
   !
  !
  interface Bundle-Ether104.2
   address-family ipv6
   !

/* BGP Configuration */
router bgp 200
 nsr
 bgp router-id 209.165.201.2
 bgp graceful-restart
 bgp log neighbor changes detail
 ibgp policy out enforce-modifications
 address-family ipv6 unicast
address-family vpnv6 unicast
neighbor-group v6-csc
  remote-as 200
  address-family ipv6 unicast
   next-hop-self
   soft-reconfiguration inbound always
neighbor-group v6-ixia
  remote-as 65001
  address-family ipv6 unicast
   route-policy pass in
   route-policy pass out
   as-override
vrf vpn2
  rd 4:2
  address-family ipv6 unicast
   label mode per-prefix
   maximum-paths ibgp 8
  neighbor 209.165.202.129::2
   use neighbor-group v6-csc
   update-source Loopback2
  neighbor 104:1:1:2::2
```

```
   use neighbor-group v6-ixia

/* OSPF Configuration */
router ospfv3 lab1
 nsr
 graceful-restart
 vrf vpn2
  graceful-restart
  router-id 4.4.1.2
  capability vrf-lite
  area 0
   interface Loopback2
    passive
   interface Bundle-Ether104.2
    network point-to-point
interface TenGigE0/3/0/29.2
    network point-to-point
interface TenGigE0/3/0/18.2
    network point-to-point
!
```

Verification

Verify the LDP CSC IPv6 configuration.

```
Router:CSC-PE1# show mpls ldp summary all
VRFs      : 254 (254 oper)
AFIs      : IPv4 (127), IPv6 (127)
Routes    : 9041 prefixes (789 ipv4, 8252 ipv6)
Bindings  : 68 prefixes (6 ipv4, 62 ipv6)
    Local     : 68 (6 ipv4, 62 ipv6)
    Remote    : 81 (6 ipv4, 75 ipv6)
Neighbors : 2000 (1 NSR, 1 GR)
Adj Groups: 2000
Hello Adj : 2255 (257 ipv4, 1998 ipv6)
Addresses : 3025 (390 ipv4, 2635 ipv6)
Interfaces: 2277 LDP configured (259 ipv4, 2018 ipv6)
            (4 auto-config)
Collaborators:
                     Connected  Registered
                     ---------  ------------------
    SysDB            Y          Y
    IM               Y          Y
    RSI              Y          -
    IP-ARM           Y          -
    IPv4-RIB         Y          Y (127/127 tables)
    IPv6-RIB         Y          Y (127/127 tables)

    LSD              Y          Y
    LDP-NSR-Partner  Y          -
    L2VPN-AToM       Y          -
    mLDP             -          N

Router:CSC-PE1# show mpls ldp vrf vpn2
parameters
LDP Parameters:
  Role: Active
  Protocol Version: 1
  RouDiscovery:
    Link Hellos:     Holdtime:15 sec, Interval:5 sec
    Targeted Hellos: Holdtime:90 sec, Interval:10 sec
    Quick-start: Enabled (by default)
    Transport address:
```

```
      IPv6: 1:1:1::2
Router ID: 1.1.1.2
  Null Label:
    IPv6: Implicit
  Session:
    Hold time: 180 sec
    Keepalive interval: 60 sec
    Backoff: Initial:15 sec, Maximum:120 sec
    Global MD5 password: Disabled
Graceful Restart:
    Enabled
    Reconnect Timeout:120 sec, Forwarding State Holdtime:180 sec
  NSR: Enabled, Sync-ed
  Timeouts:
    Housekeeping periodic timer: 10 sec
    Local binding: 300 sec
    Forwarding state in LSD: 360 sec
  Delay in AF Binding Withdrawl from peer: 180 sec
  Max:
    5000 interfaces (4000 attached, 1000 TE tunnel), 2000 peers
  OOR state
    Memory: Normal

Router:CSC-CE1# show mpls ldp vrf vpn2 ipv6 discovery detail
Local LDP Identifier: 209.165.201.1:0
Discovery Sources:
  Interfaces:
    Bundle-Ether104.2 (0x41e0) : xmit/recv
      VRF: 'vpn2' (0x60000004)
      Source address: fe80::226:51ff:fecc:6762; Transport address: 209.165.201.1::2
      Hello interval: 5 sec (due in 4.2 sec)
      Quick-start: Enabled
      LDP Id: 10.0.0.1:0
          Source address: fe80::72e4:22ff:fe57:209e; Transport address: 10:10:10::1
          Hold time: 15 sec (local:15 sec, peer:15 sec)
                     (expiring in 14.8 sec)
          Established: Apr  5 14:18:22.000 (1d01h ago)
          Last session connection failures:
             Apr  5 15:08:55.074: TCP connection closed
                 (Last up for 00:50:02)
Apr  5 15:08:55.074: TCP connection closed
                 (Last up for 00:50:02)
TenGigE0/3/0/29.2 (0xa007880) : xmit/recv
      VRF: 'vpn2' (0x60000004)
      Source address: fe80::2a7:42ff:fe56:fc75; Transport address: 09.165.201.1::2
      Hello interval: 5 sec (due in 977 msec)
      Quick-start: Enabled
      LDP Id: 1.1.1.2:0
          Source address: fe80::822d:bfff:fe17:e9b3; Transport address: 10:10:10::1
          Hold time: 15 sec (local:15 sec, peer:15 sec)
                     (expiring in 13.1 sec)
          Established: Apr  5 14:18:19.731 (1d01h ago)
          Last session connection failures:
             Apr  5 15:08:55.074: TCP connection closed
                 (Last up for 00:50:02)
```

How to Implement MPLS Layer 3 VPNs

This section contains instructions for the following tasks:

Configuring the Core Network

Configuring the core network includes the following tasks:

Assessing the Needs of MPLS VPN Customers

Before configuring an MPLS VPN, the core network topology must be identified so that it can best serve MPLS VPN customers. Perform this task to identify the core network topology.

SUMMARY STEPS

1. Identify the size of the network.
2. Identify the routing protocols in the core.
3. Determine if MPLS High Availability support is required.
4. Determine if BGP load sharing and redundant paths are required.

DETAILED STEPS

Step 1 Identify the size of the network.

Identify the following to determine the number of routers and ports required:

- How many customers will be supported?
- How many VPNs are required for each customer?
- How many virtual routing and forwarding (VRF) instances are there for each VPN?

Step 2 Identify the routing protocols in the core.

Determine which routing protocols are required in the core network.

Step 3 Determine if MPLS High Availability support is required.

MPLS VPN nonstop forwarding and graceful restart are supported on select routers and Cisco IOS XR software releases.

Step 4 Determine if BGP load sharing and redundant paths are required.

Determine if BGP load sharing and redundant paths in the MPLS VPN core are required.

Configuring Routing Protocols in the Core

To configure a routing protocol, see the *Routing Configuration Guide for Cisco ASR 9000 Series Routers*.

Configuring MPLS in the Core

To enable MPLS on all routers in the core, you must configure a Label Distribution Protocol (LDP). You can use either of the following as an LDP:

- MPLS LDP—See the *Implementing MPLS Label Distribution Protocol* chapter in the *MPLS Configuration Guide for Cisco ASR 9000 Series Routers* for configuration information.

- MPLS Traffic Engineering Resource Reservation Protocol (RSVP)—See *Implementing RSVP for MPLS-TE* module in the *MPLS Configuration Guide for Cisco ASR 9000 Series Routers* for configuration information.

Determining if FIB Is Enabled in the Core

Forwarding Information Base (FIB) must be enabled on all routers in the core, including the provider edge (PE) routers. For information on how to determine if FIB is enabled, see the *Implementing Cisco Express Forwarding* module in the *IP Addresses and Services Configuration Guide for Cisco ASR 9000 Series Routers*.

Configuring Multiprotocol BGP on the PE Routers and Route Reflectors

Perform this task to configure multiprotocol BGP (MP-BGP) connectivity on the PE routers and route reflectors.

SUMMARY STEPS

1. **configure**
2. **router bgp** *autonomous-system-number*
3. **address-family vpnv4 unicast** or **address-family vpnv6 unicast**
4. **neighbor** *ip-address* **remote-as** *autonomous-system-number*
5. **address-family vpnv4 unicast** or **address-family vpnv6 unicast**
6. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

```
RP/0/RSP0/CPU0:router# configure
```

Enters the Global Configuration mode.

Step 2 **router bgp** *autonomous-system-number*

Example:

```
RP/0/RSP0/CPU0:router(config)# router bgp 120
```

Enters BGP configuration mode allowing you to configure the BGP routing process.

Step 3 **address-family vpnv4 unicast** or **address-family vpnv6 unicast**

Example:

```
RP/0/RSP0/CPU0:router(config-bgp)# address-family vpnv4 unicast
```

Enters VPNv4 or VPNv6 address family configuration mode for the VPNv4 or VPNv6 address family.

Step 4 **neighbor** *ip-address* **remote-as** *autonomous-system-number*

Example:

```
RP/0/RSP0/CPU0:router(config-bgp)# neighbor 172.168.40.24 remote-as 2002
```

Creates a neighbor and assigns it a remote autonomous system number.

Step 5 **address-family vpnv4 unicast** or **address-family vpnv6 unicast**

Example:

```
RP/0/RSP0/CPU0:router(config-bgp-nbr)# address-family vpnv4 unicast
```

Enters VPNv4 or VPNv6 address family configuration mode for the VPNv4 or VPNv6 address family.

Step 6 Use the **commit** or **end** command.

commit - Saves the configuration changes and remains within the configuration session.

end - Prompts user to take one of these actions:

- **Yes** - Saves configuration changes and exits the configuration session.
- **No** - Exits the configuration session without committing the configuration changes.
- **Cancel** - Remains in the configuration mode, without committing the configuration changes.

Connecting MPLS VPN Customers

To connect MPLS VPN customers to the VPN, perform the following tasks:

Defining VRFs on the PE Routers to Enable Customer Connectivity

Perform this task to define VPN routing and forwarding (VRF) instances.

SUMMARY STEPS

1. **configure**
2. **vrf** *vrf-name*
3. **address-family ipv4 unicast**
4. **import route-policy** *policy-name*
5. **import route-target** [*as-number:nn* | *ip-address:nn*]
6. **export route-policy** *policy-name*
7. **export route-target** [*as-number:nn* | *ip-address:nn*]
8. **exit**
9. **exit**
10. **router bgp** *autonomous-system-number*
11. **vrf** *vrf-name*
12. **rd** { *as-number* | *ip-address* | **auto** }
13. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

```
RP/0/RSP0/CPU0:router# configure
```

Enters Global Configuration mode.

Step 2 **vrf** *vrf-name*

Example:

```
RP/0/RSP0/CPU0:router(config)# vrf vrf_1
```

Configures a VRF instance and enters VRF configuration mode.

Step 3 **address-family ipv4 unicast**

Example:

```
RP/0/RSP0/CPU0:router(config-vrf)# address-family ipv4 unicast
```

Enters VRF address family configuration mode for the IPv4 address family.

Step 4 **import route-policy** *policy-name*

Example:

```
RP/0/RSP0/CPU0:router(config-vrf-af)# import route-policy policy_A
```

Specifies a route policy that can be imported into the local VPN.

Step 5 **import route-target** [*as-number:nn* | *ip-address:nn*]

Example:

```
RP/0/RSP0/CPU0:router(config-vrf-af)# import route-target 120:1
```

Allows exported VPN routes to be imported into the VPN if one of the route targets of the exported route matches one of the local VPN import route targets.

Step 6 **export route-policy** *policy-name*

Example:

```
RP/0/RSP0/CPU0:router(config-vrf-af)# export route-policy policy_B
```

Specifies a route policy that can be exported from the local VPN.

Step 7 **export route-target** [*as-number:nn* | *ip-address:nn*]

Example:

```
RP/0/RSP0/CPU0:router(config-vrf-af)# export route-target 120:2
```

Associates the local VPN with a route target. When the route is advertised to other provider edge (PE) routers, the export route target is sent along with the route as an extended community.

Step 8 **exit**

Example:

```
RP/0/RSP0/CPU0:router(config-vrf-af)# exit
```

Exits VRF address family configuration mode and returns the router to VRF configuration mode.

Step 9 **exit**

Example:

```
RP/0/RSP0/CPU0:router(config-vrf)# exit
```

Exits VRF configuration mode and returns the router to Global Configuration mode.

Step 10 **router bgp** *autonomous-system-number*

Example:

```
RP/0/RSP0/CPU0:router(config)# router bgp 120
```

Enters BGP configuration mode allowing you to configure the BGP routing process.

Step 11 **vrf** *vrf-name*

Example:

```
RP/0/RSP0/CPU0:router(config-bgp)# vrf vrf_1
```

Configures a VRF instance and enters VRF configuration mode for BGP routing.

Step 12 **rd** { *as-number* | *ip-address* | **auto** }

Example:

```
RP/0/RSP0/CPU0:router(config-bgp-vrf)# rd auto
```

Automatically assigns a unique route distinguisher (RD) to vrf_1.

Step 13 Use the **commit** or **end** command.

commit - Saves the configuration changes and remains within the configuration session.

end - Prompts user to take one of these actions:

- **Yes** - Saves configuration changes and exits the configuration session.
- **No** - Exits the configuration session without committing the configuration changes.

- **Cancel** - Remains in the configuration mode, without committing the configuration changes.

Configuring VRF Interfaces on PE Routers for Each VPN Customer

Perform this task to associate a VPN routing and forwarding (VRF) instance with an interface or a subinterface on the PE routers.

Note You must remove IPv4/IPv6 addresses from an interface prior to assigning, removing, or changing an interface's VRF. If this is not done in advance, any attempt to change the VRF on an IP interface is rejected.

SUMMARY STEPS

1. **configure**
2. **interface** *type interface-path-id*
3. **vrf** *vrf-name*
4. **ipv4 address** *ipv4-address mask*
5. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

```
RP/0/RSP0/CPU0:router# configure
```

Enters Global Configuration mode.

Step 2 **interface** *type interface-path-id*

Example:

```
RP/0/RSP0/CPU0:router(config)# interface TenGigE 0/3/0/0
```

Enters interface configuration mode.

Step 3 **vrf** *vrf-name*

Example:

```
RP/0/RSP0/CPU0:router(config-if)# vrf vrf_A
```

Configures a VRF instance and enters VRF configuration mode.

Step 4 **ipv4 address** *ipv4-address mask*

Example:

```
RP/0/RSP0/CPU0:router(config-if)# ipv4 address 192.168.1.27 255.255.255.0
```

Configures a primary IPv4 address for the specified interface.

Step 5 Use the **commit** or **end** command.

commit - Saves the configuration changes and remains within the configuration session.

end - Prompts user to take one of these actions:

- **Yes** - Saves configuration changes and exits the configuration session.
- **No** - Exits the configuration session without committing the configuration changes.
- **Cancel** - Remains in the configuration mode, without committing the configuration changes.

Configuring BGP as the Routing Protocol Between the PE and CE Routers

Perform this task to configure PE-to-CE routing sessions using BGP.

SUMMARY STEPS

1. **configure**
2. **router bgp** *autonomous-system-number*
3. **bgp router-id** {*ip-address*}
4. **vrf** *vrf-name*
5. **label-allocation-mode per-ce**
6. **address-family ipv4 unicast**
7. Do one of the following:
   - **redistribute connected** [**metric** *metric-value*] [**route-policy** *route-policy-name*]
   - **redistribute isis** *process-id* [**level** {**1** | **1-inter-area** | **2** }] [**metric** *metric-value*] [**route-policy** *route-policy-name*]
   - **redistribute ospf** *process-id* [**match** { **external** [**1** | **2**] | **internal** | **nssa-external** [**1** | **2**] }] [**metric** *metric-value*] [**route-policy** *route-policy-name*]
   - **redistribute static** [**metric** *metric-value*] [**route-policy** *route-policy-name*]
8. **aggregate-address** *address/mask-length* [**as-set**] [**as-confed-set**] [**summary-only**] [**route-policy** *route-policy-name*]
9. **network** {*ip-address/prefix-length* | *ip-address mask* } [**route-policy** *route-policy-name*]
10. **exit**
11. **neighbor** *ip-address*
12. **remote-as** *autonomous-system-number*
13. **password** { **clear** | **encrypted** } *password*
14. **ebgp-multihop** [*ttl-value*]
15. **address-family ipv4 unicast**
16. **allowas-in** [*as-occurrence-number*]
17. **route-policy** *route-policy-name* **in**
18. **route-policy** *route-policy-name* **out**
19. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

```
RP/0/RSP0/CPU0:router# configure
```

Enters Global Configuration mode.

Step 2 **router bgp** *autonomous-system-number*

Example:

```
RP/0/RSP0/CPU0:router(config)# router bgp 120
```

Enters Border Gateway Protocol (BGP) configuration mode allowing you to configure the BGP routing process.

Step 3 **bgp router-id** {*ip-address*}

Example:

```
RP/0/RSP0/CPU0:router(config-bgp)# bgp router-id 192.168.70.24
```

Configures the local router with a router ID of 192.168.70.24.

Step 4 **vrf** *vrf-name*

Example:

```
RP/0/RSP0/CPU0:router(config-bgp)# vrf vrf_1
```

Configures a VPN routing and forwarding (VRF) instance and enters VRF configuration mode for BGP routing.

Step 5 **label-allocation-mode per-ce**

Example:

```
RP/0/RSP0/CPU0:router(config-bgp-vrf)# label-allocation-mode per-ce
```

Sets the MPLS VPN label allocation mode for each customer edge (CE) label mode allowing the provider edge (PE) router to allocate one label for every immediate next-hop.

Step 6 **address-family ipv4 unicast**

Example:

```
RP/0/RSP0/CPU0:router(config-bgp-vrf)# address-family ipv4 unicast
```

Enters VRF address family configuration mode for the IPv4 address family.

Step 7 Do one of the following:

- **redistribute connected** [**metric** *metric-value*] [**route-policy** *route-policy-name*]

- **redistribute isis** *process-id* [**level** {**1** | **1-inter-area** | **2** }] [**metric** *metric-value*] [**route-policy** *route-policy-name*]
- **redistribute ospf** *process-id* [**match** { **external** [**1** | **2**] | **internal** | **nssa-external** [**1** | **2**] }] [**metric** *metric-value*] [**route-policy** *route-policy-name*]
- **redistribute static** [**metric** *metric-value*] [**route-policy** *route-policy-name*]

Example:

```
RP/0/RSP0/CPU0:router(config-bgp-vrf-af)# redistribute connected
```

Causes routes to be redistributed into BGP. The routes that can be redistributed into BGP are:

- Connected
- Intermediate System-to-Intermediate System (IS-IS)
- Open Shortest Path First (OSPF)
- Static

Step 8 **aggregate-address** *address/mask-length* [**as-set**] [**as-confed-set**] [**summary-only**] [**route-policy** *route-policy-name*]

Example:

```
RP/0/RSP0/CPU0:router(config-bgp-vrf-af)# aggregate-address 10.0.0.0/8 as-set
```

Creates an aggregate address. The path advertised for this route is an autonomous system set consisting of all elements contained in all paths that are being summarized.

- The **as-set** keyword generates autonomous system set path information and community information from contributing paths.
- The **as-confed-set** keyword generates autonomous system confederation set path information from contributing paths.
- The **summary-only** keyword filters all more specific routes from updates.
- The **route-policy** *route-policy-name* keyword and argument specify the route policy used to set the attributes of the aggregate route.

Step 9 **network** {*ip-address/prefix-length* | *ip-address mask* } [**route-policy** *route-policy-name*]

Example:

```
RP/0/RSP0/CPU0:router(config-bgp-vrf-af)# network 172.20.0.0/16
```

Configures the local router to originate and advertise the specified network.

Step 10 **exit**

Example:

```
RP/0/RSP0/CPU0:router(config-bgp-vrf-af)# exit
```

Exits VRF address family configuration mode and returns the router to VRF configuration mode for BGP routing.

Step 11 **neighbor** *ip-address*

Example:

```
RP/0/RSP0/CPU0:router(config-bgp-vrf)# neighbor 172.168.40.24
```

Places the router in VRF neighbor configuration mode for BGP routing and configures the neighbor IP address 172.168.40.24 as a BGP peer.

Step 12 **remote-as** *autonomous-system-number*

Example:

```
RP/0/RSP0/CPU0:router(config-bgp-vrf-nbr)# remote-as 2002
```

Creates a neighbor and assigns it a remote autonomous system number.

Step 13 **password** { **clear** | **encrypted** } *password*

Example:

```
RP/0/RSP0/CPU0:router(config-bgp-vrf-nbr)# password clear pswd123
```

Configures neighbor 172.168.40.24 to use MD5 authentication with the password pswd123.

Step 14 **ebgp-multihop** [*ttl-value*]

Example:

```
RP/0/RSP0/CPU0:router(config-bgp-vrf-nbr)# ebgp-multihop
```

Allows a BGP connection to neighbor 172.168.40.24.

Step 15 **address-family ipv4 unicast**

Example:

```
RP/0/RSP0/CPU0:router(config-bgp-vrf-nbr)# address-family ipv4 unicast
```

Enters VRF neighbor address family configuration mode for BGP routing.

Step 16 **allowas-in** [*as-occurrence-number*]

Example:

```
RP/0/RSP0/CPU0:router(config-bgp-vrf-nbr-af)# allowas-in 3
```

Replaces the neighbor autonomous system number (ASN) with the PE ASN in the AS path three times.

Step 17 **route-policy** *route-policy-name* **in**

Example:

```
RP/0/RSP0/CPU0:router(config-bgp-vrf-nbr-af)# route-policy In-Ipv4 in
```

Applies the In-Ipv4 policy to inbound IPv4 unicast routes.

Step 18 **route-policy** *route-policy-name* **out**

Example:

```
RP/0/RSP0/CPU0:router(config-bgp-vrf-nbr-af)# route-policy In-Ipv4 in
```

Applies the In-Ipv4 policy to outbound IPv4 unicast routes.

Step 19 Use the **commit** or **end** command.

commit - Saves the configuration changes and remains within the configuration session.

end - Prompts user to take one of these actions:

- **Yes** - Saves configuration changes and exits the configuration session.
- **No** - Exits the configuration session without committing the configuration changes.
- **Cancel** - Remains in the configuration mode, without committing the configuration changes.

Configuring RIPv2 as the Routing Protocol Between the PE and CE Routers

Perform this task to configure provider edge (PE)-to-customer edge (CE) routing sessions using Routing Information Protocol version 2 (RIPv2).

SUMMARY STEPS

1. **configure**
2. **router rip**
3. **vrf** *vrf-name*
4. **interface** *type instance*
5. **site-of-origin** { *as-number : number* | *ip-address : number* }
6. **exit**
7. Do one of the following:
   - **redistribute bgp** *as-number* [[**external** | **internal** | **local**] [**route-policy** *name*]
   - **redistribute connected** [**route-policy** *name*]
   - **redistribute isis** *process-id* [**level-1** | **level-1-2** | **level-2**] [**route-policy** *name*]
   - **redistribute eigrp** *as-number* [**route-policy** *name*]
   - **redistribute ospf** *process-id* [**match** { **external** [**1** | **2**] | **internal** | **nssa-external** [**1** | **2**] }] [**route-policy** *name*]
   - **redistribute static** [**route-policy** *name*]
8. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

```
RP/0/RSP0/CPU0:router# configure
```

Enters Global Configuration mode.

Step 2 **router rip**

Example:

```
RP/0/RSP0/CPU0:router(config)# router rip
```

Enters the Routing Information Protocol (RIP) configuration mode allowing you to configure the RIP routing process.

Step 3 **vrf** *vrf-name*

Example:

```
RP/0/RSP0/CPU0:router(config-rip)# vrf vrf_1
```

Configures a VPN routing and forwarding (VRF) instance and enters VRF configuration mode for RIP routing.

Step 4 **interface** *type instance*

Example:

```
RP/0/RSP0/CPU0:router(config-rip-vrf)# interface TenGigE 0/3/0/0
```

Enters VRF interface configuration mode.

Step 5 **site-of-origin** { *as-number : number* | *ip-address : number* }

Example:

```
RP/0/RSP0/CPU0:router(config-rip-vrf-if)# site-of-origin 200:1
```

Identifies routes that have originated from a site so that the re-advertisement of that prefix back to the source site can be prevented. Uniquely identifies the site from which a PE router has learned a route.

Step 6 **exit**

Example:

```
RP/0/RSP0/CPU0:router(config-rip-vrf-if)# exit
```

Exits VRF interface configuration mode, and returns the router to VRF configuration mode for RIP routing.

Step 7 Do one of the following:

- **redistribute bgp** *as-number* [[**external** | **internal** | **local**] [**route-policy** *name*]
- **redistribute connected** [**route-policy** *name*]
- **redistribute isis** *process-id* [**level-1** | **level-1-2** | **level-2**] [**route-policy** *name*]
- **redistribute eigrp** *as-number* [**route-policy** *name*]
- **redistribute ospf** *process-id* [**match** { **external** [**1** | **2**] | **internal** | **nssa-external** [**1** | **2**] }] [**route-policy** *name*]
- **redistribute static** [**route-policy** *name*]

Example:

```
RP/0/RSP0/CPU0:router(config-rip-vrf)# redistribute connected
```

Causes routes to be redistributed into RIP. The routes that can be redistributed into RIP are:

- Border Gateway Protocol (BGP)
- Connected
- Enhanced Interior Gateway Routing Protocol (EIGRP)
- Intermediate System-to-Intermediate System (IS-IS)
- Open Shortest Path First (OSPF)
- Static

Step 8 Use the **commit** or **end** command.

commit - Saves the configuration changes and remains within the configuration session.

end - Prompts user to take one of these actions:

- **Yes** - Saves configuration changes and exits the configuration session.
- **No** - Exits the configuration session without committing the configuration changes.
- **Cancel** - Remains in the configuration mode, without committing the configuration changes.

Configuring Static Routes Between the PE and CE Routers

Perform this task to configure provider edge (PE)-to-customer edge (CE) routing sessions that use static routes.

Note You must remove IPv4/IPv6 addresses from an interface prior to assigning, removing, or changing an interface's VRF. If this is not done in advance, any attempt to change the VRF on an IP interface is rejected.

SUMMARY STEPS

1. **configure**
2. **router static**
3. **vrf** *vrf-name*
4. **address-family ipv4 unicast**
5. *prefix/mask* [**vrf** *vrf-name*] { *ip-address* | *type interface-path-id* }
6. *prefix/mask* [**vrf** *vrf-name*] **bfd fast-detect**
7. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

```
RP/0/RSP0/CPU0:router# configure
```

Enters Global Configuration mode.

Step 2 **router static**

Example:

```
RP/0/RSP0/CPU0:router(config)# router static
```

Enters static routing configuration mode allowing you to configure the static routing process.

Step 3 **vrf** *vrf-name*

Example:

```
RP/0/RSP0/CPU0:router(config-static)# vrf vrf_1
```

Configures a VPN routing and forwarding (VRF) instance and enters VRF configuration mode for static routing.

Step 4 **address-family ipv4 unicast**

Example:

```
RP/0/RSP0/CPU0:router(config-static-vrf)# address-family ipv4 unicast
```

Enters VRF address family configuration mode for the IPv4 address family.

Step 5 *prefix/mask* [**vrf** *vrf-name*] { *ip-address* | *type interface-path-id* }

Example:

```
RP/0/RSP0/CPU0:router(config-static-vrf-afi)# 172.168.40.24/24 vrf vrf_1 10.1.1.1
```

Assigns the static route to vrf_1.

Step 6 *prefix/mask* [**vrf** *vrf-name*] **bfd fast-detect**

Example:

```
RP/0/RSP0/CPU0:router(config-static-vrf-afi)# 172.168.40.24/24 vrf vrf_1 bfd fast-detect
```

Enables bidirectional forwarding detection (BFD) to detect failures in the path between adjacent forwarding engines.

This option is available is when the forwarding router address is specified in Step 5 .

Step 7 Use the **commit** or **end** command.

commit - Saves the configuration changes and remains within the configuration session.

end - Prompts user to take one of these actions:

- **Yes** - Saves configuration changes and exits the configuration session.
- **No** - Exits the configuration session without committing the configuration changes.
- **Cancel** - Remains in the configuration mode, without committing the configuration changes.

Configuring OSPF as the Routing Protocol Between the PE and CE Routers

Perform this task to configure provider edge (PE)-to-customer edge (CE) routing sessions that use Open Shortest Path First (OSPF).

SUMMARY STEPS

1. **configure**
2. **router ospf** *process-name*
3. **vrf** *vrf-name*
4. **router-id** {*router-id* | type interface-path-id}
5. Do one of the following:
   - **redistribute bgp** *process-id* [**metric** *metric-value*] [**metric-type** {**1** | **2**}] [**route-policy** *policy-name*] [**tag** *tag-value*]
   - **redistribute connected** [**metric** *metric-value*] [**metric-type** {**1** | **2**}] [**route-policy** *policy-name*] [**tag** *tag-value*]
   - **redistribute ospf** *process-id* [**match** {**external** [**1** | **2**] | **internal** | **nssa-external** [**1** | **2**]}] [**metric** *metric-value*] [**metric-type** {**1** | **2**}] [**route-policy** *policy-name*] [**tag** *tag-value*]
   - **redistribute static** [**metric** *metric-value*] [**metric-type** {**1** | **2**}] [**route-policy** *policy-name*] [**tag** *tag-value*]
   - **redistribute eigrp** *process-id* [**match** {**external** [**1** | **2**] | **internal** | **nssa-external** [**1** | **2**]]} [**metric** *metric-value*] [**metric-type** {**1** | **2**}] [**route-policy** *policy-name*] [**tag** *tag-value*]
   - **redistribute rip** [**metric** *metric-value*] [**metric-type** {**1** | **2**}] [**route-policy** *policy-name*] [**tag** *tag-value*]
6. **area** *area-id*
7. **interface** type interface-path-id
8. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

```
RP/0/RSP0/CPU0:router# configure
```

Enters Global Configuration mode.

Step 2 **router ospf** *process-name*

Example:

```
RP/0/RSP0/CPU0:router(config)# router ospf 109
```

Enters OSPF configuration mode allowing you to configure the OSPF routing process.

Step 3 **vrf** *vrf-name*

Example:

```
RP/0/RSP0/CPU0:router(config-ospf)# vrf vrf_1
```

Configures a VPN routing and forwarding (VRF) instance and enters VRF configuration mode for OSPF routing.

Step 4 **router-id** {*router-id* | type interface-path-id}

Example:

```
RP/0/RSP0/CPU0:router(config-ospf-vrf)# router-id 172.20.10.10
```

Configures the router ID for the OSPF routing process.

Step 5 Do one of the following:

- **redistribute bgp** *process-id* [**metric** *metric-value*] [**metric-type** {**1** | **2**}] [**route-policy** *policy-name*] [**tag** *tag-value*]
- **redistribute connected** [**metric** *metric-value*] [**metric-type** {**1** | **2**}] [**route-policy** *policy-name*] [**tag** *tag-value*]
- **redistribute ospf** *process-id* [**match** {**external** [**1** | **2**] | **internal** | **nssa-external** [**1** | **2**]}] [**metric** *metric-value*] [**metric-type** {**1** | **2**}] [**route-policy** *policy-name*] [**tag** *tag-value*]
- **redistribute static** [**metric** *metric-value*] [**metric-type** {**1** | **2**}] [**route-policy** *policy-name*] [**tag** *tag-value*]
- **redistribute eigrp** *process-id* [**match** {**external** [**1** | **2**] | **internal** | **nssa-external** [**1** | **2**]]} [**metric** *metric-value*] [**metric-type** {**1** | **2**}] [**route-policy** *policy-name*] [**tag** *tag-value*]
- **redistribute rip** [**metric** *metric-value*] [**metric-type** {**1** | **2**}] [**route-policy** *policy-name*] [**tag** *tag-value*]

Example:

```
RP/0/RSP0/CPU0:router(config-ospf-vrf)# redistribute connected
```

Causes routes to be redistributed into OSPF. The routes that can be redistributed into OSPF are:

- Border Gateway Protocol (BGP)
- Connected
- Enhanced Interior Gateway Routing Protocol (EIGRP)
- OSPF
- Static
- Routing Information Protocol (RIP)

Step 6 **area** *area-id*

Example:

```
RP/0/RSP0/CPU0:router(config-ospf-vrf)# area 0
```

Configures the OSPF area as area 0.

Step 7 **interface** type interface-path-id

Example:

```
RP/0/RSP0/CPU0:router(config-ospf-vrf-ar)# interface TenGigE 0/3/0/0
```

Associates interface TenGigE 0/3/0/0 with area 0.

Step 8 Use the **commit** or **end** command.

commit - Saves the configuration changes and remains within the configuration session.

end - Prompts user to take one of these actions:

- **Yes** - Saves configuration changes and exits the configuration session.
- **No** - Exits the configuration session without committing the configuration changes.
- **Cancel** - Remains in the configuration mode, without committing the configuration changes.

Configuring EIGRP as the Routing Protocol Between the PE and CE Routers

Perform this task to configure provider edge (PE)-to-customer edge (CE) routing sessions that use Enhanced Interior Gateway Routing Protocol (EIGRP).

Using EIGRP between the PE and CE routers allows you to transparently connect EIGRP customer networks through an MPLS-enable Border Gateway Protocol (BGP) core network so that EIGRP routes are redistributed through the VPN across the BGP network as internal BGP (iBGP) routes.

Before you begin

BGP is configured in the network. See the *Implementing BGP* module in the *Routing Configuration Guide for Cisco ASR 9000 Series Routers*

Note You must remove IPv4/IPv6 addresses from an interface prior to assigning, removing, or changing an interface's VRF. If this is not done in advance, any attempt to change the VRF on an IP interface is rejected.

SUMMARY STEPS

1. **configure**
2. **router eigrp** *as-number*
3. **vrf** *vrf-name*
4. **address-family ipv4**
5. **router-id** *router-id*
6. **autonomous-system** *as-number*
7. **default-metric** *bandwidth delay reliability loading mtu*
8. **redistribute** { { **bgp** | **connected** | **isis** | **ospf** | **rip** | **static** } [*as-number* | *instance-name*] } [**route-policy** *name*]
9. **interface** *type interface-path-id*
10. **site-of-origin** { *as-number:number* | *ip-address : number* }
11. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

```
RP/0/RSP0/CPU0:router# configure
```

Enters Global Configuration mode.

Step 2 **router eigrp** *as-number*

Example:

```
RP/0/RSP0/CPU0:router(config)# router eigrp 24
```

Enters EIGRP configuration mode allowing you to configure the EIGRP routing process.

Step 3 **vrf** *vrf-name*

Example:

```
RP/0/RSP0/CPU0:router(config-eigrp)# vrf vrf_1
```

Configures a VPN routing and forwarding (VRF) instance and enters VRF configuration mode for EIGRP routing.

Step 4 **address-family ipv4**

Example:

```
RP/0/RSP0/CPU0:router(config-eigrp-vrf)# address family ipv4
```

Enters VRF address family configuration mode for the IPv4 address family.

Step 5 **router-id** *router-id*

Example:

```
RP/0/RSP0/CPU0:router(config-eigrp-vrf-af)# router-id 172.20.0.0
```

Configures the router ID for the Enhanced Interior Gateway Routing Protocol (EIGRP) routing process.

Step 6 **autonomous-system** *as-number*

Example:

```
RP/0/RSP0/CPU0:router(config-eigrp-vrf-af)# autonomous-system 6
```

Configures the EIGRP routing process to run within a VRF.

Step 7 **default-metric** *bandwidth delay reliability loading mtu*

Example:

```
RP/0/RSP0/CPU0:router(config-eigrp-vrf-af)# default-metric 100000 4000 200 45 4470
```

Sets the metrics for an EIGRP.

Step 8 **redistribute** { { **bgp** | **connected** | **isis** | **ospf** | **rip** | **static** } [*as-number* | *instance-name*] } [**route-policy** *name*]

Example:

```
RP/0/RSP0/CPU0:router(config-eigrp-vrf-af)# redistribute connected
```

Causes connected routes to be redistributed into EIGRP.

Step 9 **interface** *type interface-path-id*

Example:

```
RP/0/RSP0/CPU0:router(config-eigrp-vrf-af)# interface TenGigE 0/3/0/0
```

Associates interface TenGigE 0/3/0/0 with the EIGRP routing process.

Step 10 **site-of-origin** { *as-number:number* | *ip-address : number* }

Example:

```
RP/0/RSP0/CPU0:router(config-eigrp-vrf-af-if)# site-of-origin 201:1
```

Configures site of origin (SoO) on interface TenGigE 0/3/0/0.

Step 11 Use the **commit** or **end** command.

commit - Saves the configuration changes and remains within the configuration session.

end - Prompts user to take one of these actions:

- **Yes** - Saves configuration changes and exits the configuration session.
- **No** - Exits the configuration session without committing the configuration changes.
- **Cancel** - Remains in the configuration mode, without committing the configuration changes.

Configuring EIGRP Redistribution in the MPLS VPN

Perform this task for every provider edge (PE) router that provides VPN services to enable Enhanced Interior Gateway Routing Protocol (EIGRP) redistribution in the MPLS VPN.

Before you begin

The metric can be configured in the route-policy configuring using the **redistribute** command (or configured with the **default-metric** command). If an external route is received from another EIGRP autonomous system or a non-EIGRP network without a configured metric, the route is not installed in the EIGRP database. If an external route is received from another EIGRP autonomous system or a non-EIGRP network without a configured metric, the route is not advertised to the CE router. See the *Implementing EIGRP*module in the *Routing Configuration Guide for Cisco ASR 9000 Series Routers*.

Restriction Redistribution between native EIGRP VPN routing and forwarding (VRF) instances is not supported. This behavior is designed.

SUMMARY STEPS

1. **configure**
2. **router eigrp** *as-number*
3. **vrf** *vrf-name*
4. **address-family ipv4**
5. **redistribute bgp** [*as-number*] [**route-policy** *policy-name*]
6. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

```
RP/0/RSP0/CPU0:router# configure
```

Enters Global Configuration mode.

Step 2 **router eigrp** *as-number*

Example:

```
RP/0/RSP0/CPU0:router(config)# router eigrp 24
```

Enters EIGRP configuration mode allowing you to configure the EIGRP routing process.

Step 3 **vrf** *vrf-name*

Example:

```
RP/0/RSP0/CPU0:router(config-eigrp)# vrf vrf_1
```

Configures a VRF instance and enters VRF configuration mode for EIGRP routing.

Step 4 **address-family ipv4**

Example:

```
RP/0/RSP0/CPU0:router(config-eigrp-vrf)# address family ipv4
```

Enters VRF address family configuration mode for the IPv4 address family.

Step 5 **redistribute bgp** [*as-number*] [**route-policy** *policy-name*]

Example:

```
RP/0/RSP0/CPU0:router(config-eigrp-vrf-af)# redistribute bgp 24 route-policy policy_A
```

Causes Border Gateway Protocol (BGP) routes to be redistributed into EIGRP.

Step 6 Use the **commit** or **end** command.

commit - Saves the configuration changes and remains within the configuration session.

end - Prompts user to take one of these actions:

- **Yes** - Saves configuration changes and exits the configuration session.
- **No** - Exits the configuration session without committing the configuration changes.
- **Cancel** - Remains in the configuration mode, without committing the configuration changes.

Providing VPN Connectivity Across Multiple Autonomous Systems with MPLS VPN Inter-AS with ASBRs Exchanging IPv4 Routes and MPLS Labels

Note This section is not applicable to Inter-AS over IP tunnels.

This section contains instructions for the following tasks:

Configuring ASBRs to Exchange IPv4 Routes and MPLS Labels

Perform this task to configure the autonomous system boundary routers (ASBRs) to exchange IPv4 routes and MPLS labels.

SUMMARY STEPS

1. **configure**
2. **router bgp** *autonomous-system-number*
3. **address-family ipv4 unicast**
4. **allocate-label all**
5. **neighbor** *ip-address*
6. **remote-as** *autonomous-system-number*
7. **address-family ipv4 labeled-unicast**
8. **route-policy** *route-policy-name* **in**
9. **route-policy** *route-policy-name* **out**
10. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

```
RP/0/RSP0/CPU0:router# configure
```

Enters Global Configuration mode.

Step 2 **router bgp** *autonomous-system-number*

Example:

```
RP/0/RSP0/CPU0:router(config)# router bgp 120
RP/0/RSP0/CPU0:router(config-bgp)#
```

Enters Border Gateway Protocol (BGP) configuration mode allowing you to configure the BGP routing process.

Step 3 **address-family ipv4 unicast**

Example:

```
RP/0/RSP0/CPU0:router(config-bgp)# address-family ipv4 unicast
RP/0/RSP0/CPU0:router(config-bgp-af)#
```

Enters global address family configuration mode for the IPv4 unicast address family.

Step 4 **allocate-label all**

Example:

```
RP/0/CPU0:router(config-bgp-af)# allocate-label all
```

Allocates the MPLS labels for a specific IPv4 unicast or VPN routing and forwarding (VRF) IPv4 unicast routes so that the BGP router can send labels with BGP routes to a neighboring router that is configured for a labeled-unicast session.

Step 5 **neighbor** *ip-address*

Example:

```
RP/0/RSP0/CPU0:router(config-bgp-af)# neighbor 172.168.40.24

RP/0/RSP0/CPU0:router(config-bgp-nbr)#
```

Places the router in neighbor configuration mode for BGP routing and configures the neighbor IP address 172.168.40.24 as a BGP peer.

Step 6 **remote-as** *autonomous-system-number*

Example:

```
RP/0/RSP0/CPU0:router(config-bgp-nbr)# remote-as 2002
```

Creates a neighbor and assigns it a remote autonomous system number.

Step 7 **address-family ipv4 labeled-unicast**

Example:

```
RP/0/RSP0/CPU0:router(config-bgp-nbr)# address-family ipv4 labeled-unicast
RP/0/RSP0/CPU0:router(config-bgp-nbr-af)
```

Enters neighbor address family configuration mode for the IPv4 labeled-unicast address family.

Step 8 **route-policy** *route-policy-name* **in**

Example:

```
RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# route-policy pass-all in
```

Applies a routing policy to updates that are received from a BGP neighbor.

- Use the *route-policy-name* argument to define the name of the of route policy. The example shows that the route policy name is defined as pass-all.
- Use the **in** keyword to define the policy for inbound routes.

Step 9 **route-policy** *route-policy-name* **out**

Example:

```
RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# route-policy pass-all out
```

Applies a routing policy to updates that are sent to a BGP neighbor.

- Use the *route-policy-name* argument to define the name of the of route policy. The example shows that the route policy name is defined as pass-all.
- Use the **out** keyword to define the policy for outbound routes.

Step 10 Use the **commit** or **end** command.

commit - Saves the configuration changes and remains within the configuration session.

end - Prompts user to take one of these actions:

- **Yes** - Saves configuration changes and exits the configuration session.
- **No** - Exits the configuration session without committing the configuration changes.
- **Cancel** - Remains in the configuration mode, without committing the configuration changes.

Configuring the Route Reflectors to Exchange VPN-IPv4 Routes

Perform this task to enable the route reflectors to exchange VPN-IPv4 routes by using multihop. This task specifies that the next-hop information and the VPN label are to be preserved across the autonomous system.

SUMMARY STEPS

1. **configure**
2. **router bgp** *autonomous-system-number*
3. **neighbor** *ip-address*
4. **remote-as** *autonomous-system-number*
5. **ebgp-multihop** [*ttl-value*]
6. **update-source** *type interface-path-id*
7. **address-family vpnv4 unicast**
8. **route-policy** *route-policy-name* **in**
9. **route-policy** *route-policy-name* **out**
10. **next-hop-unchanged**
11. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

```
RP/0/RSP0/CPU0:router# configure
```

Enters Global Configuration mode.

Step 2 **router bgp** *autonomous-system-number*

Example:

```
RP/0/RSP0/CPU0:router(config)# router bgp 120
RP/0/RSP0/CPU0:router(config-bgp)#
```

Enters Border Gateway Protocol (BGP) configuration mode allowing you to configure the BGP routing process.

Step 3 **neighbor** *ip-address*

Example:

```
RP/0/RSP0/CPU0:router(config-bgp)# neighbor 172.168.40.24
RP/0/RSP0/CPU0:router(config-bgp-nbr)#
```

Places the router in neighbor configuration mode for BGP routing and configures the neighbor IP address 172.168.40.24 as a BGP peer.

Step 4 **remote-as** *autonomous-system-number*

Example:

```
RP/0/RSP0/CPU0:router(config-bgp-nbr)# remote-as 2002
```

Creates a neighbor and assigns it a remote autonomous system number.

Step 5 **ebgp-multihop** [*ttl-value*]

Example:

```
RP/0/RSP0/CPU0:router(config-bgp-nbr)# ebgp-multihop
```

Enables multihop peerings with external BGP neighbors.

Step 6 **update-source** *type interface-path-id*

Example:

```
RP/0/RSP0/CPU0:router(config-bgp-nbr)# update-source loopback0
```

Allows BGP sessions to use the primary IP address from a particular interface as the local address.

Step 7 **address-family vpnv4 unicast**

Example:

```
RP/0/RSP0/CPU0:router(config-bgp-nbr)# address-family vpnv4 unicast
RP/0/RSP0/CPU0:router(config-bgp-nbr-af)#
```

Configures VPNv4 address family.

Step 8 **route-policy** *route-policy-name* **in**

Example:

```
RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# route-policy pass-all in
```

Applies a routing policy to updates that are received from a BGP neighbor.

- Use the *route-policy-name* argument to define the name of the of route policy. The example shows that the route policy name is defined as pass-all.
- Use the **in** keyword to define the policy for inbound routes.

Step 9 **route-policy** *route-policy-name* **out**

Example:

```
RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# route-policy pass-all out
```

Applies a routing policy to updates that are sent to a BGP neighbor.

- Use the *route-policy-name* argument to define the name of the of route policy. The example shows that the route policy name is defined as pass-all.
- Use the **out** keyword to define the policy for outbound routes.

Step 10 **next-hop-unchanged**

Example:

```
RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# next-hop-unchanged
```

Disables overwriting of the next hop before advertising to external Border Gateway Protocol (eBGP) peers.

Step 11 Use the **commit** or **end** command.

commit - Saves the configuration changes and remains within the configuration session.

end - Prompts user to take one of these actions:

- **Yes** - Saves configuration changes and exits the configuration session.
- **No** - Exits the configuration session without committing the configuration changes.
- **Cancel** - Remains in the configuration mode, without committing the configuration changes.

Configuring the Route Reflector to Reflect Remote Routes in its AS

Perform this task to enable the route reflector (RR) to reflect the IPv4 routes and labels learned by the autonomous system boundary router (ASBR) to the provider edge (PE) routers in the autonomous system. This task is accomplished by making the ASBR and PE route reflector clients of the RR.

SUMMARY STEPS

1. **configure**
2. **router bgp** *autonomous-system-number*
3. **address-family ipv4 unicast**
4. **allocate-label all**
5. **neighbor** *ip-address*
6. **remote-as** *autonomous-system-number*
7. **update-source** *type interface-path-id*
8. **address-family ipv4 labeled-unicast**
9. **route-reflector-client**
10. **neighbor** *ip-address*
11. **remote-as** *autonomous-system-number*
12. **update-source** *type interface-path-id*
13. **address-family ipv4 labeled-unicast**
14. **route-reflector-client**
15. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

```
RP/0/RSP0/CPU0:router# configure
```

Enters Global Configuration mode.

Step 2 **router bgp** *autonomous-system-number*

Example:

```
RP/0/RSP0/CPU0:router(config)# router bgp 120
```

Enters Border Gateway Protocol (BGP) configuration mode allowing you to configure the BGP routing process.

Step 3 **address-family ipv4 unicast**

Example:

```
RP/0/RSP0/CPU0:router(config-bgp)# address-family ipv4 unicast
RP/0/RSP0/CPU0:router(config-bgp-af)#
```

Enters global address family configuration mode for the IPv4 unicast address family.

Step 4 **allocate-label all**

Example:

```
RP/0/RSP0/CPU0:router(config-bgp-af)# allocate-label all
```

Allocates the MPLS labels for a specific IPv4 unicast or VPN routing and forwarding (VRF) IPv4 unicast routes so that the BGP router can send labels with BGP routes to a neighboring router that is configured for a labeled-unicast session.

Step 5 **neighbor** *ip-address*

Example:

```
RP/0/RSP0/CPU0:router(config-bgp-af)# neighbor 172.168.40.24
RP/0/RSP0/CPU0:router(config-bgp-nbr)#
```

Places the router in neighbor configuration mode for BGP routing and configures the neighbor IP address 172.168.40.24 as an ASBR eBGP peer.

Step 6 **remote-as** *autonomous-system-number*

Example:

```
RP/0/RSP0/CPU0:router(config-bgp-nbr)# remote-as 2002
```

Creates a neighbor and assigns it a remote autonomous system number.

Step 7 **update-source** *type interface-path-id*

Example:

```
RP/0/RSP0/CPU0:router(config-bgp-nbr)# update-source loopback0
```

Allows BGP sessions to use the primary IP address from a particular interface as the local address.

Step 8 **address-family ipv4 labeled-unicast**

Example:

```
RP/0/RSP0/CPU0:router(config-bgp-nbr)# address-family ipv4 labeled-unicast
RP/0/RSP0/CPU0:router(config-bgp-nbr-af)#
```

Enters neighbor address family configuration mode for the IPv4 labeled-unicast address family.

Step 9 **route-reflector-client**

Example:

```
RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# route-reflector-client
```

Configures the router as a BGP route reflector and neighbor 172.168.40.24 as its client.

Step 10 **neighbor** *ip-address*

Example:

```
RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# neighbor 10.40.25.2
RP/0/RSP0/CPU0:router(config-bgp-nbr)#
```

Places the router in neighbor configuration mode for BGP routing and configures the neighbor IP address .40.25.2 as an VPNv4 iBGP peer.

Step 11 **remote-as** *autonomous-system-number*

Example:

```
RP/0/RSP0/CPU0:router(config-bgp-nbr)# remote-as 2002
```

Creates a neighbor and assigns it a remote autonomous system number.

Step 12 **update-source** *type interface-path-id*

Example:

```
RP/0/RSP0/CPU0:router(config-bgp-nbr)# update-source loopback0
```

Allows BGP sessions to use the primary IP address from a particular interface as the local address.

Step 13 **address-family ipv4 labeled-unicast**

Example:

```
RP/0/RSP0/CPU0:router(config-bgp-nbr)# address-family ipv4 labeled-unicast
RP/0/RSP0/CPU0:router(config-bgp-nbr-af)#
```

Enters neighbor address family configuration mode for the IPv4 labeled-unicast address family.

Step 14 **route-reflector-client**

Example:

```
RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# route-reflector-client
```

Configures the neighbor as a route reflector client.

Step 15 Use the **commit** or **end** command.

commit - Saves the configuration changes and remains within the configuration session.

end - Prompts user to take one of these actions:

- **Yes** - Saves configuration changes and exits the configuration session.
- **No** - Exits the configuration session without committing the configuration changes.
- **Cancel** - Remains in the configuration mode, without committing the configuration changes.

Providing VPN Connectivity Across Multiple Autonomous Systems with MPLS VPN Inter-AS with ASBRs Exchanging VPN-IPv4 Addresses

This section contains instructions for the following tasks:

Configuring the ASBRs to Exchange VPN-IPv4 Addresses for IP Tunnels

Perform this task to configure an external Border Gateway Protocol (eBGP) autonomous system boundary router (ASBR) to exchange VPN-IPv4 routes with another autonomous system.

SUMMARY STEPS

1. **configure**
2. **router bgp** *autonomous-system-number*
3. **address-family** { **ipv4 tunnel** }
4. **address-family** { **vpnv4 unicast** }
5. **neighbor** *ip-address*
6. **remote-as** *autonomous-system-number*
7. **address-family** { **vpnv4 unicast** }
8. **route-policy** *route-policy-name* { **in** }
9. **route-policy** *route-policy-name* { **out** }
10. **neighbor** *ip-address*
11. **remote-as** *autonomous-system-number*
12. **update-source** *type interface-path-id*
13. **address-family** { **ipv4 tunnel** }
14. **address-family** { **vpnv4 unicast** }
15. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

```
RP/0/RSP0/CPU0:router# configure
```

Enters the Global Configuration mode.

Step 2 **router bgp** *autonomous-system-number*

Example:

```
RP/0/RSP0/CPU0:router(config)# router bgp 120
RP/0/RSP0/CPU0:router(config-bgp)#
```

Enters Border Gateway Protocol (BGP) configuration mode allowing you to configure the BGP routing process.

Step 3 **address-family** { **ipv4 tunnel** }

Example:

```
RP/0/RSP0/CPU0:router(config-bgp)# address-family ipv4 tunnel
RP/0/RSP0/CPU0:router(config-bgp-af)#
```

Configures IPv4 tunnel address family.

Step 4 **address-family** { **vpnv4 unicast** }

Example:

```
RP/0/RSP0/CPU0:router(config-bgp-af)# address-family vpnv4 unicast
```

Configures VPNv4 address family.

Step 5 **neighbor** *ip-address*

Example:

```
RP/0/RSP0/CPU0:router(config-bgp-af)# neighbor 172.168.40.24
RP/0/RSP0/CPU0:router(config-bgp-nbr)#
```

Places the router in neighbor configuration mode for BGP routing and configures the neighbor IP address 172.168.40.24 as an ASBR eBGP peer.

Step 6 **remote-as** *autonomous-system-number*

Example:

```
RP/0/RSP0/CPU0:router(config-bgp-nbr)# remote-as 2002
```

Creates a neighbor and assigns it a remote autonomous system number.

Step 7 **address-family** { **vpnv4 unicast** }

Example:

```
RP/0/RSP0/CPU0:router(config-bgp-nbr)# address-family vpnv4 unicast
RP/0/RSP0/CPU0:router(config-bgp-nbr-af)#
```

Configures VPNv4 address family.

Step 8 **route-policy** *route-policy-name* { **in** }

Example:

```
RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# route-policy pass-all in
```

Applies a routing policy to updates that are received from a BGP neighbor.

- Use the *route-policy-name* argument to define the name of the of route policy. The example shows that the route policy name is defined as pass-all.
- Use the **in** keyword to define the policy for inbound routes.

Step 9 **route-policy** *route-policy-name* { **out** }

Example:

```
RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# route-policy pass-all out
```

Applies a routing policy to updates that are sent from a BGP neighbor.

- Use the *route-policy-name* argument to define the name of the route policy. The example shows that the route policy name is defined as pass-all.
- Use the **out** keyword to define the policy for outbound routes.

Step 10 **neighbor** *ip-address*

Example:

```
RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# neighbor 175.40.25.2
RP/0/RSP0/CPU0:router(config-bgp-nbr)#
```

Places the router in neighbor configuration mode for BGP routing and configures the neighbor IP address 175.40.25.2 as an VPNv4 iBGP peer.

Step 11 **remote-as** *autonomous-system-number*

Example:

```
RP/0/RSP0/CPU0:router(config-bgp-nbr)# remote-as 2002
```

Creates a neighbor and assigns it a remote autonomous system number.

Step 12 **update-source** *type interface-path-id*

Example:

```
RP/0/RSP0/CPU0:router(config-bgp-nbr)# update-source loopback0
```

Allows BGP sessions to use the primary IP address from a particular interface as the local address.

Step 13 **address-family** { **ipv4 tunnel** }

Example:

```
RP/0/RSP0/CPU0:router(config-bgp-nbr)# address-family ipv4 tunnel
RP/0/RSP0/CPU0:router(config-bgp-nbr-af)#
```

Configures IPv4 tunnel address family.

Step 14 **address-family** { **vpnv4 unicast** }

Example:

```
RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# address-family vpnv4 unicast
```

Configures VPNv4 address family.

Step 15 Use the **commit** or **end** command.

commit - Saves the configuration changes and remains within the configuration session.

end - Prompts user to take one of these actions:

- **Yes** - Saves configuration changes and exits the configuration session.
- **No** - Exits the configuration session without committing the configuration changes.
- **Cancel** - Remains in the configuration mode, without committing the configuration changes.

Configuring a Static Route to an ASBR Peer

Perform this task to configure a static route to an ASBR peer.

SUMMARY STEPS

1. **configure**
2. **router static**

3. **address-family ipv4 unicast**
4. **A.B.C.D/length** *next-hop*
5. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

```
RP/0/RSP0/CPU0:router# configure
```

Enters the Global Configuration mode.

Step 2 **router static**

Example:

```
RP/0/RSP0/CPU0:router(config)# router static
RP/0/RSP0/CPU0:router(config-static)#
```

Enters router static configuration mode.

Step 3 **address-family ipv4 unicast**

Example:

```
RP/0/RSP0/CPU0:router(config-static)# address-family ipv4 unicast
RP/0/RSP0/CPU0:router(config-static-afi)#
```

Enables an IPv4 address family.

Step 4 **A.B.C.D/length** *next-hop*

Example:

```
RP/0/RSP0/CPU0:router(config-static-afi)# 10.10.10.10/32 10.9.9.9
```

Enters the address of the destination router (including IPv4 subnet mask).

Step 5 Use the **commit** or **end** command.

commit - Saves the configuration changes and remains within the configuration session.

end - Prompts user to take one of these actions:

- **Yes** - Saves configuration changes and exits the configuration session.
- **No** - Exits the configuration session without committing the configuration changes.
- **Cancel** - Remains in the configuration mode, without committing the configuration changes.

Configuring EBGP Routing to Exchange VPN Routes Between Subautonomous Systems in a Confederation

Perform this task to configure external Border Gateway Protocol (eBGP) routing to exchange VPN routes between subautonomous systems in a confederation.

Note To ensure that host routes for VPN-IPv4 eBGP neighbors are propagated (by means of the Interior Gateway Protocol [IGP]) to other routers and PE routers, specify the **redistribute connected** command in the IGP configuration portion of the confederation eBGP (CEBGP) router. If you are using Open Shortest Path First (OSPF), make sure that the OSPF process is not enabled on the CEBGP interface in which the "redistribute connected" subnet exists.

SUMMARY STEPS

1. **configure**
2. **router bgp** *autonomous-system-number*
3. **bgp confederation peers** *peer autonomous-system-number*
4. **bgp confederation identifier** *autonomous-system-number*
5. **address-family vpnv4 unicast**
6. **neighbor** *ip-address*
7. **remote-as** *autonomous-system-number*
8. **address-family vpnv4 unicast**
9. **route-policy** *route-policy-name* **in**
10. **route-policy** *route-policy-name* **out**
11. **next-hop-self**
12. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

```
RP/0/RSP0/CPU0:router# configure
```

Enters Global Configuration mode.

Step 2 **router bgp** *autonomous-system-number*

Example:

```
RP/0/RSP0/CPU0:router(config)# router bgp 120
RP/0/RSP0/CPU0:router(config-bgp)#
```

Enters BGP configuration mode allowing you to configure the BGP routing process.

Step 3 **bgp confederation peers** *peer autonomous-system-number*

Example:

```
RP/0/RSP0/CPU0:router(config-bgp)# bgp confederation peers 8
```

Configures the peer autonomous system number that belongs to the confederation.

Step 4 **bgp confederation identifier** *autonomous-system-number*

Example:

```
RP/0/RSP0/CPU0:router(config-bgp)# bgp confederation identifier 5
```

Specifies the autonomous system number for the confederation ID.

Step 5 **address-family vpnv4 unicast**

Example:

```
RP/0/RSP0/CPU0:router(config-bgp)# address-family vpnv4 unicast
RP/0/RSP0/CPU0:router(config-bgp-af)#
```

Configures VPNv4 address family.

Step 6 **neighbor** *ip-address*

Example:

```
RP/0/RSP0/CPU0:router(config-bgp-af)# neighbor 10.168.40.24
RP/0/RSP0/CPU0:router(config-bgp-nbr)#
```

Places the router in neighbor configuration mode for BGP routing and configures the neighbor IP address 10.168.40.24 as a BGP peer.

Step 7 **remote-as** *autonomous-system-number*

Example:

```
RP/0/RSP0/CPU0:router(config-bgp-nbr)# remote-as 2002
```

Creates a neighbor and assigns it a remote autonomous system number.

Step 8 **address-family vpnv4 unicast**

Example:

```
RP/0/RSP0/CPU0:router(config-bgp-nbr)# address-family vpnv4 unicast
RP/0/RSP0/CPU0:router(config-bgp-nbr-af)#
```

Configures VPNv4 address family.

Step 9 **route-policy** *route-policy-name* **in**

Example:

```
RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# route-policy In-Ipv4 in
```

Applies a routing policy to updates received from a BGP neighbor.

Step 10 **route-policy** *route-policy-name* **out**

Example:

```
RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# route-policy Out-Ipv4 out
```

Applies a routing policy to updates advertised to a BGP neighbor.

Step 11 **next-hop-self**

Example:

```
RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# next-hop-self
```

Disables next-hop calculation and let you insert your own address in the next-hop field of BGP updates.

Step 12 Use the **commit** or **end** command.

commit - Saves the configuration changes and remains within the configuration session.

end - Prompts user to take one of these actions:

- **Yes** - Saves configuration changes and exits the configuration session.
- **No** - Exits the configuration session without committing the configuration changes.
- **Cancel** - Remains in the configuration mode, without committing the configuration changes.

Configuring MPLS Forwarding for ASBR Confederations

Perform this task to configure MPLS forwarding for autonomous system boundary router (ASBR) confederations (in BGP) on a specified interface.

Note This configuration adds the implicit NULL rewrite corresponding to the peer associated with the interface, which is required to prevent BGP from automatically installing rewrites by LDP (in multihop instances).

SUMMARY STEPS

1. **configure**
2. **router bgp** *as-number*
3. **mpls activate**
4. **interface** *type interface-path-id*
5. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

```
RP/0/RSP0/CPU0:router# configure
```

Enters Global Configuration mode.

Step 2 **router bgp** *as-number*

Example:

```
RP/0/RSP0/CPU0:router(config)# router bgp 120
RP/0/RSP0/CPU0:router(config-bgp)
```

Enters BGP configuration mode allowing you to configure the BGP routing process.

Step 3 **mpls activate**

Example:

```
RP/0/RSP0/CPU0:router(config-bgp)# mpls activate
RP/0/RSP0/CPU0:router(config-bgp-mpls)#
```

Enters BGP MPLS activate configuration mode.

Step 4 **interface** *type interface-path-id*

Example:

```
RP/0/RSP0/CPU0:router(config-bgp-mpls)# interface GigabitEthernet 0/3/0/0
```

Enables MPLS on the interface.

Step 5 Use the **commit** or **end** command.

commit - Saves the configuration changes and remains within the configuration session.

end - Prompts user to take one of these actions:

- **Yes** - Saves configuration changes and exits the configuration session.
- **No** - Exits the configuration session without committing the configuration changes.
- **Cancel** - Remains in the configuration mode, without committing the configuration changes.

Configuring a Static Route to an ASBR Confederation Peer

Perform this task to configure a static route to an Inter-AS confederation peer. For more detailed information, see “Configuring a Static Route to a Peer" section.

SUMMARY STEPS

1. **configure**
2. **router static**
3. **address-family ipv4 unicast**
4. **A.B.C.D/length** *next-hop*
5. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

```
RP/0/RSP0/CPU0:router# configure
```

Enters Global Configuration mode.

Step 2 **router static**

Example:

```
RP/0/RSP0/CPU0:router(config)# router static
RP/0/RSP0/CPU0:router(config-static)#
```

Enters router static configuration mode.

Step 3 **address-family ipv4 unicast**

Example:

```
RP/0/RSP0/CPU0:router(config-static)# address-family ipv4 unicast
RP/0/RSP0/CPU0:router(config-static-afi)#
```

Enables an IPv4 address family.

Step 4 **A.B.C.D/length** *next-hop*

Example:

```
RP/0/RSP0/CPU0:router(config-static-afi)# 10.10.10.10/32 10.9.9.9
```

Enters the address of the destination router (including IPv4 subnet mask).

Step 5 Use the **commit** or **end** command.

commit - Saves the configuration changes and remains within the configuration session.

end - Prompts user to take one of these actions:

- **Yes** - Saves configuration changes and exits the configuration session.
- **No** - Exits the configuration session without committing the configuration changes.

- **Cancel** - Remains in the configuration mode, without committing the configuration changes.

Configuring Carrier Supporting Carrier

Perform the tasks in this section to configure Carrier Supporting Carrier (CSC):

Identifying the Carrier Supporting Carrier Topology

Before you configure the MPLS VPN CSC with BGP, you must identify both the backbone and customer carrier topology.

Note You can connect multiple CSC-CE routers to the same PE, or you can connect a single CSC-CE router to multiple CSC-PEs using more than one CSC-CE interface to provide redundancy and multiple path support in a CSC topology.

Perform this task to identify the carrier supporting carrier topology.

SUMMARY STEPS

1. Identify the type of customer carrier, ISP, or MPLS VPN service provider.
2. Identify the CE routers.
3. Identify the customer carrier core router configuration.
4. Identify the customer carrier edge (CSC-CE) routers.
5. Identify the backbone carrier router configuration.

DETAILED STEPS

Step 1 Identify the type of customer carrier, ISP, or MPLS VPN service provider.

Sets up requirements for configuration of carrier supporting carrier network.

Step 2 Identify the CE routers.

Sets up requirements for configuration of CE to PE connections.

Step 3 Identify the customer carrier core router configuration.

Sets up requirements for configuration between core (P) routers and between P routers and edge routers (PE and CSC-CE routers).

Step 4 Identify the customer carrier edge (CSC-CE) routers.

Sets up requirements for configuration of CSC-CE to CSC-PE connections.

Step 5 Identify the backbone carrier router configuration.

Sets up requirements for configuration between CSC core routers and between CSC core routers and edge routers (CSC-CE and CSC-PE routers).

Configuring the Backbone Carrier Core

Configuring the backbone carrier core requires setting up connectivity and routing functions for the CSC core and the CSC-PE routers. To do so, you must complete the following high-level tasks:

- Verify IP connectivity in the CSC core.
- Verify LDP configuration in the CSC core.

Note This task is not applicable to CSC over IP tunnels.

- Configure VRFs for CSC-PE routers.
- Configure multiprotocol BGP for VPN connectivity in the backbone carrier.

Configuring the CSC-PE and CSC-CE Routers

Perform the following tasks to configure links between a CSC-PE router and the carrier CSC-CE router for an MPLS VPN CSC network that uses BGP to distribute routes and MPLS labels:

The following figure shows the configuration for the peering with directly connected interfaces between CSC-PE and CSC-CE routers. This configuration is used as the example in the tasks that follow.

Figure 7: Configuration for Peering with Directly Connected Interfaces Between CSC-PE and CSC-CE Routers

Configuring a Static Route to a Peer

Perform this task to configure a static route to an Inter-AS or CSC-CE peer.

When you configure an Inter-AS or CSC peer, BGP allocates a label for a /32 route to that peer and performs a NULL label rewrite. When forwarding a labeled packet to the peer, the router removes the top label from the label stack; however, in such an instance, BGP expects a /32 route to the peer. This task ensures that there is, in fact, a /32 route to the peer.

Please be aware of the following facts before performing this task:

- A /32 route is not required to establish BGP peering. A route using a shorter prefix length will also work.
- A shorter prefix length route is not associated with the allocated label; even though the BGP session comes up between the peers, without the static route, forwarding will not work.

Note To configure a static route on a CSC-PE, you must configure the router under the VRF (as noted in the detailed steps).

SUMMARY STEPS

1. **configure**
2. **router static**
3. **address-family ipv4 unicast**
4. **A.B.C.D/length** *next-hop*
5. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

```
RP/0/RSP0/CPU0:router# configure
```

Enters the Global Configuration mode.

Step 2 **router static**

Example:

```
RP/0/RSP0/CPU0:router(config)# router static
```

Enters router static configuration mode.

Step 3 **address-family ipv4 unicast**

Example:

```
RP/0/RSP0/CPU0:router(config-static)# address-family ipv4 unicast
```

Enables an IPv4 address family.

Note To configure a static route on a CSC-PE, you must first configure the VRF using the **vrf** command before **address-family**.

Step 4 **A.B.C.D/length** *next-hop*

Example:

```
RP/0/RSP0/CPU0:router(config-static-afi)# 10.10.10.10/32 10.9.9.9
```

Enters the address of the destination router (including IPv4 subnet mask).

Step 5 Use the **commit** or **end** command.

commit - Saves the configuration changes and remains within the configuration session.

end - Prompts user to take one of these actions:

- **Yes** - Saves configuration changes and exits the configuration session.
- **No** - Exits the configuration session without committing the configuration changes.

- **Cancel** - Remains in the configuration mode, without committing the configuration changes.

Verifying the MPLS Layer 3 VPN Configuration

Perform this task to verify the MPLS Layer 3 VPN configuration.

SUMMARY STEPS

1. **show running-config router bgp** *as-number* **vrf** *vrf-name*
2. **show running-config routes**
3. **show ospf vrf** *vrf-name* **database**
4. **show running-config router bgp** *as-number* **vrf** *vrf-name* **neighbor** *ip-address*
5. **show bgp vrf** *vrf-name* **summary**
6. **show bgp vrf** *vrf-name* **neighbors** *ip-address*
7. **show bgp vrf** *vrf-name*
8. **show route vrf** *vrf-name ip-address*
9. **show bgp vpn unicast summary**
10. **show running-config router isis**
11. **show running-config mpls**
12. **show isis adjacency**
13. **show mpls ldp forwarding**
14. **show bgp vpnv4 unicast** or **show bgp vrf** *vrf-name*
15. **show bgp vrf** *vrf-name* **imported-routes**
16. **show route vrf** *vrf-name ip-address*
17. **show cef vrf** *vrf-name ip-address*
18. **show cef vrf** *vrf-name ip-address* **location** *node-id*
19. **show bgp vrf** *vrf-name ip-address*
20. **show ospf vrf** *vrf-name* **database**

DETAILED STEPS

Step 1 **show running-config router bgp** *as-number* **vrf** *vrf-name*

Example:

```
RP/0/RSP0/CPU0:router# show running-config router bgp 3 vrf vrf_A
```

Displays the specified VPN routing and forwarding (VRF) content of the currently running configuration.

Step 2 **show running-config routes**

Example:

```
RP/0/RSP0/CPU0:router# show running-config routes
```

Displays the Open Shortest Path First (OSPF) routes table in the currently running configuration.

Step 3 **show ospf vrf** *vrf-name* **database**

Example:

```
RP/0/RSP0/CPU0:router# show ospf vrf vrf_A database
```

Displays lists of information related to the OSPF database for a specified VRF.

Step 4 **show running-config router bgp** *as-number* **vrf** *vrf-name* **neighbor** *ip-address*

Example:

```
RP/0/RSP0/CPU0:router# show running-config router bgp 3 vrf vrf_A neighbor 172.168.40.24
```

Displays the Border Gateway Protocol (BGP) VRF neighbor content of the currently running configuration.

Step 5 **show bgp vrf** *vrf-name* **summary**

Example:

```
RP/0/RSP0/CPU0:router# show bgp vrf vrf_A summary
```

Displays the status of the specified BGP VRF connections.

Step 6 **show bgp vrf** *vrf-name* **neighbors** *ip-address*

Example:

```
RP/0/RSP0/CPU0:router# show bgp vrf vrf_A neighbors 172.168.40.24
```

Displays information about BGP VRF connections to the specified neighbors.

Step 7 **show bgp vrf** *vrf-name*

Example:

```
RP/0/RSP0/CPU0:router# show bgp vrf vrf_A
```

Displays information about a specified BGP VRF.

Step 8 **show route vrf** *vrf-name ip-address*

Example:

```
RP/0/RSP0/CPU0:router# show route vrf vrf_A 10.0.0.0
```

Displays the current routes in the Routing Information Base (RIB) for a specified VRF.

Step 9 **show bgp vpn unicast summary**

Example:

```
RP/0/RSP0/CPU0:router# show bgp vpn unicast summary
```

Displays the status of all BGP VPN unicast connections.

Step 10 **show running-config router isis**

Example:

```
RP/0/RSP0/CPU0:router# show running-config router isis
```

Displays the Intermediate System-to-Intermediate System (IS-IS) content of the currently running configuration.

Step 11 **show running-config mpls**

Example:

```
RP/0/RSP0/CPU0:router# show running-config mpls
```

Displays the MPLS content of the currently running-configuration.

Step 12 **show isis adjacency**

Example:

```
RP/0/RSP0/CPU0:router# show isis adjacency
```

Displays IS-IS adjacency information.

Step 13 **show mpls ldp forwarding**

Example:

```
RP/0/RSP0/CPU0:router# show mpls ldp forwarding
```

Displays the Label Distribution Protocol (LDP) forwarding state installed in MPLS forwarding.

Step 14 **show bgp vpnv4 unicast** or **show bgp vrf** *vrf-name*

Example:

```
RP/0/RSP0/CPU0:router# show bgp vpnv4 unicast
```

Displays entries in the BGP routing table for VPNv4 or VPNv6 unicast addresses.

Step 15 **show bgp vrf** *vrf-name* **imported-routes**

Example:

```
RP/0/RSP0/CPU0:router# show bgp vrf vrf_A imported-routes
```

Displays BGP information for routes imported into specified VRF instances.

Step 16 **show route vrf** *vrf-name ip-address*

Example:

```
RP/0/RSP0/CPU0:router# show route vrf vrf_A 10.0.0.0
```

Displays the current specified VRF routes in the RIB.

Step 17 **show cef vrf** *vrf-name ip-address*

Example:

```
RP/0/RSP0/CPU0:router# show cef vrf vrf_A 10.0.0.1
```

Displays the IPv4 Cisco Express Forwarding (CEF) table for a specified VRF.

Step 18 **show cef vrf** *vrf-name ip-address* **location** *node-id*

Example:

```
RP/0/RSP0/CPU0:router# show cef vrf vrf_A 10.0.0.1 location 0/1/cpu0
```

Displays the IPv4 CEF table for a specified VRF and location.

Step 19 **show bgp vrf** *vrf-name ip-address*

Example:

```
RP/0/RSP0/CPU0:router# show bgp vrf vrf_A 10.0.0.0
```

Displays entries in the BGP routing table for VRF vrf_A.

Step 20 **show ospf vrf** *vrf-name* **database**

Example:

```
RP/0/RSP0/CPU0:router# show ospf vrf vrf_A database
```

Displays lists of information related to the OSPF database for a specified VRF.

Configuring L3VPN over GRE

Perform the following tasks to configure L3VPN over GRE:

Creating a GRE Tunnel between Provider Edge Routers

Perform this task to configure a GRE tunnel between provider edge routers.

SUMMARY STEPS

1. **configure**
2. **interface tunnel-ip** *number*
3. **ipv4 address** *ipv4-address subnet-mask*
4. **ipv6 address** *ipv6-prefix/prefix-length*
5. **tunnel mode gre ipv4**
6. **tunnel source** *type path-id*
7. **tunnel destination** *ip-address*
8. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

```
RP/0/RSP0/CPU0:router# configure
```

Enters the Global Configuration mode.

Step 2 **interface tunnel-ip** *number*

Example:

```
RP/0/RSP0/CPU0:router(config)# interface tunnel-ip 4000
```

Enters tunnel interface configuration mode.

- number is the number associated with the tunnel interface.

Step 3 **ipv4 address** *ipv4-address subnet-mask*

Example:

```
RP/0/RSP0/CPU0:router(config-if)# ipv4 address 10.0.0.1 255.0.0.0
```

Specifies the IPv4 address and subnet mask for the interface.

- ipv4-address specifies the IP address of the interface.
- subnet-mask specifies the subnet mask of the interface.

Step 4 **ipv6 address** *ipv6-prefix/prefix-length*

Example:

```
RP/0/RSP0/CPU0:router(config-if)# ipv6 address 100:1:1:1::1/64
```

Specifies an IPv6 network assigned to the interface.

Step 5 **tunnel mode gre ipv4**

Example:

```
RP/0/RSP0/CPU0:router(config-if)# tunnel mode gre ipv4
```

Sets the encapsulation mode of the tunnel interface to GRE.

Step 6 **tunnel source** *type path-id*

Example:

```
RP/0/RSP0/CPU0:router(config-if)# tunnel source TenGigE0/2/0/1
```

Specifies the source of the tunnel interface.

Step 7 **tunnel destination** *ip-address*

Example:

```
RP/0/RSP0/CPU0:router(config-if)# tunnel destination 145.12.5.2
```

Defines the tunnel destination.

Step 8 Use the **commit** or **end** command.

commit - Saves the configuration changes and remains within the configuration session.

end - Prompts user to take one of these actions:

- **Yes** - Saves configuration changes and exits the configuration session.
- **No** - Exits the configuration session without committing the configuration changes.
- **Cancel** - Remains in the configuration mode, without committing the configuration changes.

Configuring IGP between Provider Edge Routers

Perform this task to configure IGP between provider edge routers.

SUMMARY STEPS

1. **configure**
2. **router ospf** *process-name*
3. **nsr**
4. **router-id** { *router-id* }
5. **mpls ldp sync**
6. **dead-interval** *seconds*
7. **hello-interval** *seconds*
8. **area** *area-id*
9. **interface tunnel-ip** *number*
10. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

```
RP/0/RSP0/CPU0:router# configure
```

Enters the Global Configuration mode.

Step 2 **router ospf** *process-name*

Example:

```
RP/0/RSP0/CPU0:router(config)# router ospf 1
```

Enables OSPF routing for the specified routing process and places the router in router configuration mode.

Step 3 **nsr**

Example:

```
RP/0/RSP0/CPU0:router(config-ospf)# nsr
```

Activates BGP NSR.

Step 4 **router-id** { *router-id* }

Example:

```
RP/0/RSP0/CPU0:router(config-ospf)# router-id 10.0.0.1
```

Configures a router ID for the OSPF process.

Note We recommend using a stable IP address as the router ID.

Step 5 **mpls ldp sync**

Example:

```
RP/0/RSP0/CPU0:router(config-ospf)# mpls ldp sync
```

Enables MPLS LDP synchronization.

Step 6 **dead-interval** *seconds*

Example:

```
RP/0/RSP0/CPU0:router(config-ospf)# dead-interval 60
```

Sets the time to wait for a hello packet from a neighbor before declaring the neighbor down.

Step 7 **hello-interval** *seconds*

Example:

```
RP/0/RSP0/CPU0:router(config-ospf)# hello-interval 15
```

Specifies the interval between hello packets that OSPF sends on the interface.

Step 8 **area** *area-id*

Example:

```
RP/0/RSP0/CPU0:router(config-ospf)# area 0
```

Enters area configuration mode and configures an area for the OSPF process.

Step 9 **interface tunnel-ip** *number*

Example:

```
RP/0/RSP0/CPU0:router(config-ospf)# interface tunnel-ip 4
```

Enters tunnel interface configuration mode.

- number is the number associated with the tunnel interface.

Step 10 Use the **commit** or **end** command.

commit - Saves the configuration changes and remains within the configuration session.

end - Prompts user to take one of these actions:

- **Yes** - Saves configuration changes and exits the configuration session.
- **No** - Exits the configuration session without committing the configuration changes.
- **Cancel** - Remains in the configuration mode, without committing the configuration changes.

Configuring LDP/GRE on the Provider Edge Routers

Perform this task to configure LDP/GRE on the provider edge routers.

SUMMARY STEPS

1. **configure**
2. **mpls ldp**
3. **router-id** { *router-id* }
4. **discovery hello holdtime** *seconds*
5. **discovery hello interval** *seconds*
6. **nsr**
7. **graceful-restart**
8. **graceful-restart reconnect-timeout** *seconds*
9. **graceful-restart forwarding-state-holdtime** *seconds*
10. **holdtime** *seconds*
11. **neighbor** *ip-address*
12. **interface tunnel-ip** *number*
13. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

```
RP/0/RSP0/CPU0:router# configure
```

Enters the Global Configuration mode.

Step 2 **mpls ldp**

Example:

```
RP/0/RSP0/CPU0:router(config)# mpls ldp
```

Enables MPLS LDP configuration mode.

Step 3 **router-id** { *router-id* }

Example:

```
RP/0/RSP0/CPU0:router(config-ldp)# router-id 10.0.0.1
```

Configures a router ID for the OSPF process.

Note We recommend using a stable IP address as the router ID.

Step 4 **discovery hello holdtime** *seconds*

Example:

```
RP/0/RSP0/CPU0:router(config-ldp)# discovery hello holdtime 40
```

Defines the period of time a discovered LDP neighbor is remembered without receipt of an LDP Hello message from the neighbor.

Note We recommend using a stable IP address as the router ID.

Step 5 **discovery hello interval** *seconds*

Example:

```
RP/0/RSP0/CPU0:router(config-ldp)# discovery hello holdtime 20
```

Defines the period of time between the sending of consecutive Hello messages.

Step 6 **nsr**

Example:

```
RP/0/RSP0/CPU0:router(config-ldp)# nsr
```

Activates BGP NSR.

Step 7 **graceful-restart**

Example:

```
RP/0/RSP0/CPU0:router(config-ldp)# graceful-restart
```

Enables graceful restart on the router.

Step 8 **graceful-restart reconnect-timeout** *seconds*

Example:

```
RP/0/RSP0/CPU0:router(config-ldp)# graceful-restart recoonect-timeout 180
```

Defines the time for which the neighbor should wait for a reconnection if the LDP session is lost.

Step 9 **graceful-restart forwarding-state-holdtime** *seconds*

Example:

```
RP/0/RSP0/CPU0:router(config-ldp)# graceful-restart forwarding-state-holdtime 300
```

Defines the time that the neighbor should retain the MPLS forwarding state during a recovery.

Step 10 **holdtime** *seconds*

Example:

```
RP/0/RSP0/CPU0:router(config-ldp)# holdtime 90
```

Configures the hold time for an interface.

Step 11 **neighbor** *ip-address*

Example:

```
RP/0/RSP0/CPU0:router(config-ldp)# neighbor 10.1.1.0
```

Defines a neighboring router.

Step 12 **interface tunnel-ip** *number*

Example:

```
RP/0/RSP0/CPU0:router(config-ldp)# interface tunnel-ip 4
```

Enters tunnel interface configuration mode.

- number is the number associated with the tunnel interface.

Step 13 Use the **commit** or **end** command.

commit - Saves the configuration changes and remains within the configuration session.

end - Prompts user to take one of these actions:

- **Yes** - Saves configuration changes and exits the configuration session.
- **No** - Exits the configuration session without committing the configuration changes.
- **Cancel** - Remains in the configuration mode, without committing the configuration changes.

Configuring L3VPN

Perform this task to configure L3VPN.

SUMMARY STEPS

1. **configure**
2. **vrf** *vrf-name*
3. **address-family { ipv4 | ipv6 } unicast**
4. **import route-target** [*as-number:nn* | *ip-address:nn*]
5. **export route-target** [*as-number:nn* | *ip-address:nn*]
6. **interface** *type interface-path-id*
7. **vrf** *vrf-name*
8. **ipv4 address** *ipv4-address subnet-mask*
9. **dot1q native vlan** *vlan-id*
10. **router bgp** *as-number*
11. **nsr**

12. **bgp router-id** *ip-address*
13. **address-family {vpnv4 | vpnv6} unicast**
14. **neighbor** *ip-address*
15. **remote-as** *as-number*
16. **update-source** *type interface-path-id*
17. **address-family { vpnv4 | vpnv6 } unicast**
18. **route-policy** *route-policy-name* **in**
19. **route-policy** *route-policy-name* **out**
20. **vrf** *vrf-name*
21. **rd { as-number:nn | ip-address:nn | auto }**
22. **address-family { ipv4 | ipv6 } unicast**
23. **redistribute connected** [**metric** *metric-value*] [*route-policy route-policy-name*]
24. **redistribute static** [**metric** *metric-value*] [*route-policy route-policy-name*]
25. **neighbor** *ip-address*
26. **remote-as** *as-number*
27. **ebg-multihop** *ttl-value*
28. **address-family { ipv4 | ipv6 } unicast**
29. **route-policy** *route-policy-name* **in**
30. **route-policy** *route-policy-name* **out**
31. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

```
RP/0/RSP0/CPU0:router# configure
```

Enters the Global Configuration mode.

Step 2 **vrf** *vrf-name*

Example:

```
RP/0/RSP0/CPU0:router(config)# vrf vpn1
```

Configures a VRF instance.

Step 3 **address-family { ipv4 | ipv6 } unicast**

Example:

```
RP/0/RSP0/CPU0:router(config-vrf)# address-family { ipv4 | ipv6 } unicast
```

Specifies either the IPv4 or IPv6 address family and enters address family configuration submode.

Step 4 **import route-target** [*as-number:nn* | *ip-address:nn*]

Example:

```
RP/0/RSP0/CPU0:router(config-vrf)# import route-target 2:1
```

Specifies a list of route target (RT) extended communities. Only prefixes that are associated with the specified import route target extended communities are imported into the VRF.

Step 5 **export route-target** [*as-number:nn* | *ip-address:nn*]

Example:

```
RP/0/RSP0/CPU0:router(config-vrf)# export route-target 1:1
```

Specifies a list of route target extended communities. Export route target communities are associated with prefixes when they are advertised to remote PEs. The remote PEs import them into VRFs which have import RTs that match these exported route target communities.

Step 6 **interface** *type interface-path-id*

Example:

```
RP/0/RSP0/CPU0:router(config)# interface TenGigE0/2/0/0.1
```

Enters interface configuration mode and configures an interface.

Step 7 **vrf** *vrf-name*

Example:

```
RP/0/RSP0/CPU0:router(config-if)# vrf vpn1
```

Configures a VRF instance.

Step 8 **ipv4 address** *ipv4-address subnet-mask*

Example:

```
RP/0/RSP0/CPU0:router(config-if)# ipv4 address 172.16.0.1 255.240.0.0
```

Specifies the IPv4 address and subnet mask for the interface.

- ipv4-address specifies the IP address of the interface.
- subnet-mask specifies the subnet mask of the interface.

Step 9 **dot1q native vlan** *vlan-id*

Example:

```
RP/0/RSP0/CPU0:router(config-if)# dot1q native
vlan 1
```

Assigns the native VLAN ID of a physical interface trunking 802.1Q VLAN traffic.

Step 10 **router bgp** *as-number*

Example:

```
RP/0/RSP0/CPU0:router(config)# router bgp 1
```

Specifies the autonomous system number and enters the BGP configuration mode, allowing you to configure the BGP routing process.

Step 11 **nsr**

Example:

```
RP/0/RSP0/CPU0:router(config-bgp)# nsr
```

Activates BGP NSR.

Step 12 **bgp router-id** *ip-address*

Example:

```
RP/0/RSP0/CPU0:router(config-bgp)# bgp router-id 10.0.0.1
```

Configures the local router with a specified router ID.

Step 13 **address-family {vpnv4 | vpnv6} unicast**

Example:

```
RP/0/RSP0/CPU0:router(config-bgp)# address-family vpnv4 unicast
```

Enters address family configuration submode for the specified address family.

Step 14 **neighbor** *ip-address*

Example:

```
RP/0/RSP0/CPU0:router(config-bgp)# neighbor 192.168.0.1
```

Places the router in neighbor configuration mode for BGP routing and configures the neighbor IP address as a BGP peer.

Step 15 **remote-as** *as-number*

Example:

```
RP/0/RSP0/CPU0:router(config-bgp-nbr)#remote-as 1
```

Creates a neighbor and assigns a remote autonomous system number to it..

Step 16 **update-source** *type interface-path-id*

Example:

```
RP/0/RSP0/CPU0:router(config-bgp-nbr)#update-source Loopback0
```

Allows sessions to use the primary IP address from a specific interface as the local address when forming a session with a neighbor.

Step 17 **address-family { vpnv4 | vpnv6 } unicast**

Example:

```
RP/0/RSP0/CPU0:router(config-bgp-nbr)# address-family vpnv4 unicast
```

Enters address family configuration submode for the specified address family.

Step 18 **route-policy** *route-policy-name* **in**

Example:

```
RP/0/RSP0/CPU0:router(config-bgp-nbr-af)#route-policy pass-all in
```

Defines a route policy and enters route policy configuration mode.

Step 19 **route-policy** *route-policy-name* **out**

Example:

```
RP/0/RSP0/CPU0:router(config-bgp-nbr-af)#route-policy pass-all out
```

Defines a route policy and enters route policy configuration mode.

Step 20 **vrf** *vrf-name*

Example:

```
RP/0/RSP0/CPU0:router(config)# vrf vpn1
```

Configures a VRF instance.

Step 21 **rd { as-number:nn | ip-address:nn | auto }**

Example:

```
RP/0/RSP0/CPU0:router(config-vrf)#rd 1:1
```

Configures the route distinguisher.

Step 22 **address-family { ipv4 | ipv6 } unicast**

Example:

```
RP/0/RSP0/CPU0:router(config-vrf)# address-family ipv4 unicast
```

Specifies either the IPv4 or IPv6 address family and enters address family configuration submode.

Step 23 **redistribute connected** [**metric** *metric-value*] [*route-policy route-policy-name*]

Example:

```
RP/0/RSP0/CPU0:router(config-vrf-af)#
redistribute connected
```

Configures the local router with a specified router ID.

Step 24 **redistribute static** [**metric** *metric-value*] [*route-policy route-policy-name*]

Example:

```
RP/0/RSP0/CPU0:router(config-vrf-af)#
redistribute static
```

Causes routes from the specified instance to be redistributed into BGP.

Step 25 **neighbor** *ip-address*

Example:

```
RP/0/RSP0/CPU0:router(config-bgp)# neighbor 172.16.0.2
```

Places the router in neighbor configuration mode for BGP routing and configures the neighbor IP address as a BGP peer.

Step 26 **remote-as** *as-number*

Example:

```
RP/0/RSP0/CPU0:router(config-bgp-nbr)#remote-as 7501
```

Creates a neighbor and assigns a remote autonomous system number to it..

Step 27 **ebg-multihop** *ttl-value*

Example:

```
RP/0/RSP0/CPU0:router(config-bgp-nbr)
#ebgp-multihop 10
```

Configures the CE neighbor to accept and attempt BGP connections to external peers residing on networks that are not directly connected.

Step 28 **address-family { ipv4 | ipv6 } unicast**

Example:

```
RP/0/RSP0/CPU0:router(config-bgp-nbr)# address-family ipv4 unicast
```

Specifies either the IPv4 or IPv6 address family and enters address family configuration submode.

Step 29 **route-policy** *route-policy-name* **in**

Example:

```
RP/0/RSP0/CPU0:router(config-bgp-nbr-af)#route-policy
BGP_pass_all in
```

Configures the local router with a specified router.

Step 30 **route-policy** *route-policy-name* **out**

Example:

```
RP/0/RSP0/CPU0:router(config-bgp-nbr-af)#route-policy BGP_pass_all out
```

Defines a route policy and enters route policy configuration mode.

Step 31 Use the **commit** or **end** command.

commit - Saves the configuration changes and remains within the configuration session.

end - Prompts user to take one of these actions:

- **Yes** - Saves configuration changes and exits the configuration session.
- **No** - Exits the configuration session without committing the configuration changes.
- **Cancel** - Remains in the configuration mode, without committing the configuration changes.

Configuration Examples for Implementing MPLS Layer 3 VPNs

The following section provides sample configurations for MPLS L3VPN features:

Configuring an MPLS VPN Using BGP: Example

The following example shows the configuration for an MPLS VPN using BGP on “vrf vpn1”:

```
address-family ipv4 unicast
    import route-target
      100:1
    !
    export route-target
      100:1
    !
  !
!
route-policy pass-all
  pass
end-policy
!
interface Loopback0
  ipv4 address 10.0.0.1 255.255.255.255
!
interface TenGigE 0/1/0/0
  vrf vpn1
  ipv4 address 10.0.0.2 255.0.0.0
!
interface TenGigE 0/1/0/1
  ipv4 address 10.0.0.1 255.0.0.0
!
router ospf 100
  area 100
    interface loopback0
    interface TenGigE 0/1/0/1
  !
!
router bgp 100
  address-family vpnv4 unicast
  retain route-target route-policy policy1
  neighbor 10.0.0.3
    remote-as 100
    update-source Loopback0
    address-family vpnv4 unicast
  !
  vrf vpn1
    rd 100:1
    address-family ipv4 unicast
      redistribute connected
    !
    neighbor 10.0.0.1
      remote-as 200
      address-family ipv4 unicast
        as-override
        route-policy pass-all in
        route-policy pass-all out
      !
      advertisement-interval 5
    !
  !
```

```
!
mpls ldp
  route-id looback0
  interface TenGigE 0/1/0/1
!
```

Configuring the Routing Information Protocol on the PE Router: Example

The following example shows the configuration for the RIP on the PE router:

```
vrf vpn1
  address-family ipv4 unicast
    import route-target
      100:1
    !
    export route-target
      100:1
    !
  !
!
route-policy pass-all
  pass
end-policy
!

interface TenGigE 0/1/0/0
  vrf vpn1
  ipv4 address 10.0.0.2 255.0.0.0
!

router rip
 vrf vpn1
  interface TenGigE 0/1/0/0
  !
  timers basic 30 90 90 120
  redistribute bgp 100
  default-metric 3
  route-policy pass-all in
 !
```

Configuring the PE Router Using EIGRP: Example

The following example shows the configuration for the Enhanced Interior Gateway Routing Protocol (EIGRP) on the PE router:

```
Router eigrp 10
 vrf VRF1
  address-family ipv4
   router-id 10.1.1.2
   default-metric 100000 2000 255 1 1500
   as 62
   redistribute bgp 2000
   interface Loopback0
   !
   interface TenGigE 0/6/0/0
```

Configuration Examples for MPLS VPN CSC

Configuration examples for the MPLS VPN CSC include:

Configuring the Backbone Carrier Core: Examples

Configuration examples for the backbone carrier core included in this section are as follows:

Configuring VRFs for CSC-PE Routers: Example

The following example shows how to configure a VPN routing and forwarding instance (VRF) for a CSC-PE router:

```
config
  vrf vpn1
   address-family ipv4 unicast
    import route-target 100:1
    export route-target 100:1
   end
```

Configuring the Links Between CSC-PE and CSC-CE Routers: Examples

This section contains the following examples:

Configuring a CSC-PE: Example

In this example, a CSC-PE router peers with a PE router, 10.1.0.2, in its own AS. It also has a labeled unicast peering with a CSC-CE router, 10.0.0.1.

```
config
 router bgp 2
  address-family vpnv4 unicast
  neighbor 10.1.0.2
      remote-as 2
     update-source loopback0
     address-family vpnv4 unicast
    vrf customer-carrier
     rd 1:100
     address-family ipv4 unicast
      allocate-label all
      redistribute static
    neighbor 10.0.0.1
     remote-as 1
     address-family ipv4 labeled-unicast
      route-policy pass-all in
      route-policy pass-all out
      as-override
end
```

Configuring a CSC-CE: Example

The following example shows how to configure a CSC-CE router. In this example, the CSC-CE router peers CSC-PE router 10.0.0.2 in AS 2.

```
config
 router bgp 1
  address-family ipv4 unicast
     redistribute ospf 200
     allocate-label all
    neighbor 10.0.0.2
     remote-as 2
     address-family ipv4 labeled-unicast
```

```
     route-policy pass-all in
     route-policy pass-all out
end
```

Configuring a Static Route to a Peer: Example

The following example shows how to configure a static route to an Inter-AS or CSC-CE peer:

```
config
 router static
  address-family ipv4 unicast
  10.0.0.2/32 172.16.0.1
end
```

Configuring a Static Route to a Peer: Example

This example shows how to configure a static route to an Inter-AS or CSC-CE peer:

```
config
 router static
  address-family ipv4 unicast
  10.0.0.2/32 40.1.1.1
end
```

Configuring L3VPN over GRE: Example

The following example shows how to configure L3VPN over GRE:

Sample configuration to create a GRE tunnel between PE1 and PE2:

```
RP/0/RSP0/CPU0:PE1#sh run int tunnel-ip 1
interface tunnel-ip1
 ipv4 address 172.16.0.1 255.240.0.0
 ipv6 address 100:1:1:1::1/64
 tunnel mode gre ipv4
 tunnel source TenGigE0/2/0/1
 tunnel destination 145.12.5.2
!
RP/0/RSP0/CPU0:PE2#sh run int tunnel-ip 1
interface tunnel-ip1
 ipv4 address 172.16.0.2 255.240.0.0
 ipv6 address 100:1:1:1::2/64
 tunnel mode gre ipv4
 tunnel source TenGigE0/1/0/2
 tunnel destination 192.168.0.1
```

Configure IGP between PE1 and PE2:

Sample configuration for PE1 is given below. PE2 will also have a similar configuration.

```
RP/0/RSP0/CPU0:PE1#sh run router ospf 1
router ospf 1
 nsr
 router-id 10.0.0.1 <=== Loopback0
 mpls ldp sync
```

```
 mtu-ignore enable
 dead-interval 60
 hello-interval 15
 area 0
  interface TenGigE0/2/0/1
  !
 RP/0/RSP0/CPU0:PE1#sh run router ospf 0
router ospf 0
 nsr
 router-id 10.0.0.1
 mpls ldp sync
 dead-interval 60
 hello-interval 15
 area 0
  interface Loopback0
  !
  interface tunnel-ip1
  !

* Check for OSPF neighbors

RP/0/RSP0/CPU0:PE1#sh ospf neighbor

Neighbors for OSPF 0

Neighbor ID     Pri   State           Dead Time   Address         Interface
4.4.4.4         1     FULL/  -        00:00:47    172.16.0.2       tunnel-ip1      <==
Neighbor PE2
    Neighbor is up for 00:13:40
Neighbors for OSPF 1

Neighbor ID     Pri   State           Dead Time   Address         Interface
2.2.2.2         1     FULL/DR         00:00:50    192.168.0.1      TenGigE0/2/0/1  <==
Neighbor P1
    Neighbor is up for 00:13:43
```

Configure LDP/GRE on PE1 and PE2:

```
RP/0/RSP0/CPU0:PE1#sh run mpls ldp
mpls ldp
 router-id 10.0.0.1 <=== Loopback0
 discovery hello holdtime 45
 discovery hello interval 15
 nsr
 graceful-restart
 graceful-restart reconnect-timeout 180
 graceful-restart forwarding-state-holdtime 300
 holdtime 90
 log
  neighbor
 !
 interface tunnel-ip1
 !

*Check for mpls forwarding

RP/0/RSP0/CPU0:PE1#sh mpls forwarding prefix 10.0.0.2/8
Local  Outgoing    Prefix             Outgoing     Next Hop        Bytes
Label  Label       or ID              Interface                    Switched
---- ------- -------------- -------- ----------- ----------
```

```
16003  Pop         10.0.0.2/8         ti1          172.16.0.2       0
```

Configure L3VPN

```
RP/0/RSP0/CPU0:PE1#sh run vrf vpn1
vrf vpn1
 address-family ipv4 unicast
  import route-target
   2:1
  !
  export route-target
   1:1
  !
RP/0/RSP0/CPU0:PE1#sh run int tenGigE 0/2/0/0.1
interface TenGigE0/2/0/0.1
 vrf vpn1
 ipv4 address 172.16.0.1 255.255.255.0
 encapsulation dot1q 1
!

RP/0/RSP0/CPU0:PE1#sh run router bgp
router bgp 1
 nsr
 bgp router-id 10.0.0.1 <===Loopback0
 address-family vpnv4 unicast
 !
 neighbor 192.168.0.1      <===iBGP session with PE2
  remote-as 1
  update-source Loopback0
  address-family vpnv4 unicast
   route-policy pass-all in
   route-policy pass-all out
  !
 !
 vrf vpn1
  rd 1:1
  address-family ipv4 unicast
   redistribute connected
   redistribute static
  !
  neighbor 172.16.0.2   <=== VRF neighbor
   remote-as 7501
   ebgp-multihop 10
   address-family ipv4 unicast
    route-policy BGP_pass_all in
    route-policy BGP_pass_all out
   !

* Check vrf ping to the 172.16.0.2

RP/0/RSP0/CPU0:PE1#ping vrf vpn1 172.16.0.2
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 172.16.0.2, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/1/3 ms

* Send traffic to vrf routes adverstised and verify that mpls counters increase in tunnel
interface accounting

RP/0/RSP0/CPU0:PE1#sh int tunnel-ip1 accounting
tunnel-ip1
  Protocol              Pkts In         Chars In     Pkts Out        Chars Out
```

```
IPV4_MULTICAST              3              276            3              276
 MPLS                  697747         48842290            0                0
```

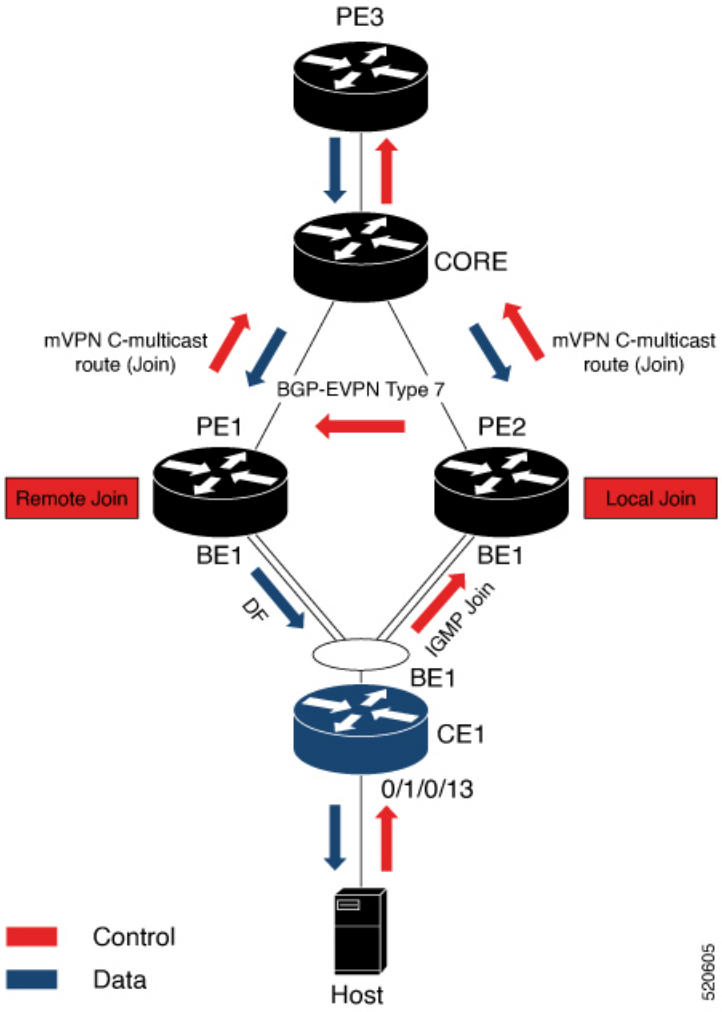
EVPN IGMP L3 Synchronization

The EVPN IGMP L3 Synchronization feature enables you to synchronize IGMPv2 and IGMPv3 reports on multihomed PEs over EVPN. The multihomed PEs synchronize IGMP JOIN using BGP EVPN route-type 7. The multihomed PEs synchronize IGMP LEAVE using BGP EVPN route-type 8. This feature provides better serviceability.

Restrictions

- This feature is supported on the Cisco ASR9000 3rd generation line cards.
- This feature is supported on the Cisco ASR 9000 4th generation line cards with the Cisco IOS XR 64-bit operating system.
- This feature only supports receivers behind multihomed PEs.
- This feature does not support multicast source behind multihomed PEs.
- This feature supports only EVPN multihoming active-active mode.
- This feature does not support the coexistence of L2 IGMP synchronization and L3 IGMP synchronization. You must configure only one mode at any point in time.

Topology



Consider a topology where CE1 is multihomed to PE1 and PE2. When the host sends IGMPv2 or IGMPv3 JOIN to CE1. CE1 hashes the JOIN to either PE2 or PE1. When CE1 hashes the JOIN to PE2. IGMP learns the group as local on PE2. IGMP notifies EVPN and updates MRIB on PE2. BGP advertises IGMP JOIN to PE1 using BGP EVPN route-type 7. Both PE1 and PE2 sends the mVPN C-multicast route (Join) to the source, which is PE3. Both PE1 and PE2 act as a designated router (DR) and receives traffic from PE3. Only one of the PE, either PE1 or PE2 which is the designated forwarder (DF), sends traffic to CE1.

In multihoming active-active mode, both PEs receive the traffic from the core. However, only one of the PEs forwards traffic to CE to avoid duplicate traffic. To enable this, bucket IDs are used. Bucket ID is allotted based on the evpn-route-sync id configured either under VRF or under EVPN. You must configure the same evpn-route-sync id under VRF on both the PEs.

Configure each VRF with a different evpn-route-sync id to enable load balancing. Each VRF is allocated with a different bucket ID. For example, if you configure VRF RED on PE1, and VRF BLUE on PE2, then the routes in VRF RED are allotted bucket ID 1, and routes in VRF BLUE are allotted bucket ID 2. PE1 becomes the DF for bucket ID 1, and PE2 becomes the DF for the bucket ID 2.

Configure EVPN IGMP L3 Synchronization

This section describes how to configure the EVPN IGMP L3 Synchronization feature.

Configuration Example

Peform this task to configure EVPN IGMP L3 Synchronization.

```
/* PE1 Configuration */
Router# configure
Router(config)#  interface Loopback0
Router(config-if)# ipv4 address 10.0.0.1 255.0.0.0
Router(config-if)# exit
Router(config)#  vrf vpn101 -> ESI on non-default VRF
Router(config-vrf)# evpn-route-sync 101
Router(config-vrf)# exit
Router(config)# lacp system mac 0004.0005.0006
Router(config)# commit

Router# configure
Router(config)# router bgp 100
Router(config-bgp)# bgp router-id 10.0.0.1
Router(config-bgp)# address-family l2vpn evpn
Router(config-bgp-af)# exit
Router(config-bgp)# neighbor 172.16.0.1
Router(config-bgp-nbr)#r remote-as 100
Router(config-bgp-nbr)# update-source Loopback0
Router(config-bgp-nbr)# address-family l2vpn evpn
Router(config-bgp-nbr-af)# commit

Router# configure
Router(config)# evpn
Router(config-evpn)# route-sync 2001 -> Associate EVI to default VRF
Router(config-evpn-instance)# vrf default
Router(config-evpn-instance)# exit
Router(config-evpn)# group 1
Router(config-evpn-group)# core interface TenGigE0/1/0/0/4
Router(config-evpn-group)# core interface TenGigE0/1/0/0/5
Router(config-evpn-group)# exit
Router(config-evpn)# interface Bundle-Ether1
Router(config-evpn-ac)# ethernet-segment
Router(config-evpn-ac-es)# identifier type 0 00.01.00.ac.00.00.01.0a.00
Router(config-evpn-ac-es)# core-isolation-group 1
Router(config-evpn-ac)# commit

Router# configure
Router(config)# interface Bundle-Ether1
Router(config-if)# bundle wait-while 100
Router(config-if)# exit
Router(config)# interface bundle-ether 1.10
Router(config-subif)# vrf vpn101
Router(config-subif)# ipv4 address 192.168.0.1 255.255.0.0
Router(config-subif)# encapsulation dot1q 10
Router(config-subif)# exit
Router(config)# interface bundle-ether 1.20
Router(config-subif)# ipv4 address 192.168.1.1 255.255.0.0
Router(config-subif)# encapsulation dot1q 11
Router(config-subif)# commit


/* PE2 Configuration */
```

```
Router# configure
Router(config)#  interface Loopback0
Router(config-if)# ipv4 address 172.16.0.1 255.240.0.0
Router(config-if)# exit
Router(config)#  vrf vpn101 -> ESI on non-default VRF
Router(config-vrf)# evpn-route-sync 101
Router(config-vrf)# exit
Router(config)# lacp system mac 0004.0005.0006
Router(config)# commit

Router# configure
Router(config)# router bgp 100
Router(config-bgp)# bgp router-id 172.16.0.1
Router(config-bgp)# address-family l2vpn evpn
Router(config-bgp-af)# exit
Router(config-bgp)# neighbor 10.0.0.1
Router(config-bgp-nbr)#r remote-as 100
Router(config-bgp-nbr)# update-source Loopback0
Router(config-bgp-nbr)# address-family l2vpn evpn
Router(config-bgp-nbr-af)# commit

Router# configure
Router(config)# evpn
Router(config-evpn)# route-sync 2001 -> Associate EVI to default VRF
Router(config-evpn-instance)# vrf default
Router(config-evpn-instance)# exit
Router(config-evpn)# group 1
Router(config-evpn-group)# core interface TenGigE0/1/0/0/4
Router(config-evpn-group)# core interface TenGigE0/1/0/0/5
Router(config-evpn-group)# exit
Router(config-evpn)# interface Bundle-Ether1
Router(config-evpn-ac)# ethernet-segment
Router(config-evpn-ac-es)# identifier type 0 00.01.00.ac.00.00.01.0a.00
Router(config-evpn-ac-es)# core-isolation-group 1
Router(config-evpn-ac)# commit

Router# configure
Router(config)# interface Bundle-Ether1
Router(config-if)# bundle wait-while 100
Router(config-if)# exit
Router(config)# interface bundle-ether 1.10
Router(config-subif)# vrf vpn101
Router(config-subif)# ipv4 address 192.168.0.1 255.255.0.0
Router(config-subif)# encapsulation dot1q 10
Router(config-subif)# exit
Router(config)# interface bundle-ether 1.20
Router(config-subif)# ipv4 address 192.168.1.1 255.255.0.0
Router(config-subif)# encapsulation dot1q 11
Router(config-subif)# commit
```

Running Configuration

This section shows the EVPN IGMP L3 synchronization running configuration.

```
/* PE1 Configuratin */
interface Loopback0
 ipv4 address 10.0.0.1 255.0.0.0
!
vrf vpn101  -> ESI on non-default VRF
 evpn-route-sync 101
 !
```

```
lacp system mac 0004.0005.0006
 !
!

router bgp 100
bgp router-id 10.0.0.1
address-family l2vpn evpn
!
neighbor 172.16.0.1
 remote-as 100
 update-source Loopback0
 address-family l2vpn evpn
!
!

evpn
 route-sync 2001  -> Associate EVI to default VRF
  vrf default
!
 group 1
  core interface TenGigE0/1/0/0/4
  core interface TenGigE0/1/0/0/5
 !

 interface Bundle-Ether1
  ethernet-segment
   identifier type 0 00.01.00.ac.00.00.01.0a.00
  !
  core-isolation-group 1
!


interface Bundle-Ether1
 bundle wait-while 100
!
interface Bundle-Ether1.10
 vrf vpn101
 ipv4 address 192.168.0.1 255.255.0.0
 encapsulation dot1q 10
 !
interface Bundle-Ether1.20
 ipv4 address 192.168.1.1 255.255.0.0
 encapsulation dot1q 11
 !


/* PE2 Configuration */
interface Loopback0
 ipv4 address 172.16.0.1  255.240.0.0
!
vrf vpn101  -> ESI on non default VRF
 evpn-route-sync 101
 !
lacp system mac 0004.0005.0006
!

router bgp 100
bgp router-id 172.16.0.1
address-family l2vpn evpn
!
neighbor 10.0.0.1
 remote-as 100
 update-source Loopback0
 address-family l2vpn evpn
!
```

```
!

evpn
 route-sync 2001   -> Associate EVI to default VRF
  vrf default
!
group 1
  core interface TenGigE0/1/0/0/4
  core interface TenGigE0/1/0/0/5
 !

 interface Bundle-Ether1
  ethernet-segment
   identifier type 0 00.01.00.ac.00.00.01.0a.00
  !

  core-isolation-group 1
!
interface Bundle-Ether1
 bundle wait-while 100
!
interface Bundle-Ether1.10
 vrf vpn101
 ipv4 address 192.168.0.1 255.255.0.0
 encapsulation dot1q 10
 !
interface Bundle-Ether1.20
 ipv4 address 192.168.1.1 255.255.0.0
 encapsulation dot1q 11
 !
```

Verification

Verify that you have configured EVPN IGMP L3 synchronization successfully.

```
/* Verify EVPN ethernet-segment peers */
Router# show evpn ethernet-segment

Ethernet Segment Id      Interface                         Nexthops
------------------------ --------------------------------- --------------------
0000.0100.ac00.0001.0a00 BE1                               10.0.0.1
                                                           172.16.0.1
-------------------------------------------------------------------------------

 /* Verify multihome interface is enabled in IGMP */

Router# show igmp vrf vpn101 interface bundle-ether 1.10
Bundle-Ether1.10 is up, line protocol is up
  Internet address is 192.168.0.1 255.255.0.0
  IGMP_AFD is enabled on interface
  Multihoming is enabled on interface [Stale : False].  -> multihome must be enabled
  Current IGMP version is 3


/* Verify bucket IDS in control plane and hardware */
Router# show mrib evpn bucket-db

EVPN Bucket Database
--------------------

IFName IFHandle BucketID State Uptime Delete In Progress
```

```
Bundle-Ether1 0x20008e0 0 Forward 3d17h N
Bundle-Ether1 0x20008e0 1 Blocked 3d17h N
Bundle-Ether1 0x20008e0 2 Forward 3d17h N
Bundle-Ether1 0x20008e0 3 Blocked 3d17h N
Bundle-Ether1 0x20008e0 4 Forward 3d17h N
Bundle-Ether1 0x20008e0 5 Blocked 3d17h N
Bundle-Ether1 0x20008e0 6 Forward 3d17h N
Bundle-Ether1 0x20008e0 7 Blocked 3d17h N
Bundle-Ether1 0x20008e0 8 Forward 3d17h N
Bundle-Ether1 0x20008e0 9 Blocked 3d17h N
Bundle-Ether1 0x20008e0 10 Forward 3d17h N
Bundle-Ether1 0x20008e0 11 Blocked 3d17h N

Router# show mfib platform evpn bucket location 0/1/CPU0
LC Type: A9K-4X100GE-TR
-------------------------------------------------------
ESI Interface        Handle     Bucket ID State Stale
-------------------------------------------------------
Bundle-Ether1        0x4000620          0       DF     F
Bundle-Ether1        0x4000620          1      NDF     F
Bundle-Ether1        0x4000620          2       DF     F
Bundle-Ether1        0x4000620          3      NDF     F
Bundle-Ether1        0x4000620          4       DF     F
Bundle-Ether1        0x4000620          5      NDF     F
Bundle-Ether1        0x4000620          6       DF     F
Bundle-Ether1        0x4000620          7      NDF     F
Bundle-Ether1        0x4000620          8       DF     F
Bundle-Ether1        0x4000620          9      NDF     F
Bundle-Ether1        0x4000620         10       DF     F
Bundle-Ether1        0x4000620         11      NDF     F
-------------------------------------------------------
If duplicate traffic is seen or one of the PEs is dropping traffic even though the routes
are present, it could be that bucket ID states are incorrect in hardware or control plane.
```

Verify IGMP report in IGMP, EVPN, and BGP.

```
Router:PE1#  show igmp vrf vpn101 groups 209.165.201.1 detail
Interface:      Bundle-Ether1.10
Group:          209.165.201.1
Uptime:         01:11:24
Router mode:    EXCLUDE (Expires: never)
Host mode:      INCLUDE
Last reporter:  10.0.0.2
Suppress:       0
EVPN Remote group: True (Stale: False)  -> Learnt over EVPN
Source list is empty


Router:PE1# show evpn igmp detail
EVI   Ethernet Segment         (S,G)                                          Source
Type
----- ----------------------- ---------------------------------------------- --------------
------
101   0000.0100.ac00.0001.0a00 (0.0.0.0,209.165.201.1)                         172.16.0.1
JOIN
  Ethernet Tag      : 10
  IGMP Version      : V3 [IS_EX]

Router#show bgp l2vpn evpn rd 172.16.0.1:101
[7][0000.0100.ac00.0001.0a00][10][32][0.0.0.0][32][
209.165.201.1][32][172.16.0.1]/240
BGP routing table entry for
[7][0000.0100.ac00.0001.0a00][10][32][0.0.0.0][32][209.165.201.1][32][172.16.0.1]/240,
Route Distinguisher: 172.16.0.1:101
```

```
Versions:
  Process           bRIB/RIB  SendTblVer
  Speaker               6352        6352
Last Modified: Mar 31 00:09:50.666 for 01:42:22
Paths: (1 available, best #1)
  Not advertised to any peer
  Path #1: Received by speaker 0
  Not advertised to any peer
  Local
    172.16.0.1 (metric 3) from 172.16.0.1 (172.16.0.1)
      Origin IGP, localpref 100, valid, internal, best, group-best, import-candidate,
not-in-vrf
      Received Path ID 0, Local Path ID 1, version 6352
      Extended community: EVPN ES Import:0001.00ac.0000 EVI RT:0064.0000.0065
0x060e:0000.ffff.ffff
      IGMP Flags: 0xc

Router:PE2# show igmp vrf vpn101 groups 209.165.201.1 detail

Interface:      Bundle-Ether1.10
Group:          209.165.201.1
Uptime:         01:15:14
Router mode:    EXCLUDE (Expires: 00:02:09)
Host mode:      INCLUDE
Last reporter:  10.0.0.2
Suppress:       0
EVPN Remote group: False (Stale: False)    -> Learnt locally
Source list is empty


Router:PE2# show evpn igmp detail

EVI   Ethernet Segment         (S,G)                          Source             Type
----- ----------------------- ------------------------------ ------------------ ------
101   0000.0100.ac00.0001.0a00 (0.0.0.0,209.165.201.1)      Bundle-Ether1.10    JOIN
  Ethernet Tag      : 10

  IGMP Version      : V3 [IS_EX]

Router:PE2 #show bgp l2vpn evpn rd 172.16.0.1:101
[7][0000.0100.ac00.0001.0a00][10][32][0.0.0.0][32]
[209.165.201.1][32][172.16.0.1]/240
BGP routing table entry for
[7][0000.0100.ac00.0001.0a00][10][32][0.0.0.0][32][209.165.201.1][32][172.16.0.1]/240,
 Route Distinguisher: 172.16.0.1
Versions:
  Process           bRIB/RIB  SendTblVer
  Speaker               9328        9328
Last Modified: Mar 31 00:09:47.824 for 01:41:49
Paths: (1 available, best #1)
  Advertised to peers (in unique update groups):
    10.0.0.1
  Path #1: Received by speaker 0
  Advertised to peers (in unique update groups):
    10.0.0.1
  Local
    0.0.0.0 from 0.0.0.0 (172.16.0.1)
      Origin IGP, localpref 100, valid, redistributed, best, group-best, import-candidate,
 rib-install
      Received Path ID 0, Local Path ID 1, version 9328
      Extended community: EVPN ES Import:0001.00ac.0000 EVI RT:0064.0000.0065
0x060e:0000.ffff.ffff
      EVPN ESI: 0000.0000.0000.0000.0000
      IGMP Flags: 0xc
```

```
IGMPv2 report for group is local on PE2 and remote on PE.
```

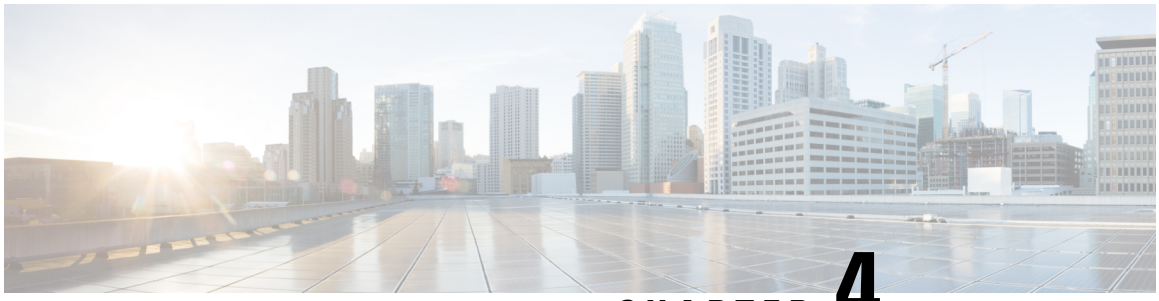
Verify MRIB entries

For IGMPv2 hosts, only *,G OLEs have bucket IDs displayed in MRIB. The bucket IDs are not displayed for S,G OLE.

For IGMPv3 hosts, S,G OLE bucket IDs are displayed in MRIB.

```
Router:PE1# show mrib vrf vpn101 route 209.165.201.1
(*,209.165.201.1) RPF nbr: 10.0.0.5 Flags: C RPF
  Up: 02:23:51
  Incoming Interface List
    mdtvpn101 Flags: A NS MI, Up: 02:23:51
  Outgoing Interface List
    Bundle-Ether1.10 (0/1/CPU0) Flags: F NS LI MH, Up: 02:23:50

(192.168.0.4,209.165.201.1) RPF nbr: 10.0.0.5 Flags: RPF
  Up: 02:23:07
  Incoming Interface List
    mdtvpn101 Flags: A MI, Up: 02:23:07
  Outgoing Interface List
    Bundle-Ether1.10 (0/1/CPU0) Flags: F NS, Up: 02:23:07


Router:PE1# show mrib vrf vpn101 route 209.165.201.1 priv
Tue Mar 31 02:33:15.978 UTC
(*,209.165.201.1) RPF nbr: 10.0.0.5 Flags: C RPF
  Up: 02:26:22, Route node: 0x556459daff90, In PD retry list: N
  RPF-ID: 1, Encap-ID: 0, EPtr: 0x0, New EPtr: 0x0, New EID: 0, Hd: 0x0, Cts: 0, 0, 0, 0
  Acc: 1, Fwd: 10 (10), Encap-next: 0x0
  Incoming Interface List
   mdtvpn101 Flags: A NS MI, Up: 02:26:22, type 0, Ptrs: 0x556459e14a58, 0x0 0x0 0x0 0x0,
 Bundle Ptrs: 0x0(vp)
 , 0x0(vn) 0x0(pp) 0x0(pn)
   Outgoing Interface List
     Bundle-Ether1.10 (0/1/CPU0, 0x6004980) Flags: F NS LI MH, Up: 02:26:20, type 0, Ptrs:
0x55645a8ba750, 0x0
   0x0 0x0 0x00x55645a5397d8(l) , Bundle Ptrs: 0xf9b20000f8(vp), 0x47a00d37000400(vn)
0x2100000000000040(pp)
0xa19000000001504(pn)
       LI add redist count: 1
       Platform MH MRIB Annotation: ESI: 0x4000620 Bucket ID: 5
 (192.168.0.4,209.165.201.1) RPF nbr: 10.0.0.5 Flags: RPF
   Up: 02:25:37, Route node: 0x55645aabec00, In PD retry list: N
   RPF-ID: 1, Encap-ID: 0, EPtr: 0x0, New EPtr: 0x0, New EID: 0, Hd: 0x0, Cts: 0, 0, 0, 0
   Acc: 1, Fwd: 10 (10), Encap-next: 0x0
   Incoming Interface List
     mdtvpn101 Flags: A MI, Up: 02:25:37, type 0, Ptrs: 0x55645acf7450, 0x0 0x0 0x0 0x0,
 Bundle Ptrs: 0x100000020830000(vp)
 , 0x0(vn) 0x0(pp) 0x100000000000000(pn)
   Outgoing Interface List
     Bundle-Ether1.10 (0/1/CPU0, 0x6004940) Flags: F NS, Up: 02:25:37, type 0, Ptrs:
 0x55645aceb090, 0x0 0x0 0x0 0x00x55645
 a5397f8(l) , Bundle Ptrs: 0x0(vp), 0x0(vn) 0x0(pp) 0x0(pn)


Router:PE2# show mrib vrf vpn101 route 209.165.201.1

(*,209.165.201.1) RPF nbr: 10.0.0.5 Flags: C RPF
  Up: 02:16:24
  Incoming Interface List
```

```
    mdtvpn101 Flags: A NS MI, Up: 02:16:24
  Outgoing Interface List
    Bundle-Ether1.10 (0/0/CPU0) Flags: F NS LI MH, Up: 02:16:24

(192.168.0.4,209.165.201.1) RPF nbr: 10.0.0.5 Flags: RPF
  Up: 02:16:06
  Incoming Interface List
    mdtvpn101 Flags: A MI, Up: 02:16:06
  Outgoing Interface List
    Bundle-Ether1.10 (0/1/CPU0) Flags: F NS, Up: 02:16:06


Router:PE2# show mrib vrf vpn101 route 209.165.201.1 priv
Tue Mar 31 02:27:18.748 UTC
(*,209.165.201.1) RPF nbr: 10.0.0.5 Flags: C RPF
  Up: 02:17:38, Route node: 0x55bb45ca3948, In PD retry list: N
  RPF-ID: 1, Encap-ID: 0, EPtr: 0x0, New EPtr: 0x0, New EID: 0, Hd: 0x0, Cts: 0, 0, 0, 0
  Acc: 1, Fwd: 10 (10), Encap-next: 0x0
  Incoming Interface List
   mdtvpn101 Flags: A NS MI, Up: 02:17:38, type 0, Ptrs: 0x55bb463db210, 0x0 0x0 0x0 0x0,
 Bundle Ptrs: 0x0(vp), 0x0(vn) 0x0(pp) 0x0(pn)
  Outgoing Interface List
    Bundle-Ether1.10 (0/0/CPU0, 0x4d00) Flags: F NS LI MH, Up: 02:17:38, type 0, Ptrs:
 0x55bb45271ae8, 0x0 0x0 0x0 0x00x55bb459d3708(l) , Bundle Ptrs: 0x0(vp), 0x0(vn) 0x0(pp)
  0x0(pn)
      LI add redist count: 3
      Platform MH MRIB Annotation: ESI: 0x2000be0 Bucket ID: 5
 (192.168.0.4,209.165.201.1) RPF nbr: 10.0.0.5 Flags: RPF
   Up: 02:17:19, Route node: 0x55bb45e546b0, In PD retry list: N
   RPF-ID: 1, Encap-ID: 0, EPtr: 0x0, New EPtr: 0x0, New EID: 0, Hd: 0x0, Cts: 0, 0, 0, 0
   Acc: 1, Fwd: 10 (10), Encap-next: 0x0
Incoming Interface List
 mdtvpn101 Flags: A MI, Up: 02:17:19, type 0, Ptrs: 0x55bb4526a708, 0x0 0x0 0x0 0x0,
Bundle Ptrs: 0x0(vp), 0x0(vn) 0x0(pp) 0x0(pn)
 Outgoing Interface List
  Bundle-Ether1.10 (0/1/CPU0, 0x6009ac0) Flags: F NS, Up: 02:17:19, type 0, Ptrs:
0x55bb460068f0, 0x0 0x0 0x0 0x00x55bb459d36a8(l) , Bundle Ptrs: 0x0(vp), 0x0(vn) 0x0(pp)
  0x0(pn)

.
```



CHAPTER 4

Implementing IPv6 VPN Provider Edge Transport over MPLS

IPv6 Provider Edge or IPv6 VPN Provider Edge (6PE/VPE) uses the existing MPLS IPv4 core infrastructure for IPv6 transport. 6PE/VPE enables IPv6 sites to communicate with each other over an MPLS IPv4 core network using MPLS label switched paths (LSPs).

This feature relies heavily on multiprotocol Border Gateway Protocol (BGP) extensions in the IPv4 network configuration on the provider edge (PE) router to exchange IPv6 reachability information (in addition to an MPLS label) for each IPv6 address prefix. Edge routers are configured as dual-stack, running both IPv4 and IPv6, and use the IPv4 mapped IPv6 address for IPv6 prefix reachability exchange.

For detailed information about the commands used to configure 6PE/VPE, see the *VPN and Ethernet Services Command Reference for Cisco ASR 9000 Series Routers*.

Feature History for Implementing 6PE/VPE Transport over MPLS

Release	Modification
Release 3.9.1	This feature was introduced.
Release 4.0.0	Support was added for the 6PE and 6VPE features for IPv6 L3VPN on A9K-SIP-700. Support was added for the BGP per VRF/CE label allocation for 6PE feature.
Release 4.1.0	Support for the Open Shortest Path First version 3 (OSPFv3) IPv6 VPN Provider Edge (6VPE) feature was added.

Prerequisites for Implementing 6PE/VPE

The following prerequisites are required to implement 6PE/VPE:

- You must be in a user group associated with a task group that includes the proper task IDs. The command reference guides include the task IDs required for each command.

  If you suspect user group assignment is preventing you from using a command, contact your AAA administrator for assistance.

- Familiarity with MPLS and BGP4 configuration and troubleshooting.

Information About 6PE/VPE

To configure the 6PE/VPE feature, you should understand the concepts that are described in these sections:

Overview of 6PE/VPE

Multiple techniques are available to integrate IPv6 services over service provider core backbones:

- Dedicated IPv6 network running over various data link layers
- Dual-stack IPv4-IPv6 backbone
- Existing MPLS backbone leverage

These solutions are deployed on service providers' backbones when the amount of IPv6 traffic and the revenue generated are in line with the necessary investments and the agreed-upon risks. Conditions are favorable for the introduction of native IPv6 services, from the edge, in a scalable way, without any IPv6 addressing restrictions and without putting a well-controlled IPv4 backbone in jeopardy. Backbone stability is essential for service providers that have recently stabilized their IPv4 infrastructure.

Service providers running an MPLS/IPv4 infrastructure follow similar trends because several integration scenarios that offer IPv6 services on an MPLS network are possible. Cisco Systems has specially developed Cisco 6PE or IPv6 Provider Edge Router over MPLS, to meet all those requirements.

Inter-AS support for 6PE requires support of Border Gateway Protocol (BGP) to enable the address families and to allocate and distribute PE and ASBR labels.

Note Cisco IOS XR displays actual IPv4 next-hop addresses for IPv6 labeled-unicast and VPNv6 prefixes. IPv4-mapped-to-IPv6 format is not supported.

Benefits of 6PE/VPE

Service providers who currently deploy MPLS experience these benefits of Cisco 6PE/VPE:

- Minimal operational cost and risk—No impact on existing IPv4 and MPLS services.
- Provider edge routers upgrade only—A 6PE/VPE router can be an existing PE router or a new one dedicated to IPv6 traffic.
- No impact on IPv6 customer edge routers—The ISP can connect to any customer CE running Static, IGP or EGP.
- Production services ready—An ISP can delegate IPv6 prefixes.

- IPv6 introduction into an existing MPLS service—6PE/VPE routers can be added at any time.

IPv6 on the Provider Edge and Customer Edge Routers

Service Provider Edge Routers

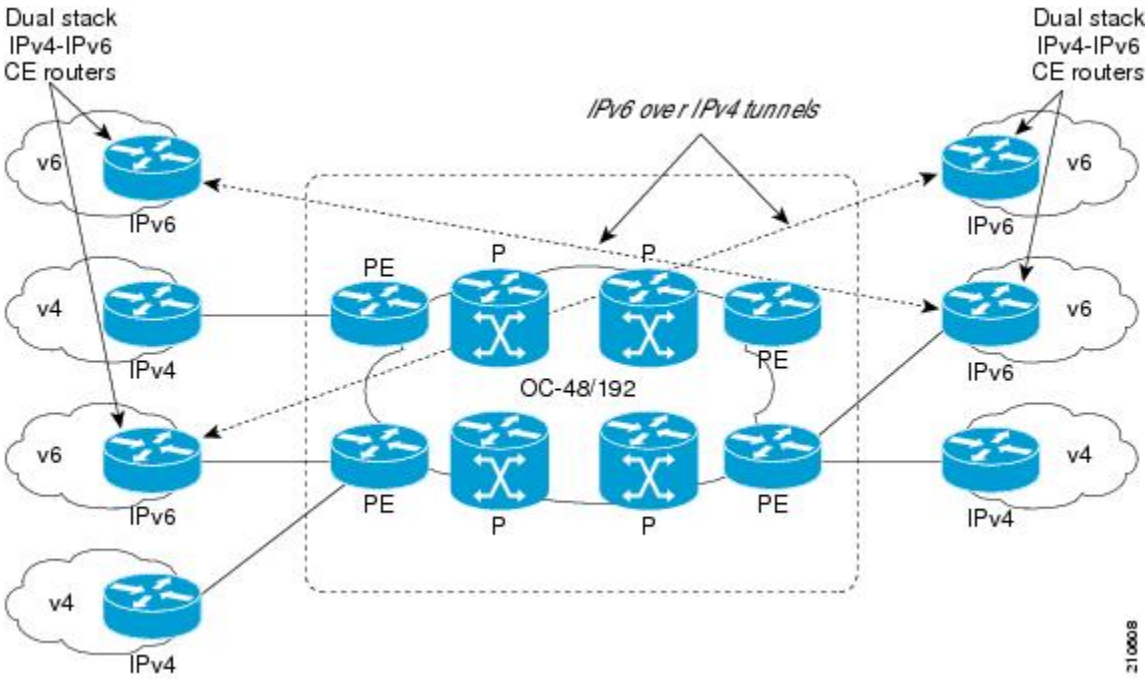
6PE is particularly applicable to service providers who currently run an MPLS network. One of its advantages is that there is no need to upgrade the hardware, software, or configuration of the core network, and it eliminates the impact on the operations and the revenues generated by the existing IPv4 traffic. MPLS is used by many service providers to deliver services to customers. MPLS as a multiservice infrastructure technology is able to provide layer 3 VPN, QoS, traffic engineering, fast re-routing and integration of ATM and IP switching.

Customer Edge Routers

Using tunnels on the CE routers is the simplest way to deploy IPv6 over MPLS networks. It has no impact on the operation or infrastructure of MPLS and requires no changes to the P routers in the core or to the PE routers. However, tunnel meshing is required as the number of CEs to connect increases, and it is difficult to delegate a global IPv6 prefix for an ISP.

The following figure illustrates the network architecture using tunnels on the CE routers.

Figure 8: IPv6 Using Tunnels on the CE Routers



IPv6 Provider Edge Multipath

Internal and external BGP multipath for IPv6 allows the IPv6 router to load balance between several paths (for example, same neighboring autonomous system (AS) or sub-AS, or the same metric) to reach its destination. The 6PE multipath feature uses multiprotocol internal BGP (MP-IBGP) to distribute IPv6 routes over the MPLS IPv4 core network and to attach an MPLS label to each route.

When MP-IBGP multipath is enabled on the 6PE router, all labeled paths are installed in the forwarding table with MPLS information (label stack) when MPLS information is available. This functionality enables 6PE to perform load balancing.

OSPFv3 6VPE

The Open Shortest Path First version 3 (OSPFv3) IPv6 VPN Provider Edge (6VPE) feature adds VPN routing and forwarding (VRF) and provider edge-to-customer edge(PE-CE) routing support to Cisco IOS XR OSPFv3 implementation. This feature allows:

- Multiple VRF support per OSPFv3 routing process
- OSPFV3 PE-CE extensions

Multiple VRF Support

OSPFv3 supports multiple VRFs in a single routing process that allows scaling to tens and hundreds of VRFs without consuming too much route processor (RP) resources.

Multiple OSPFv3 processes can be configured on a single router. In large-scale VRF deployments, this allows partition VRF processing across multiple RPs. It is also used to isolate default routing table or high impact VRFs from the regular VRFs. It is recommended to use a single process for all the VRFs. If needed, a second OSPFv3 process must be configured for IPv6 routing.

Note The maximum of four OSPFv3 processes are supported.

OSPFv3 PE-CE Extensions

IPv6 protocol is being vastly deployed in today's customer networks. Service Providers (SPs) need to be able to offer Virtual Private Network (VPN) services to their customers for supporting IPv6 protocol, in addition to the already offered VPN services for IPv4 protocol.

In order to support IPv6, routing protocols require additional extensions for operating in the VPN environment. Extensions to OSPFv3 are required in order for OSPFv3 to operate at the PE-CE links.

VRF Lite

VRF lite feature enables VRF deployment without BGP or MPLS based backbone. In VRF lite, the PE routers are directly connected using VRF interfaces. For OSPFv3, the following needs to operate differently in the VRF lite scenario, as opposed to the deployment with BGP or MPLS backbone:

- DN bit processing—In VRF lite environment, the DN bit processing is disabled.
- ABR status—In VRF context (except default VRF), OSPFv3 router is automatically set as an ABR, regardless to it’s connectivity to area 0. This automatic ABR status setting is disabled in the VRF lite environment.

Note To enable VRF Lite, issue the **capability vrf-lite** command in the OSPFv3 VRF configuration submode.

How to Implement 6PE/VPE

This section includes these implementation procedures:

Configuring 6PE/VPE

This task describes how to configure 6PE/VPE on PE routers to transport the IPv6 prefixes across the IPv4 cloud.

Ensure that you configure 6PE/VPE on PE routers participating in both the IPv4 cloud and IPv6 clouds.

- For 6PE, you can use all routing protocols supported on Cisco IOS XR software such as BGP, OSPF, IS-IS, EIGRP, RIP, and Static to learn routes from both clouds. However, for 6VPE, you can use only the BGP, EIGRP and Static routing protocols to learn routes. Also, 6VPE supports OSPFv3 routing protocol between PE and CE routers.
- BGP uses the **per-vrf** label mode for transporting local and redistributed IP prefixes. Before IOS XR Release 7.5.3, BGP assigned a random label for the prefixes. Starting from Release 7.5.3, BGP assigns a label value of **2**, the IPv6 Explicit NULL Label, for the same prefixes.

SUMMARY STEPS

1. **configure**
2. **router bgp** *as-number*
3. **neighbor** *ip-address*
4. **remote-as** *as-number*
5. **address-family ipv6 labeled-unicast**
6. **exit**
7. **exit**
8. **address-family ipv6 unicast**
9. **allocate-label** [**all** | **route-policy** *policy_name*]
10. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

```
RP/0/RSP0/CPU0:router# configure
```

Enters the Global Configuration mode.

Step 2 **router bgp** *as-number*

Example:

```
RP/0/RSP0/CPU0:router(config)# router bgp 1
```

Enters the number that identifies the autonomous system (AS) in which the router resides.

Range for 2-byte numbers is 1 to 65535. Range for 4-byte numbers is 1.0 to 65535.65535.

Step 3 **neighbor** *ip-address*

Example:

```
RP/0/RSP0/CPU0:router(config-bgp)# neighbor 10.0.0.1
```

Enters neighbor configuration mode for configuring Border Gateway Protocol (BGP) routing sessions.

Step 4 **remote-as** *as-number*

Example:

```
RP/0/RSP0/CPU0:router(config-bgp-nbr)# remote-as 100
```

Creates a neighbor and assigns a remote autonomous system number to it.

Step 5 **address-family ipv6 labeled-unicast**

Example:

```
RP/0/RSP0/CPU0:router(config-bgp-nbr)# address-family ipv6 labeled-unicast
```

Specifies IPv6 labeled-unicast address prefixes.

Note This option is also available in IPv6 neighbor configuration mode and VRF neighbor configuration mode.

Step 6 **exit**

Example:

```
RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# exit
```

Exits BGP address-family submode.

Step 7 **exit**

Example:

```
RP/0/RSP0/CPU0:router(config-bgp-nbr)# exit
```

Exits BGP neighbor submode.

Step 8 **address-family ipv6 unicast**

Example:

```
RP/0/RSP0/CPU0:router(config-bgp)# address-family ipv6 unicast
```

Specifies IPv6 unicast address prefixes.

Step 9 **allocate-label** [**all** | **route-policy** *policy_name*]

Example:

```
RP/0/RSP0/CPU0:router(config-bgp-af)# allocate-label all
```

Allocates MPLS labels for specified IPv4 unicast routes.

Note The **route-policy** keyword provides finer control to filter out certain routes from being advertised to the neighbor.

Step 10 Use the **commit** or **end** command.

commit - Saves the configuration changes and remains within the configuration session.

end - Prompts user to take one of these actions:

- **Yes** - Saves configuration changes and exits the configuration session.
- **No** - Exits the configuration session without committing the configuration changes.
- **Cancel** - Remains in the configuration mode, without committing the configuration changes.

Configuring PE to PE Core

This task describes how to configure a Provider Edge (PE) to PE Core.

For information on configuring VPN Routing and Forwarding (VRF), refer to the *Implementing BGP* module of the *Routing Configuration Guide for Cisco ASR 9000 Series Routers*.

SUMMARY STEPS

1. **configure**
2. **router bgp**
3. **address-family vpnv6 unicast**
4. **bgp dampening** [*half-life* [*reuse suppress max-suppress-time*] | **route-policy** *route-policy-name*]
5. **bgp client-to-client reflection { cluster-id | disable }**
6. **neighbor** *ip-address*
7. **remote-as** *as-number*
8. **description** *text*
9. **password** { **clear** | **encrypted** } *password*
10. **shutdown**
11. **timers** *keepalive hold-time*
12. **update-source type** *interface-id*
13. **address-family vpnv6 unicast**
14. **route-policy** *route-policy-name* { **in** | **out** }
15. **exit**
16. **vrf** *vrf-name*
17. **rd** { *as-number : nn* | *ip-address : nn* | **auto** }
18. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

```
RP/0/RSP0/CPU0:router# configure
```

Enters the Global Configuration mode.

Step 2 **router bgp**

Example:

```
RP/0/RSP0/CPU0:router(config)# router bgp 10
```

Specifies the BGP AS number and enters the BGP configuration mode, allowing you to configure the BGP routing process.

Step 3 **address-family vpnv6 unicast**

Example:

```
RP/0/RSP0/CPU0:router(config-bgp)# address-family vpnv6 unicast
```

Specifies the vpnv6 address family and enters address family configuration submode.

Step 4 **bgp dampening** [*half-life* [*reuse suppress max-suppress-time*] | **route-policy** *route-policy-name*]

Example:

```
RP/0/RSP0/CPU0:router(config-bgp-af)# bgp dampening 30 1500 10000 120
```

Configures BGP dampening for the specified address family.

Step 5 **bgp client-to-client reflection { cluster-id | disable }**

Example:

```
RP/0/RSP0/CPU0:router(config-bgp-af)# bgp client-to-client
reflection disable
```

Configures client to client route reflection.

Step 6 **neighbor** *ip-address*

Example:

```
RP/0/RSP0/CPU0:router(config-bgp)# neighbor 10.0.0.1
```

Places the router in neighbor configuration mode for BGP routing and configures the neighbor IP address as a BGP peer.

Step 7 **remote-as** *as-number*

Example:

```
RP/0/RSP0/CPU0:router(config-bgp-nbr)# remote-as 100
```

Creates a neighbor and assigns a remote autonomous system number to it.

Step 8 **description** *text*

Example:

```
RP/0/RSP0/CPU0:router(config-bgp-nbr)# description neighbor 172.16.1.1
```

Provides a description of the neighbor. The description is used to save comments and does not affect software function.

Step 9 **password** { **clear** | **encrypted** } *password*

Example:

```
RP/0/RSP0/CPU0:router(config-bgp-nbr)# password encrypted 123abc
```

Enables Message Digest 5 (MD5) authentication on the TCP connection between the two BGP neighbors.

Step 10 **shutdown**

Example:

```
RP/0/RSP0/CPU0:router(config-bgp-nbr)# router bgp 1
```

Terminates any active sessions for the specified neighbor and removes all associated routing information.

Step 11 **timers** *keepalive hold-time*

Example:

```
RP/0/RSP0/CPU0:router(config-bgp-nbr)# timers 12000 200
```

Set the timers for the BGP neighbor.

Step 12 **update-source type** *interface-id*

Example:

```
RP/0/RSP0/CPU0:router(config-bgp-nbr)# update-source TenGigE 0/1/5/0
```

Allows iBGP sessions to use the primary IP address from a specific interface as the local address when forming an iBGP session with a neighbor.

Step 13 **address-family vpnv6 unicast**

Example:

```
RP/0/RSP0/CPU0:router(config-bgp-nbr)# address-family vpvn6 unicast
```

Enters VPN neighbor address family configuration mode.

Step 14 **route-policy** *route-policy-name* { **in** | **out** }

Example:

```
RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# route-policy pe-pe-vpn-out out
```

Specifies a routing policy for an outbound route. The policy can be used to filter routes or modify route attributes.

Step 15 **exit**

Example:

```
RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# exit
```

Exits address family configuration and neighbor submode.

Step 16 **vrf** *vrf-name*

Example:

```
RP/0/RSP0/CPU0:router(config-bgp)# vrf vrf-pe
```

Configures a VRF instance.

Step 17 **rd** { *as-number : nn* | *ip-address : nn* | **auto** }

Example:

```
RP/0/RSP0/CPU0:router(config-bgp-vrf)# rd 345:567
```

Configures the route distinguisher.

Use the auto keyword if you want the router to automatically assign a unique RD to the VRF.

Step 18 Use the **commit** or **end** command.

commit - Saves the configuration changes and remains within the configuration session.

end - Prompts user to take one of these actions:

- **Yes** - Saves configuration changes and exits the configuration session.
- **No** - Exits the configuration session without committing the configuration changes.
- **Cancel** - Remains in the configuration mode, without committing the configuration changes.

Configuring OSPFv3 as the Routing Protocol Between the PE and CE Routers

Perform this task to configure provider edge (PE)-to-customer edge (CE) routing sessions that use Open Shortest Path First version 3 (OSPFv3).

SUMMARY STEPS

1. **configure**
2. **router ospfv3** *process-name*

3. **vrf** *vrf-name*
4. **capability** *vrf-lite*
5. **router-id** {*router-id* | *type interface-path-id* }
6. **domain-id type** { **0005** | **0105** | **0205** | **8005** } **value** *domain-id*
7. Do one of the following:
   - **redistribute bgp** *process-id* [**metric** *metric-value*] [**metric-type** {**1** | **2**}] [**route-policy12route-policy** *policy-name*] [**tag** *tag-value*]
   - **redistribute connected** [**metric** *metric-value*] [**metric-type** {**1** | **2**}] [**route-policy***policy-name*] [**tag** *tag-value*]
   - **redistribute ospf** *process-id [* **match** {**external** [**1** | **2**] | **internal** | **nssa-external** [**1** | **2**]}] [**metric** *metric-value*] [**metric-type** {**1** | **2**}] [**route-policy** *policy-name*] [**tag** *tag-value*]
   - **redistribute static**[**metric** *metric-value*] [**metric-type {1 | 2}] [route-policy** *policy-name]* [**tag** *policy-name*] [**tag** *tag-value*]
   - **redistribute eigrp** *process-id* **[match {external [1 | 2] | internal | nssa-external [1 | 2]]} [metric** *metric-value]* **[metric-type {1 | 2}] [route-policy** *policy-name*] [**tag** *tag-value*]
   - **redistribute rip** [**metric** *metric-value*] [**metric-type {1 | 2}] [route-policy** *policy-name]* [**tag** *tag-value*]
8. **area** *area-id*
9. **interface** {*type interface-path-id*}
10. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

```
RP/0/RSP0/CPU0:router# configure
```

Enters the Global Configuration mode.

Step 2 **router ospfv3** *process-name*

Example:

```
RP/0/RSP0/CPU0:router(config)# router ospfv3 109
```

Enters OSPF configuration mode allowing you to configure the OSPF version 3 routing process.

Step 3 **vrf** *vrf-name*

Example:

```
RP/0/RSP0/CPU0:router(config-ospf)# vrf vrf_1
```

Configures a VPN routing and forwarding (VRF) instance and enters VRF configuration mode for OSPF routing.

Step 4 **capability** *vrf-lite*

Example:

```
RP/0/RSP0/CPU0:router(config-ospf-vrf)# capability vrf-lite
```

Enables VRF Lite feature.

Step 5 **router-id** {*router-id* | *type interface-path-id* }

Example:

```
RP/0/RSP0/CPU0:router(config-ospf-vrf)# router-id 172.20.10.10
```

Configures the router ID for the OSPF routing process.

Note Router ID configuration is required for each VRF.

Step 6 **domain-id type** { **0005** | **0105** | **0205** | **8005** } **value** *domain-id*

Example:

```
RP/0/RSP0/CPU0:router(config-ospf-vrf)# domain-id type 0005 value CAFE00112233
```

Specifies the domain ID.

Step 7 Do one of the following:

- **redistribute bgp** *process-id* [**metric** *metric-value*] [**metric-type** {**1** | **2**}] [**route-policy12route-policy** *policy-name*] [**tag** *tag-value*]
- **redistribute connected** [**metric** *metric-value*] [**metric-type** {**1** | **2**}] [**route-policy***policy-name*] [**tag** *tag-value*]
- **redistribute ospf** *process-id [* **match** {**external** [**1** | **2**] | **internal** | **nssa-external** [**1** | **2**]}] [**metric** *metric-value*] [**metric-type** {**1** | **2**}] [**route-policy** *policy-name*] [**tag** *tag-value*]
- **redistribute static**[**metric** *metric-value*] [**metric-type {1 | 2}] [route-policy** *policy-name]* [**tag** *policy-name*] [**tag** *tag-value*]
- **redistribute eigrp** *process-id* **[match {external [1 | 2] | internal | nssa-external [1 | 2]]} [metric** *metric-value]* **[metric-type {1 | 2}] [route-policy** *policy-name*] [**tag** *tag-value*]
- **redistribute rip** [**metric** *metric-value*] [**metric-type {1 | 2}] [route-policy** *policy-name]* [**tag** *tag-value*]

Example:

```
RP/0/RSP0/CPU0:router(config-ospf-vrf)# redistribute connected
```

Causes routes to be redistributed into OSPF. The routes that can be redistributed into OSPF are:

- Border Gateway Protocol (BGP)
- Connected
- Enhanced Interior Gateway Routing Protocol (EIGRP)
- OSPF
- Static
- Routing Information Protocol (RIP)

Step 8 **area** *area-id*

Example:

```
RP/0/RSP0/CPU0:router(config-ospf-vrf)# area 0
```

Configures the OSPF area as area 0.

Step 9 **interface** {*type interface-path-id*}

Example:

```
RP/0/RSP0/CPU0:router(config-ospf-vrf-ar)# interface GigabitEthernet 0/3/0/0
```

Associates interface GigabitEthernet 0/3/0/0 with area 0.

Step 10 Use the **commit** or **end** command.

commit - Saves the configuration changes and remains within the configuration session.

end - Prompts user to take one of these actions:

- **Yes** - Saves configuration changes and exits the configuration session.
- **No** - Exits the configuration session without committing the configuration changes.
- **Cancel** - Remains in the configuration mode, without committing the configuration changes.

Configuration Examples for 6PE/VPE

This section includes the following configuration example:

Configuring 6PE on a PE Router: Example

This sample configuration shows the configuration of 6PE on a PE router:

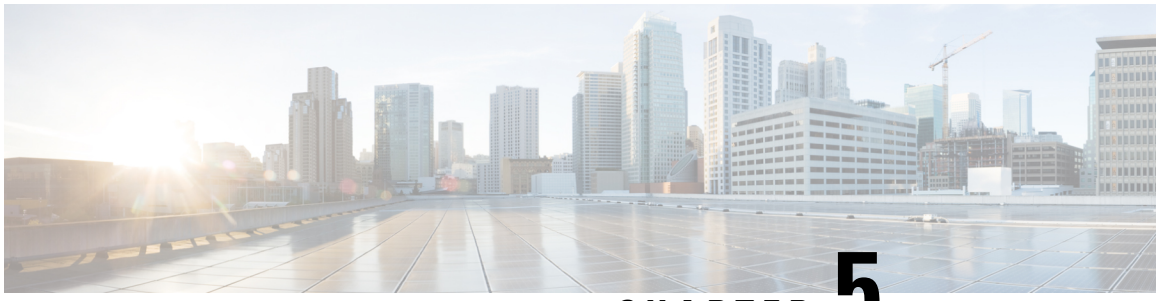
```
interface TenGigE0/3/0/0
 ipv6 address 2001::1/64
!
router isis ipv6-cloud
 net 49.0000.0000.0001.00
 address-family ipv6 unicast
  single-topology
 interface TenGigE0/3/0/0
  address-family ipv6 unicast
  !
!
router bgp 55400
 bgp router-id 54.6.1.1
 address-family ipv4 unicast
 !
 address-family ipv6 unicast
  network 55:5::/64
  redistribute connected
  redistribute isis ipv6-cloud
   allocate-label all

 !
 neighbor 34.4.3.3
  remote-as 55400
  address-family ipv4 unicast
```

```
  !
  address-family ipv6 labeled-unicast
```

Configuring 6VPE on a PE Router: Example

This sample configuration shows the configuration of 6VPE on a PE router:

```
vrf vpn1
 address-family ipv6 unicast
  import route-target
   200:2
  !
  export route-target
   200:2

interface Loopback0
 ipv4 address 10.0.0.1 255.255.255.255

interface GigabitEthernet0/0/0/1
 vrf  vpn1
 ipv6 address 2001:c003:a::2/64

router bgp 1
 bgp router-id 10.0.0.1
 bgp redistribute-internal
 bgp graceful-restart
 address-family ipv4 unicast
!

address-family vpnv6 unicast
 !
 neighbor 10.0.0.2                >>>> Remote peer loopback address.
  remote-as 1
  update-source Loopback0
  address-family ipv4 unicast
  !
  address-family vpnv6 unicast
   route-policy pass-all in
   route-policy pass-all out
 !

  vrf vpn1
   rd 100:2
   bgp router-id 140.140.140.140
   address-family ipv6 unicast
   redistribute connected
 !

 neighbor 2001:c003:a::1
   remote-as 6502
   address-family ipv6 unicast
   route-policy pass-all in
   route-policy pass-all out
 !
```



CHAPTER 5

Implementing Generic Routing Encapsulation

Generic Routing Encapsulation (GRE) is a tunneling protocol developed by Cisco Systems that encapsulates a wide variety of network layer protocols inside virtual point-to-point links over an Internet Protocol internetwork.

Feature History for Configuring Link Bundling on Cisco IOS XR Software

Release	Modification
Release 4.3.0	These feature were supported on the Cisco ASR 9000 Series Aggregation Services Routers: • MPLS/L3VPNoGRE on ASR 9000 Enhanced Ethernet Line Card and Cisco ASR 9000 Series SPA Interface Processor-700 • RSVP/TEoGRE on ASR 9000 Enhanced Ethernet Line Card and Cisco ASR 9000 Series SPA Interface Processor-700 • VRF aware GRE on ASR 9000 Enhanced Ethernet Line Card and Cisco ASR 9000 Series SPA Interface Processor-700 • L2VPN (VPWS and VPLS) on GRE for ASR 9000 Enhanced Ethernet Line Card only
Release 5.1.1	Support for GRE Tunnel Key and Tunnel Key-Ignore was introduced.
Release 5.2.2	Support for GRE tunnel on an IPv6 transport network.
Release 5.3.2	Support for GRE IPv4 Transport Over MPLS was introduced.
Release 6.0.1	Support for GRE IPv6 Transport Over MPLS was introduced.

Prerequisites for Configuring Generic Routing Encapsulation

Before configuring Link Bundling, be sure that these tasks and conditions are met:

- You must be in a user group associated with a task group that includes the proper task IDs. The command reference guides include the task IDs required for each command.

If you suspect user group assignment is preventing you from using a command, contact your AAA administrator for assistance.

Information About Generic Routing Encapsulation

To implement the GRE feature, you must understand these concepts:

GRE Overview

Generic Routing Encapsulation (GRE) tunneling protocol provides a simple generic approach to transport packets of one protocol over another protocol by means of encapsulation.

GRE encapsulates a payload, that is, an inner packet that needs to be delivered to a destination network inside an outer IP packet. GRE tunnel endpoints send payloads through GRE tunnels by routing encapsulated packets through intervening IP networks. Other IP routers along the way do not parse the payload (the inner packet); they only parse the outer IP packet as they forward it towards the GRE tunnel endpoint. Upon reaching the tunnel endpoint, GRE encapsulation is removed and the payload is forwarded to it's ultimate destination.

MPLS networks provide VPN functionality by tunneling customer data through public networks using routing labels. Service Providers (SP) provide MPLS L3VPN, 6PE/6VPE and L2VPN services to their customers who have interconnected private networks.

MPLS and L3VPN are supported over regular interfaces on Cisco ASR 9000 Series Aggregation Services Routers through GRE tunnels over an IPv4 transport network. MPLS support is extended over IPv4 GRE tunnels between routers as the provider core may not be fully MPLS aware.

GRE Features

The following sections list the GRE features:

Note An IPv6 GRE tunnel does not support features that involve transport of MPLS packets through a GRE tunnel.

MPLS/L3VPN over GRE

The MPLS VPN over GRE feature provides a mechanism for tunneling Multiprotocol Label Switching (MPLS) packets over a non-MPLS network. This feature utilizes MPLS over generic routing encapsulation (MPLSoGRE) to encapsulate MPLS packets inside IP tunnels. The encapsulation of MPLS packets inside IP tunnels creates a virtual point-to-point link across non-MPLS networks.

L3VPN over GRE basically means encapsulating L3VPN traffic in GRE header and its outer IPv4 header with tunnel destination and source IP addresses after imposing zero or more MPLS labels, and transporting it across the tunnel over to the remote tunnel end point. The incoming packet can be a pure IPv4 packet or an MPLS packet. If the incoming packet is IPv4, the packet enters the tunnel through a VRF interface, and if the incoming packet is MPLS, then the packet enters through an MPLS interface. In the IPv4 case, before encapsulating in the outer IPv4 and GRE headers, a VPN label corresponding to the VRF prefix and any IGP

label corresponding to the IGP prefix of the GRE tunnel destination is imposed on the packet. In the case of MPLS, the top IGP label is swapped with any label corresponding to the GRE tunnel destination address.

PE-to-PE Tunneling

The provider-edge-to-provider-edge (PE-to-PE) tunneling configuration provides a scalable way to connect multiple customer networks across a non-MPLS network. With this configuration, traffic that is destined to multiple customer networks is multiplexed through a single GRE tunnel.

Note A similar nonscalable alternative is to connect each customer network through separate GRE tunnels (for example, connecting one customer network to each GRE tunnel).

As shown in the following figure, the PE devices assign VPN routing and forwarding (VRF) numbers to the customer edge (CE) devices on each side of the non-MPLS network.

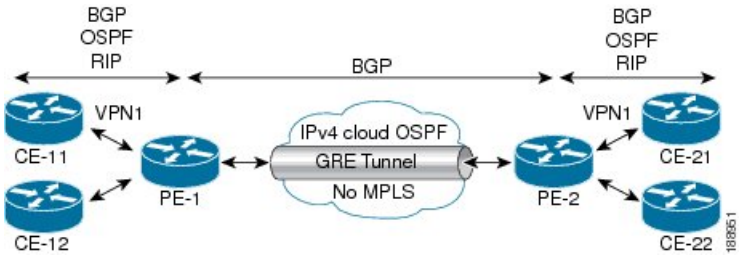
The PE devices use routing protocols such as Border Gateway Protocol (BGP), Open Shortest Path First (OSPF), or Routing Information Protocol (RIP) to learn about the IP networks behind the CE devices. The routes to the IP networks behind the CE devices are stored in the associated CE device's VRF routing table.

The PE device on one side of the non-MPLS network uses the routing protocols (that operate within the non-MPLS network) to learn about the PE device on the other side of the non-MPLS network. The learned routes that are established between the PE devices are then stored in the main or default routing table.

The opposing PE device uses BGP to learn about the routes that are associated with the customer networks that are behind the PE devices. These learned routes are not known to the non-MPLS network.

The following figure shows BGP defining a static route to the BGP neighbor (the opposing PE device) through the GRE tunnel that spans the non-MPLS network. Because routes that are learned by the BGP neighbor include the GRE tunnel next hop, all customer network traffic is sent using the GRE tunnel.

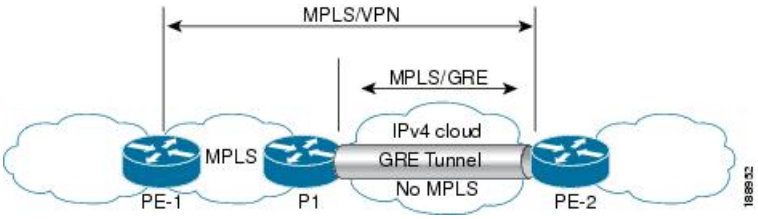
Figure 9: PE-to-PE Tunneling



P-to-PE Tunneling

As shown in the following figure, the provider-to-provider-edge (P-to-PE) tunneling configuration provides a way to connect a PE device (P1) to an MPLS segment (PE-2) across a non-MPLS network. In this configuration, MPLS traffic that is destined to the other side of the non-MPLS network is sent through a single GRE tunnel.

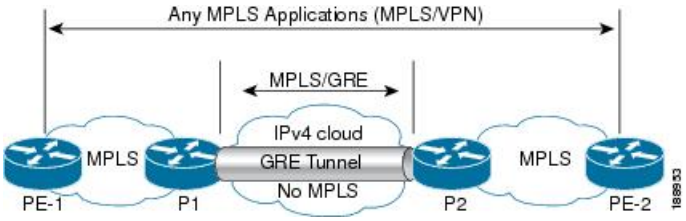
Figure 10: P-to-PE Tunneling



P-to-P Tunneling

As shown in the following figure, the provider-to-provider (P-to-P) configuration provides a method of connecting two MPLS segments (P1 to P2) across a non-MPLS network. In this configuration, MPLS traffic that is destined to the other side of the non-MPLS network is sent through a single GRE tunnel.

Figure 11: P-to-P Tunneling



6PE/6VPE

Service Providers (SPs) use a stable and established core with IPv4/MPLS backbone for providing IPv4 VPN services. The 6PE/6VPE feature facilitates SPs to offer IPv6 VPN services over this backbone without an IPv6 core. The provide edge (PE) routers run MP-iBGP (Multi-Protocol iBGP) to advertise v6 reachability and v6 label distribution. For 6PE, the labels are allocated per IPv6 prefix learnt from connected customer edge (CE) routers and for 6VPE, the PE router can be configured to allocate labels on a per-prefix or per-CE/VRF level.

6PE/6VPE over GRE

While IPv4/MPLS allows SPs to transport IPv6 traffic across IPv4 core (IPv6 unaware), MPLS over GRE allows MPLS traffic to be tunneled through MPLS unaware networks. These two features together facilitate IPv6 traffic to be transported across IPv6 as well as MPLS unaware core segments. Only the PE routers need to be aware of MPLS and IPv6 (Dual stack).

The 6PE/6VPE over GRE feature allows the use of IPv4 GRE tunnels to provide IPv6 VPN over MPLS functionality to reach the destination v6 prefixes via the BGP next hop through MPLS & IPv6 unaware core.

MPLS Forwarding

When IPv6 traffic is received from one customer site, the ingress PE device uses MPLS to tunnel IPv6 VPN packets over the backbone toward the egress PE device identified as the BGP next hop. The ingress PE device prefixes the IPv6 packets with the outer and inner labels before placing the packet on the egress interface.

Under normal operation, a P device along the forwarding path does not lookup the frame beyond the first label. The P device either swaps the incoming label with an outgoing one or removes the incoming label if the next device is a PE device. Removing the incoming label is called penultimate hop popping. The remaining

label (BGP label) is used to identify the egress PE interface toward the customer site. The label also hides the protocol version (IPv6) from the last P device, which it would otherwise need to forward an IPv6 packet.

A P device is ignorant of the IPv6 VPN routes. The IPv6 header remains hidden under one or more MPLS labels. When the P device receives an MPLS-encapsulated IPv6 packet that cannot be delivered, it has two options. If the P device is IPv6 aware, it exposes the IPv6 header, builds an Internet Control Message Protocol (ICMP) for IPv6 message, and sends the message, which is MPLS encapsulated, to the source of the original packet. If the P device is not IPv6 aware, it drops the packet.

6PE/6VPE over GRE

As discussed earlier, 6PE/6VPE over GRE basically means enabling IPv6/IPv6 VPN over MPLS over GRE.

The ingress PE device uses IPv4 generic routing encapsulation (GRE) tunnels combined with 6PE/6VPE over MPLS to tunnel IPv6 VPN packets over the backbone toward the egress PE device identified as the BGP next hop.

The PE devices establish MP-iBGP sessions and MPLS LDP sessions just as in the case of 6PE/6VPE. The difference here is that these sessions are established over GRE tunnels, which also means that the PEs are just one IGP hop away. The P routers in the tunnel path only need to forward the traffic to the tunnel destination, which is an IPv4 address.

This is how the IPv6 LSP is setup for label switching the IPv6 traffic:

- After the LDP and BGP sessions are established, the PEs exchange IPv6 prefixes that they learn from the CEs and the corresponding IPv6 labels, just as in the case of IPv4 VPN.
- The IPv6 labels occupy the inner most position in the label stack.
- The IPv4 labels corresponding to the PE IPv4 addresses occupy the outer position in the stack.
- When IPv6 traffic needs to be forwarded from PE1 to PE2, the outer PE2 IPv4 label is used to label switch the traffic to PE2, and the inner IPv6 label is used to send the packet out of the interface connected to the CE.

GRE Tunnel Key

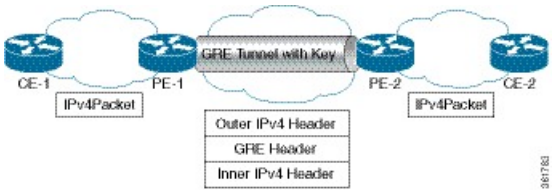
The GRE Tunnel Key feature enables the encapsulation router to add a four-byte key, as part of the GRE header, during encapsulation. In the decapsulation router, the GRE key of an incoming packet should match the key value configured under the GRE tunnel. During decapsulation, if a mismatch between the key value of the incoming GRE packet and the key value configured under the GRE tunnel is identified, the incoming packet is dropped.

Note

- GRE tunnel key feature is supported only on Cisco ASR 9000 Enhanced Ethernet line cards. It is mandatory to have ingress and egress line cards as Enhanced Ethernet line cards.
- Either the same key or different keys can be configured under multiple GRE tunnels for a given router. However, more than one tunnel, having the same tunnel source and destination but a different tunnel key is not supported because the source and destination pair for various configured tunnels must be unique irrespective of the key value. Also, two tunnels with the same tunnel source and destination, but one tunnel being with key and the other tunnel being without key is not supported.
- Different traffic streams passing through the same GRE tunnel contains the same GRE key configured for that tunnel.
- Use the **tunnel key** command to configure the key value at both ends of a GRE tunnel.

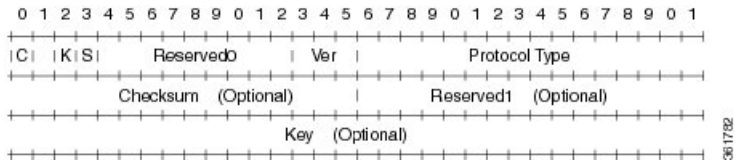
The following figure shows a simple representation of the GRE tunnel key configuration:

Figure 12: GRE Tunnel with Key



The following figure shows the complete format of the GRE header with the key field:

Figure 13: GRE Header



GRE Tunnel Key-Ignore

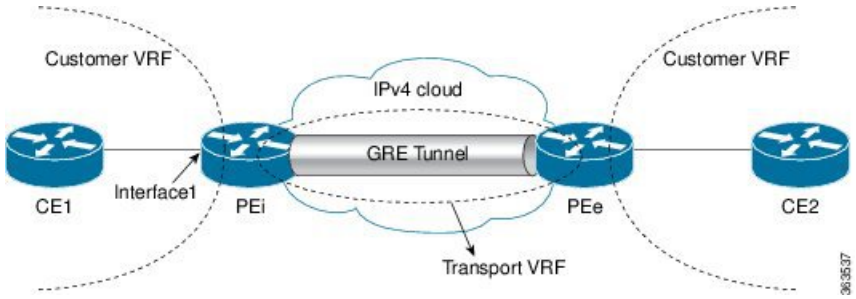
If a GRE key is configured on only one endpoint router of a GRE tunnel, the other router that has no GRE key configured discards any incoming tunnel packet that has a GRE key. To enable this router to ignore GRE keys and accept incoming data plane packets on the GRE tunnel, run the **tunnel key-ignore** command. Control plane packets over a GRE tunnel are accepted only if there is no GRE tunnel key configured on both the tunnel endpoints or both the endpoints are configured with a GRE key and the control plane packet passes the GRE key validation. Hence, in the above scenario, both the routers discard any incoming control plane packets from the GRE tunnel.

Note Do not configure a GRE key on the GRE tunnel endpoint router if you have configured the router to ignore GRE keys. Configuring a GRE key overrides the **tunnel key-ignore** command and thus cancels the skipping of GRE key validation. This results in the router accepting from the incoming tunnel traffic only those packets that have the matching GRE key.

GRE tunnel in VRF domains

You can configure an IPv4/IPv6 GRE tunnel between two interfaces that belong to a Virtual Forwarding and Routing (VRF) instance. This contains or limits the tunnel path within this specific VRF instance. For example, packets can be sent internally within a default or non-default VRF instance separated through an intermediate VRF that contains the GRE tunnel.

Figure 14: GRE tunnel in a VRF instance



In the above topology, a GRE tunnel is configured in the core network, which is an IPv4 cloud. For packets entering through Interface1, the provider edge (PE) devices PEi and PEe are the tunnel head and tunnel exit respectively.

The VRF configured on Interface1 is the customer VRF. Packets entering this interface are routed using this customer VRF to the tunnel. The routing by the customer VRF is called inner IP packet routing. You can configure the tunnel to be visible to the customer VRF instance using the **vrf** *vrf-name* command. This enables only the configured VRF instance to use the tunnel, that is, forward traffic from PEi into this tunnel and also receive all incoming PEi tunnel packets.

The VRF configured on the tunnel using the **tunnel vrf** command is the transport VRF. The packet entering the tunnel is encapsulated with the tunnel source and destination addresses. The transport VRF routes this encapsulated payload between the tunnel endpoints. The routing by the transport VRF is the outer IP packet routing. If no transport VRF is configured for the tunnel, the PEi device looks up the tunnel endpoint addresses in the default VRF instance, that is, the global routing table.

Restrictions on a GRE tunnel

The following restrictions are applicable for a GRE tunnel:

- GRE over BVI is not supported.
- MPLS packets cannot be transported within an IPv6 GRE tunnel. Therefore, the following features are not supported on an IPv6 GRE tunnel:
  - MPLS/L3VPN over GRE
  - 6PE/6VPE
  - 6PE/6VPE over GRE
- Multicast packets cannot be transported within an IPv6 GRE tunnel.
- Multicast packets cannot be transported within an IPv4 GRE tunnel that is configured in a transport VRF.
- Keep-Alive packets are not supported on an IPv6 GRE tunnel. You can use the Bidirectional Forwarding Detection (BFD) protocol to detect link failures in an IPv6 GRE tunnel.
- The IPv4 addresses are mandatory for configuring GRE tunnels under the VRF, as this would ensure the traffic flows through the tunnel in an expected manner. Use either an IP unnumbered interface or a loopback interface belonging to that VRF for establishing the GRE tunnels under a VRF. Though the tunnel may come up without the aforementioned configuration, the traffic may not pass over the GRE tunnel, since the IP information on the tunnel interface is not available for forwarding the traffic correctly.

Also, for the VRF information to be written in hardware database the IP information is required. Therefore, the IP unnumbered GRE tunnels may not work as expected as they may not forward traffic on the device.

GRE IPv4/IPv6 Transport Over MPLS

The Generic Routing Encapsulation (GRE) IPv4/IPv6 transport over Multiprotocol Label Switching (MPLS) feature provides a mechanism to configure GRE tunnels, where the tunnel destination IPv4/IPv6 address is reachable through an MPLS label switched path (LSP). With this feature, IPv4, IPv6, routing protocols - OSPF, ISIS, and L2VPN and L3VPN packets are accepted as payload packets for GRE encapsulation . IPv4/IPv6 is supported as the GRE delivery protocol.

This feature overcomes the restriction of not being able to configure the tunnel destination endpoint through an MPLS LSP during tunnel configuration.

The GRE IPv4/IPv6 transport over MPLS feature facilitates creation of GRE tunnels over LSPs, through L3VPN inter-AS (autonomous system) options:

- External Border Gateway Protocol (EBGP) redistribution of labeled VPN IPv4/IPv6 routes from an AS to a neighboring AS.
- Multi-hop EBGP redistribution of labeled VPN IPv4/IPv6 routes between source and destination ASs, with EBGP redistribution of labeled IPv4/IPv6 routes from an AS to a neighboring AS.

Multipoint GRE IPv4/IPv6 transport over MPLS is also supported.

The GRE IPv4/IPv6 transport over MPLS feature is supported on the following types of Cisco ASR 9000 line cards:

- Cisco ASR 9000 Enhanced Ethernet line card
- Cisco ASR 9000 High Density 100GE Ethernet line card

Limitations

- GRE IPv4/IPv6 transport over MPLS-TE tunnels is not supported.
- GREoMPLS with IP Fast Reroute (IPFRR).

How to Configure Generic Routing Encapsulation

Configuring a GRE Tunnel

Perform this task to configure a GRE tunnel.

SUMMARY STEPS

1. **configure**
2. **interface tunnel-ip** *number*
3. **vrf** *vrf-name*

4. **ipv4 address** *ipv4-address mask*
5. **tunnel mode gre** {*ipv4* | *ipv6*}
6. **tunnel source** { *ip-address* | **type** *path-id* }
7. **tunnel destination** *ip-address*
8. **tunnel vrf** *transport-vrf-name*
9. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

```
RP/0/RSP0/CPU0:router# configure
```

Enters the Global Configuration mode.

Step 2 **interface tunnel-ip** *number*

Example:

```
RP/0/RSP0/CPU0:router(config)# interface tunnel-ip 4000
```

Enters tunnel interface configuration mode.

- number is the number associated with the tunnel interface.

Step 3 **vrf** *vrf-name*

Example:

```
RP/0/RSP0/CPU0:router(config-if)# vrf vrf1
```

(Optional) Specifies the VRF domain that can route packets into and from the tunnel.

Note This step is not required if the tunnel is available for global routing and therefore, is not specific to a VRF.

Step 4 **ipv4 address** *ipv4-address mask*

Example:

```
RP/0/RSP0/CPU0:router(config-if)# ipv4 address 10.1.1.1 255.255.255.0
```

Specifies the IPv4 address and subnet mask for the interface.

- ipv4-address specifies the IP address of the interface.
- subnet-mask specifies the subnet mask of the interface.

Step 5 **tunnel mode gre** {*ipv4* | *ipv6*}

Example:

```
RP/0/RSP0/CPU0:router(config-if)# tunnel mode gre ipv4
```

Specify whether the transport network is an IPv4 or IPv6 network. The default GRE tunnel mode is IPv4.

Note The tunnel source and destination addresses should match the tunnel mode. A mismatch in configuration causes the tunnel to fail without any error message.

Step 6 **tunnel source** { *ip-address* | **type** *path-id* }

Example:

```
RP/0/RSP0/CPU0:router(config-if)# tunnel source TenGigE0/2/0/1
```

Specifies the source of the tunnel interface.

Note It is recommended that the tunnel source is identified using the interface ID and not the IP address. Using the interface ID enables the router to mark the tunnel as down when the interface is down and the routing protocol tries to find and use an alternate route to the tunnel route.

Step 7 **tunnel destination** *ip-address*

Example:

```
RP/0/RSP0/CPU0:router(config-if)# tunnel destination 145.12.5.2
```

Defines the tunnel destination.

Step 8 **tunnel vrf** *transport-vrf-name*

Example:

```
RP/0/RSP0/CPU0:router(config-if)# tunnel vrf vrf99
```

(Optional) Associates the transport VRF with the tunnel. The transport VRF contains the interfaces over which the tunnel sends as well as receives packets (outer IP packet routing).

Note This step is not required if the tunnel endpoints belong to the global routing table.

Step 9 Use the **commit** or **end** command.

commit - Saves the configuration changes and remains within the configuration session.

end - Prompts user to take one of these actions:

- **Yes** - Saves configuration changes and exits the configuration session.
- **No** - Exits the configuration session without committing the configuration changes.
- **Cancel** - Remains in the configuration mode, without committing the configuration changes.

Configuring the Tunnel Key

Perform this task to configure the tunnel key for the GRE encapsulated packets. You need to perform same configuration steps on the other endpoint router of the tunnel ensuring that the key value is the same at both the local and remote GRE interfaces.

SUMMARY STEPS

1. **configure**
2. **interface tunnel-ip** *number*
3. **ipv4 address** *ipv4-address subnet-mask*
4. **tunnel key** *value*
5. *(Optional)* **tunnel tos** *tos-value*
6. **tunnel source** *type path-id*
7. **tunnel destination** *ip-address*
8. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

```
RP/0/RSP0/CPU0:router# configure
```

Enters the Global Configuration mode.

Step 2 **interface tunnel-ip** *number*

Example:

```
RP/0/RSP0/CPU0:router(config)# interface tunnel-ip 10
```

Enters tunnel interface configuration mode.

- number is the number associated with the tunnel interface.

Step 3 **ipv4 address** *ipv4-address subnet-mask*

Example:

```
RP/0/RSP0/CPU0:router(config-if)# ipv4 address 101.0.9.1 255.255.255.0
```

Specifies the IPv4 address and subnet mask for the interface.

- ipv4-address specifies the IP address of the interface.
- subnet-mask specifies the subnet mask of the interface.

Step 4 **tunnel key** *value*

Example:

```
RP/0/RSP0/CPU0:router(config-if)# tunnel key 10
```

Enables tunnel key.

Step 5 *(Optional)* **tunnel tos** *tos-value*

Example:

```
RP/0/RSP0/CPU0:router(config-if)# tunnel tos 96
```

Specifies the value of the TOS field in the tunnel encapsulating packets.

Step 6 **tunnel source** *type path-id*

Example:

```
RP/0/RSP0/CPU0:router(config-if)# tunnel source Loopback10
```

Specifies the source of the tunnel interface.

Step 7 **tunnel destination** *ip-address*

Example:

```
RP/0/RSP0/CPU0:router(config-if)# tunnel destination 33.0.9.33
```

Defines the tunnel destination.

Step 8 Use the **commit** or **end** command.

commit - Saves the configuration changes and remains within the configuration session.

end - Prompts user to take one of these actions:

- **Yes** - Saves configuration changes and exits the configuration session.
- **No** - Exits the configuration session without committing the configuration changes.
- **Cancel** - Remains in the configuration mode, without committing the configuration changes.

Configuring the Tunnel Key-Ignore

Perform this task to configure the tunnel key-ignore for the GRE encapsulated packets. You need to perform same configuration steps on the other endpoint router of the tunnel.

SUMMARY STEPS

1. **configure**
2. **interface tunnel-ip** *number*
3. **ipv4 address** *ipv4-address subnet-mask*

4. **tunnel key-ignore**
5. *(Optional)* **tunnel tos** *tos-value*
6. **tunnel source** *type path-id*
7. **tunnel destination** *ip-address*
8. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

```
RP/0/RSP0/CPU0:router# configure
```

Enters the Global Configuration mode.

Step 2 **interface tunnel-ip** *number*

Example:

```
RP/0/RSP0/CPU0:router(config)# interface tunnel-ip 10
```

Enters tunnel interface configuration mode.

- number is the number associated with the tunnel interface.

Step 3 **ipv4 address** *ipv4-address subnet-mask*

Example:

```
RP/0/RSP0/CPU0:router(config-if)# ipv4 address 101.0.9.1 255.255.255.0
```

Specifies the IPv4 address and subnet mask for the interface.

- ipv4-address specifies the IP address of the interface.
- subnet-mask specifies the subnet mask of the interface.

Step 4 **tunnel key-ignore**

Example:

```
RP/0/RSP0/CPU0:router(config-if)# tunnel key-ignore
```

Enables tunnel key-ignore.

Step 5 *(Optional)* **tunnel tos** *tos-value*

Example:

```
RP/0/RSP0/CPU0:router(config-if)# tunnel tos 96
```

Specifies the value of the TOS field in the tunnel encapsulating packets.

Step 6 **tunnel source** *type path-id*

Example:

```
RP/0/RSP0/CPU0:router(config-if)# tunnel source Loopback10
```

Specifies the source of the tunnel interface.

Step 7 **tunnel destination** *ip-address*

Example:

```
RP/0/RSP0/CPU0:router(config-if)# tunnel destination 33.0.9.33
```

Defines the tunnel destination.

Step 8 Use the **commit** or **end** command.

commit - Saves the configuration changes and remains within the configuration session.

end - Prompts user to take one of these actions:

- **Yes** - Saves configuration changes and exits the configuration session.
- **No** - Exits the configuration session without committing the configuration changes.
- **Cancel** - Remains in the configuration mode, without committing the configuration changes.

Configuring a VRF Interface

Perform this task to configure a VRF interface.

SUMMARY STEPS

1. **configure**
2. **interface** *type interface-path-id*
3. **vrf** *vrf-name*
4. **ipv4 address** *ipv4-address mask*
5. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

```
RP/0/RSP0/CPU0:router# configure
```

Enters the Global Configuration mode.

Step 2 **interface** *type interface-path-id*

Example:

```
RP/0/RSP0/CPU0:router(config)# interface tunnel-ip 100
```

Enters interface configuration mode.

Step 3 **vrf** *vrf-name*

Example:

```
RP/0/RSP0/CPU0:router(config-if)# vrf vrf_A
```

Configures a VRF instance and enters VRF configuration mode.

Step 4 **ipv4 address** *ipv4-address mask*

Example:

```
RP/0/RSP0/CPU0:router(config-if)# ipv4 address 192.168.1.27 255.255.255.0
```

Configures a primary IPv4 address for the specified interface.

Step 5 Use the **commit** or **end** command.

commit - Saves the configuration changes and remains within the configuration session.

end - Prompts user to take one of these actions:

- **Yes** - Saves configuration changes and exits the configuration session.
- **No** - Exits the configuration session without committing the configuration changes.
- **Cancel** - Remains in the configuration mode, without committing the configuration changes.

Configuring VRF Routing Protocol

Perform this task to configure the VRF routing protocol.

SUMMARY STEPS

1. **configure**
2. **router ospf** *process-name*
3. **vrf** *vrf-name*
4. **router-id** {*router-id* | *type interface-path-id*}
5. **area** *area-id*
6. **interface** *type interface-path-id*
7. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

```
RP/0/RSP0/CPU0:router# configure
```

Enters the Global Configuration mode.

Step 2 **router ospf** *process-name*

Example:

```
RP/0/RSP0/CPU0:router(config)# router ospf 109
```

Enters OSPF configuration mode allowing you to configure the OSPF routing process.

Step 3 **vrf** *vrf-name*

Example:

```
RP/0/RSP0/CPU0:router(config-ospf)# vrf vrf_1
```

Configures a VPN routing and forwarding (VRF) instance and enters VRF configuration mode for OSPF routing.

Step 4 **router-id** {*router-id* | *type interface-path-id*}

Example:

```
RP/0/RSP0/CPU0:router(config-ospf-vrf)# router-id 172.20.10.10
```

Configures the router ID for the OSPF routing process.

Step 5 **area** *area-id*

Example:

```
RP/0/RSP0/CPU0:router(config-ospf-vrf)# area 0
```

Configures the OSPF area as area 0.

Step 6 **interface** *type interface-path-id*

Example:

```
RP/0/RSP0/CPU0:router(config-ospf-vrf-ar)# interface GigabitEthernet 0/3/0/0
```

Associates interface GigabitEthernet 0/3/0/0 with area 0.

Step 7 Use the **commit** or **end** command.

commit - Saves the configuration changes and remains within the configuration session.

end - Prompts user to take one of these actions:

- **Yes** - Saves configuration changes and exits the configuration session.
- **No** - Exits the configuration session without committing the configuration changes.

- **Cancel** - Remains in the configuration mode, without committing the configuration changes.

Configuring IGP for Remote PE Reachability

Perform this task to configure IGP for remote PE reachability.

SUMMARY STEPS

1. **configure**
2. **router ospf** *process-name*
3. **router-id** {*router-id*}
4. **area** *area-id*
5. **interface tunnel-ip** *number*
6. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

```
RP/0/RSP0/CPU0:router# configure
```

Enters the Global Configuration mode.

Step 2 **router ospf** *process-name*

Example:

```
RP/0/RSP0/CPU0:router(config)# router ospf 1
```

Enables OSPF routing for the specified routing process and places the router in router configuration mode.

Step 3 **router-id** {*router-id*}

Example:

```
RP/0/RSP0/CPU0:router(config-ospf)# router-id 1.1.1.1
```

Configures a router ID for the OSPF process.

Note We recommend using a stable IP address as the router ID.

Step 4 **area** *area-id*

Example:

```
RP/0/RSP0/CPU0:router(config-ospf)# area 0
```

Enters area configuration mode and configures an area for the OSPF process.

Step 5 **interface tunnel-ip** *number*

Example:

```
RP/0/RSP0/CPU0:router(config-ospf-ar)# interface tunnel-ip 4
```

Enters tunnel interface configuration mode.

- number is the number associated with the tunnel interface.

Step 6 Use the **commit** or **end** command.

commit - Saves the configuration changes and remains within the configuration session.

end - Prompts user to take one of these actions:

- **Yes** - Saves configuration changes and exits the configuration session.
- **No** - Exits the configuration session without committing the configuration changes.
- **Cancel** - Remains in the configuration mode, without committing the configuration changes.

Configuring LDP on GRE Tunnel

Perform this task to configure LDP on a GRE tunnel.

SUMMARY STEPS

1. **configure**
2. **mpls ldp**
3. **router-id** {*router-id*}
4. **interface tunnel-ip** *number*
5. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

```
RP/0/RSP0/CPU0:router# configure
```

Enters the Global Configuration mode.

Step 2 **mpls ldp**

Example:

```
RP/0/RSP0/CPU0:router(config)# mpls ldp
```

Enables MPLS LDP configuration mode.

Step 3 **router-id** {*router-id*}

Example:

```
RP/0/RSP0/CPU0:router(config-ldp)# router-id 1.1.1.1
```

Configures a router ID for the OSPF process.

Note We recommend using a stable IP address as the router ID.

Step 4 **interface tunnel-ip** *number*

Example:

```
RP/0/RSP0/CPU0:router(config-ldp)# interface tunnel-ip 4
```

Enters tunnel interface configuration mode.

- number is the number associated with the tunnel interface.

Step 5 Use the **commit** or **end** command.

commit - Saves the configuration changes and remains within the configuration session.

end - Prompts user to take one of these actions:

- **Yes** - Saves configuration changes and exits the configuration session.
- **No** - Exits the configuration session without committing the configuration changes.
- **Cancel** - Remains in the configuration mode, without committing the configuration changes.

Configuring MP-iBGP to Exchange VPN-IPv4 Routes

Perform this task to configure MP-iBGP to exchange VPN-IPv4 routes.

SUMMARY STEPS

1. **configure**
2. **router bgp** *as-number*
3. **router-id** *ip-address*
4. **neighbor** *ip-address*
5. **remote-as** *as-number*
6. **update-source** *type interface-path-id*
7. **address-family { vpnv4 | vpnv6 unicast }**
8. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **configure**

Example:

```
RP/0/RSP0/CPU0:router# configure
```

Enters the Global Configuration mode.

Step 2 **router bgp** *as-number*

Example:

```
RP/0/RSP0/CPU0:router(config)# router bgp 1
```

Specifies the autonomous system number and enters the BGP configuration mode, allowing you to configure the BGP routing process.

Step 3 **router-id** *ip-address*

Example:

```
RP/0/RSP0/CPU0:router(config-bgp)# router-id 1.1.1.1
```

Configures the local router with a specified router ID.

Step 4 **neighbor** *ip-address*

Example:

```
RP/0/RSP0/CPU0:router(config-bgp)# neighbor 4.4.4.4
```

Places the router in neighbor configuration mode for BGP routing and configures the neighbor IP address as a BGP peer.

Step 5 **remote-as** *as-number*

Example:

```
RP/0/RSP0/CPU0:router(config-bgp-nbr)#remote-as 1
```

Creates a neighbor and assigns a remote autonomous system number to it.

Step 6 **update-source** *type interface-path-id*

Example:

```
RP/0/RSP0/CPU0:router(config-bgp-nbr)#update-source Loopback0
```

Allows sessions to use the primary IP address from a specific interface as the local address when forming a session with a neighbor.

Step 7 **address-family { vpnv4 | vpnv6 unicast }**

Example:

```
RP/0/RSP0/CPU0:router(config-bgp-nbr)# address-family vpnv4 unicast
```

Enters address family configuration submode for the specified address family.

Step 8 Use the **commit** or **end** command.

commit - Saves the configuration changes and remains within the configuration session.

end - Prompts user to take one of these actions:

- **Yes** - Saves configuration changes and exits the configuration session.
- **No** - Exits the configuration session without committing the configuration changes.
- **Cancel** - Remains in the configuration mode, without committing the configuration changes.

Configuration Examples for Generic Routing Encapsulation

This section provides examples to configure GRE:

Configuring an IPv4 GRE Tunnel: Example

This example shows how to configure an IPv4 GRE tunnel:

```
configure
  interface tunnel-ip1
   ipv4 address 12.0.0.1 255.255.255.0
   tunnel source Loopback0
   tunnel destination 200.200.200.1
end
```

Configuring an IPv6 GRE Tunnel: Example

```
interface tunnel-ip 1
 vrf RED
 ipv4 address 10.1.1.2/24
 ipv6 address 10::2/64
 tunnel mode gre ipv6
 tunnel source GigabitEthernet 0/0/0/0
 tunnel destination 100::1
 tunnel vrf BLUE
 !
```

Verifying GRE tunnel Configuration: Example

```
vrf blue
 description connected to IXIA in blue VRF
 address-family ipv4 unicast
  import route-target
   100:1
  !
  export route-target
100:1
!

vrf red
 description connected to core interface in red VRF
 address-family ipv4 unicast
  import route-target
   200:1
  !
  export route-target
200:1
```

```
!


interface tunnel-ip1
 vrf blue
 ipv4 address 10.10.10.1 255.255.255.0
 tunnel source Loopback0
 keepalive
 tunnel vrf red
 tunnel destination 12.12.12.12


RP/0/RSP0/CPU0:ios#ping vrf red 12.12.12.12
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 12.12.12.12, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/1/2 ms


RP/0/RSP0/CPU0:ios#ping vrf blue 10.10.10.1
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.10.10.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/4/19 ms
```

Configuring Global VRF: Example

This example shows how to configure global VRF:

```
configure
 vrf VRF1
  address-family ipv4 unicast
  import route-target 120:1
  export route-target 120:2
exit
exit
  router bgp120
  vrf VRF1
  rd auto
end
```

Configuring a VRF Interface: Example

This example shows how to configure a VRF interface:

```
configure
 interface tunnel-ip 100
  vrf VRF1
  ipv4 address 1.1.1.1 255.255.255.0
  ipv6 address 100::2/64
end
```

Configuring VRF Routing Protocol: Example

This example shows how to configure VRF routing protocol:

```
configure
  router ospf109
   vrf VRF1
   router-id 172.20.10.10
   area0
   interface GigabitEthernet0/3/0/0
end
```

Configuring IGP for Remote PE Reachability: Example

This example shows how to configure IGP for remote provider edge (PE) reachability:

```
configure
  router ospf109
   router-id 172.20.10.10
   area0
   interface tunnel-ip1
end
```

Configuring LDP on GRE Tunnel: Example

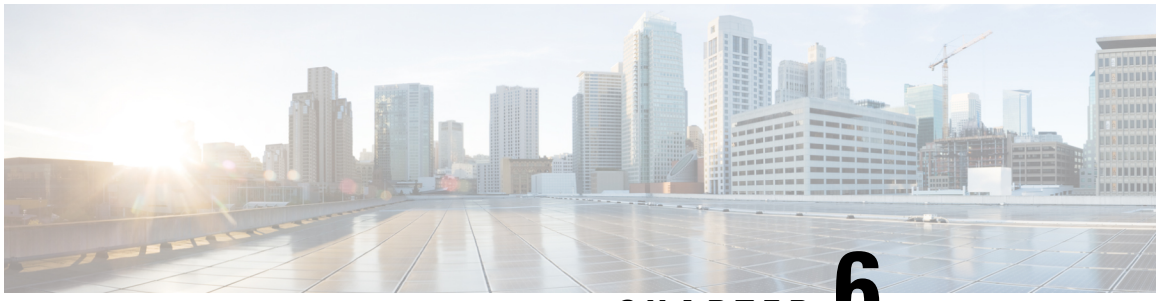
This example shows how to configure LDP on a GRE tunnel:

```
configure
 mpls ldp
 router-id 172.20.10.10
  interface tunnel-ip1
 end
```

Configuring MP-iBGP to Exchange VPN-IPv4 Routes: Example

This example shows how to configure MP-iBGP to exchange VPN-IPv4 routes:

```
configure
 router bgp100
  router-id 172.20.10.10
  neighbor 2.2.2.2 remote-as 100
  update-source Loopback0
  address-family vpnv4 unicast
end
```



CHAPTER 6

Implementing VXLAN

This module provides configuration information for layer 3 VXLAN on Cisco ASR 9000 Series Router. For conceptual information on VXLAN, see *Implementing VXLAN* chapter in the *L2VPN and Ethernet Services Configuration Guide for Cisco ASR 9000 Series Routers*.

Table 3: Feature History for VXLAN

Release	Modification
Release 5.2.0	This feature was introduced on Cisco ASR 9000 Series Router.

Configuring a Layer 3 VXLAN gateway

A layer 3 VXLAN gateway provides routing between VXLAN segment and any other network segment such as VXLAN, VLAN or L3VPN. The following sections show how to configure an ASR 9000 series router as a Layer 3 VXLAN gateway between a VLAN and a VXLAN segment in different networks.

Prerequisites

The following are the prerequisites to configuring a Cisco ASR 9000 series router as a VXLAN Layer 2 gateway:

- Configure a loopback interface. It serves as a source interface for the local VTEP.
- Configure unicast reachability to remote VTEPs.
- Configure Bidirectional Protocol Independent Multicast (Bidir PIM) or PIM Sparse Mode. For more information, see the *Multicast Configuration Guide for Cisco ASR 9000 Series Routers*.

Restrictions

Consider the following restrictions while configuring VXLAN:

- You configure VXLAN only on Overlay Transport Virtualization (OTV) and VXLAN UDP ports.

- The source interface can only be a loopback interface.
- You cannot share a VNI or a multicast group or a source interface across multiple NVE interfaces.
- The VNI range and the multicast range both can only be specified contiguously. A non-contiguous range with comma separated values is not supported.
- The VNI to multicast group mapping can be only either 1:1 or N:1. For example,
  - The "member vni 5000 mcast-group 239.1.1.1" command configures a valid 1:1 mapping.
  - The "member vni 5000-5005 mcast-group 239.1.1.1" command configures a valid N:1 mapping.
- When a VNI is configured as a part of a VNI range, it can be modified or deleted only as part of the same range. For example, if the "member vni 5000-5002 mcast-group 239.1.1.1" command is configured, you cannot disassociate just the VNI 5001 from the NVE interface with a "no member vni 5001" command.
- Static MAC configuration is not supported.
- You can configure a maximum of 128k Layer 2 and Layer 3 sub-interfaces per system. The configuration can be a combination of both Layer 2 sub-interfaces and Layer 3 sub-interfaces; or either fully Layer 2 sub-interfaces or Layer 3 sub-interfaces.

  Though the system allows you to configure more than 128k sub-interfaces per system, you cannot use this configuration for services. Though the system displays a warning message on reaching the threshold of 128k sub-interfaces, the configuration is still applied. However, you cannot use this configuration for services.

Creating and configuring the Network Virtualization Endpoint (NVE) interface

Perform this task to create an NVE interface and configure it as a VXLAN Tunnel EndPoint (VTEP) for VxLAN.

SUMMARY STEPS

1. **interface nve** *nve-identifier*
2. **source-interface loopback** *loopback-interface-identifier*
3. **member vni** *vni_number* [*-end_vni_range*] **mcast-group** *ip_address* [*end_ip_address_range*]
4. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **interface nve** *nve-identifier*

Example:

```
RP/0/RSP0/CPU0:router(config)# interface nve 1
```

Creates the NVE interface and enters the NVE interface configuration sub-mode.

Step 2 **source-interface loopback** *loopback-interface-identifier*

Example:

```
RP/0/RSP0/CPU0:router(config-if)# source-interface loopback 1
```

Sets a loopback interface as the source interface for the VTEP.

Step 3 **member vni** *vni_number* [*-end_vni_range*] **mcast-group** *ip_address* [*end_ip_address_range*]

Example:

```
RP/0/RSP0/CPU0:router(config-if)# member vni 1-10 mcast-group 224.2.2.2
```

Associates a single VxLAN or a contiguous range of VxLANs with the NVE interface using their VxLAN Network Identifiers (VNIs) and specifies a multicast address or a contiguous multicast address range associated with these VNIs.

Note The mapping between the VNIs and the multicast groups is either one-to-one or many-to-one.

Step 4 Use the **commit** or **end** command.

commit - Saves the configuration changes and remains within the configuration session.

end - Prompts user to take one of these actions:

- **Yes** - Saves configuration changes and exits the configuration session.
- **No** - Exits the configuration session without committing the configuration changes.
- **Cancel** - Remains in the configuration mode, without committing the configuration changes.

Configuring the L3 bridge virtual interface

Perform this task to configure the IPv4 address for a bridge virtual interface for L3 routing.

SUMMARY STEPS

1. **interface BVI** *BVI-identifier*
2. **ipv4 address** *ip-address* {*/prefix* | *subnet mask*}
3. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **interface BVI** *BVI-identifier*

Example:

```
RP/0/RSP0/CPU0:router(config)# interface BVI 1
```

Enters the bridge virtual interface configuration mode.

Step 2 **ipv4 address** *ip-address* {*/prefix* | *subnet mask*}

Example:

```
RP/0/RSP0/CPU0:router(config-if)# ipv4 address 1.1.1.1 255.0.0.0
```

Sets the IPv4 address for the bridge virtual interface.

Step 3 Use the **commit** or **end** command.

commit - Saves the configuration changes and remains within the configuration session.

end - Prompts user to take one of these actions:

- **Yes** - Saves configuration changes and exits the configuration session.
- **No** - Exits the configuration session without committing the configuration changes.
- **Cancel** - Remains in the configuration mode, without committing the configuration changes.

Configuring a bridge domain

Perform this task to configure a bridge domain.

SUMMARY STEPS

1. **l2vpn**
2. **bridge group** *bridge-group-name*
3. **bridge-domain** *bridge-domain-name*
4. **member vni** *vxlan-id*
5. **routed interface BVI** *BVI-id*
6. Use the **commit** or **end** command.

DETAILED STEPS

Step 1 **l2vpn**

Example:

```
RP/0/RSP0/CPU0:router(config)# l2vpn
```

Enters the l2vpn configuration mode.

Step 2 **bridge group** *bridge-group-name*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn)# bridge group bgroup1
```

Enters the bridge group configuration mode.

Step 3 **bridge-domain** *bridge-domain-name*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg)# bridge-domain bdomain1
```

Enters the bridge domain configuration mode.

Step 4 **member vni** *vxlan-id*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)# member vni 10
```

Associates a member VNI with the bridge domain.

Step 5 **routed interface BVI** *BVI-id*

Example:

```
RP/0/RSP0/CPU0:router(config-l2vpn-bg-bd)# routed interface BVI 1
```

Sets the bridge virtual interface for the bridge domain.

Step 6 Use the **commit** or **end** command.

commit - Saves the configuration changes and remains within the configuration session.

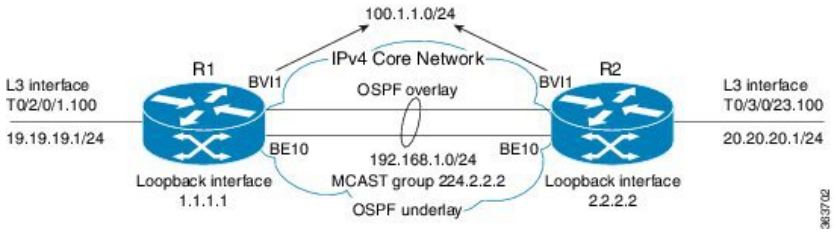
end - Prompts user to take one of these actions:

- **Yes** - Saves configuration changes and exits the configuration session.
- **No** - Exits the configuration session without committing the configuration changes.
- **Cancel** - Remains in the configuration mode, without committing the configuration changes.

Configuration Example for Implementing Layer 3 VXLAN Gateway

The following example shows layer 3 VXLAN gateway configuration on two Provider Edge (PE) routers, R1 and R2, from a sample network topology that has the core network simplified as a bundle link connection between the PE routers.

Figure 15: Network with Layer 3 VXLAN Gateways



Configuration at R1:

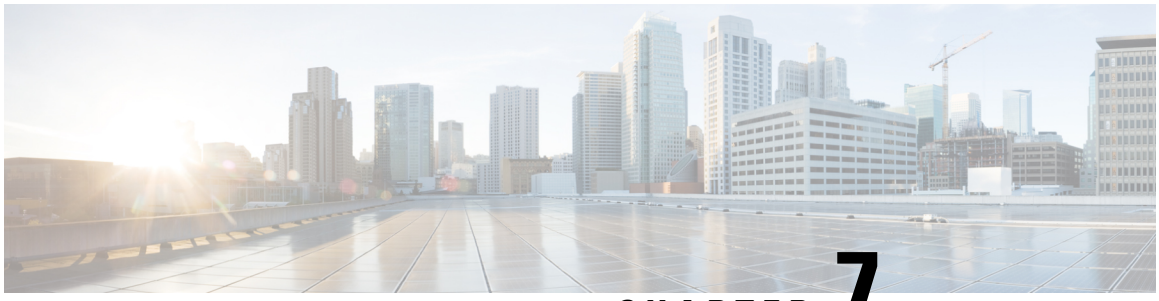
```
interface Bundle-Ether10
  ipv4 address 192.168.1.1/24
!
interface Loopback0
  ipv4 address 1.1.1.1/32
!
interface T0/2/0/1
 no shut
!
interface T0/2/0/1.100
 encapsulation dot1q 100
 ipv4 address 19.19.19.1/24
!
interface BVI1
 ipv4 address 100.1.1.1 255.255.255.0
 ipv6 address 100::1/64
!
router ospf underlay
 router-id 1.1.1.1
 area 0
  interface Bundle-Ether10
  interface Loopback0
!
Interface nve 1
  member vni 1 mcast-group 224.2.2.2 0.0.0.0
  overlay-encapsulation vxlan
```

```
  source-interface Loopback0
!
router ospf overlay
  area 0
    interface bvi1
 interface T0/2/0/1.100
!
l2vpn
 bridge group vxlan
  bridge-domain vxlan
    routed interface BVI1
    member vni 1
  !
multicast-routing
 address-family ipv4
  interface loopback0
    enable
  interface Bundle-Ether10
     enable
!
router pim
 address-family ipv4
  rp-address 1.1.1.1 bidir
```

Configuration at R2:

```
interface Bundle-Ether10
  ipv4 address 192.168.1.2/24
!
interface Loopback0
  ipv4 address 2.2.2.2/32
!
interface T0/3/0/23
 no shut
!
interface T0/3/0/23.100
 encapsulation dot1q 100
 ipv4 address 20.20.20.1/24
!
interface BVI1
 ipv4 address 100.1.1.2 255.255.255.0
 ipv6 address 100::2/64
router ospf underlay
 router-id 2.2.2.2
 area 0
  interface Bundle-Ether10
  interface Loopback0
!
Interface nve 1
  member vni 1 mcast-group 224.2.2.2 0.0.0.0
  overlay-encapsulation vxlan
  source-interface Loopback0
!
router ospf overlay
  area 0
    interface bvi1
    interface T0/3/0/23.100
!
l2vpn
 bridge group vxlan
  bridge-domain vxlan
    routed interface BVI1
    member vni 1
  !
multicast-routing
```

```
 address-family ipv4
  interface loopback0
    enable
  interface Bundle-Ether10
     enable
!
router pim
 address-family ipv4
  rp-address 1.1.1.1 bidir
```



CHAPTER 7

Implementing IP in IP Tunnel

This chapter module provides conceptual and configuration information for IP in IP tunnels on Cisco ASR 9000 Series Router.

Note For a complete description of the IP in IP tunnel commands listed in this chapter, see the *VPN and Ethernet Services Command Reference for Cisco ASR 9000 Series Routers*.

Table 4: Feature History for IP in IP tunnel

Release	Modification
Release 5.3.1	This feature was introduced on Cisco ASR 9000 Series Router.

IP in IP Tunneling

IP in IP tunneling refers to the encapsulation of an IP packet as a payload in another IP packet. ASR9K routers support IP in IP tunnels with all possible combinations of IPv4 and IPv6; that is, IPv4 over IPv4, IPv6 over IPv4, IPv4 over IPv6, and IPv6 over IPv6. For example, an IPv4 over IPv6 refers to an IPv4 packet as a payload encapsulated within an IPv6 packet and routed across an IPv6 network to reach the destination IPv4 network, where it is decapsulated.

IP in IP tunneling does not require any additional header such as a GRE header used in the GRE tunnels. So, IP in IP tunneling is preferred over GRE tunnels if both the networks are IP networks.

Restrictions

The following are not supported in IP in IP tunnels:

- MPLS
- Multicast packets

- Keep-Alive packets
- Path MTU (Maximum Transmission Unit) discovery
- DF (Do not Fragment) bit configuration in IPv6 tunnel mode.

Note If DF bit is configured for the tunnel interface, you cannot enable IPv6 tunnel mode.

Configuring IP in IP Tunnel

Perform the following steps to configure an IP in IP tunnel.

SUMMARY STEPS

1. **configure**
2. **interface tunnel-ip** *tunnel-id*
3. {**ipv4** | **ipv6**} **address** *ip-address*
4. **tunnel mode** {*ipv4* | *ipv6*}
5. **tunnel source** {*interface-id* | *ipv4/v6-address*}
6. **tunnel destination** *ipv4/v6-address*
7. *(Optional)* **tunnel df-bit** {*copy* | *disable*}
8. *(Optional)* **tunnel tos** *tos-value*
9. Use the **commit** or **end** command.

DETAILED STEPS

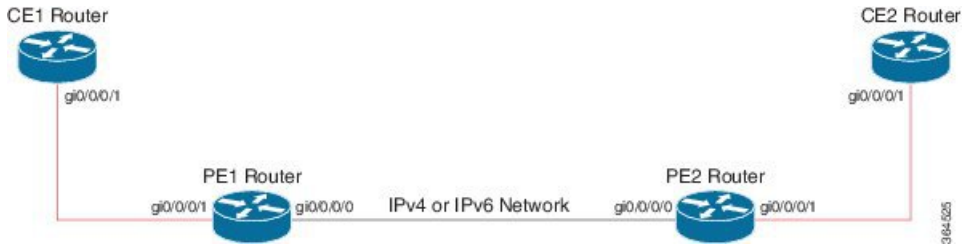
	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RSP0/CPU0:router# configure`	Enters the Global Configuration mode.
Step 2	**interface tunnel-ip** *tunnel-id* **Example:** `RP/0/RSP0/CPU0:router(config)# interface tunnel-ip 1`	Creates a tunnel interface and enters the tunnel configuration sub-mode. The *tunnel-id* is the numeric identifier for the tunnel interface.
Step 3	{**ipv4** | **ipv6**} **address** *ip-address* **Example:** `RP/0/RSP0/CPU0:router(config-if)# ipv6 address 10::1/64`	Sets the IPv4 or IPv6 address, as required, for the tunnel interface.

	Command or Action	Purpose
Step 4	**tunnel mode** {*ipv4* | *ipv6*} **Example:** `RP/0/RSP0/CPU0:router(config-if)# tunnel mode ipv6`	Sets the tunnel mode as IPv4 or IPv6. This states the tunnel is in an IPv4 or IPv6 transport network.
Step 5	**tunnel source** {*interface-id* | *ipv4/v6-address*} **Example:** `RP/0/RSP0/CPU0:router(config-if)# tunnel source GigabitEthernet0/0/0/0`	Specifies an IP address or an interface that serves as the tunnel source. The encapsulated packet uses this IP address as the source address. If the tunnel mode is set to IPv4 or IPv6, an IPv4 or IPv6 address is selected as a tunnel source address from the specified source interface respectively.
Step 6	**tunnel destination** *ipv4/v6-address* **Example:** `RP/0/RSP0/CPU0:router(config-if)# tunnel destination 100::2`	Specifies the destination IP address for the tunnel. The encapsulated packet uses this IP address as the destination address.
Step 7	*(Optional)* **tunnel df-bit** {*copy* | *disable*} **Example:** `RP/0/RSP0/CPU0:router(config-if)# tunnel df-bit disable`	**Note** This is valid only for a tunnel that uses an IPv4 transport network. Configures the DF bit value for the outer IP packet. For details on this **tunnel df-bit** command, see the *VPN and Ethernet Services Command Reference for Cisco ASR 9000 Series Routers*.
Step 8	*(Optional)* **tunnel tos** *tos-value* **Example:** `RP/0/RSP0/CPU0:router(config-if)# tunnel tos 1`	Sets the TOS value for the outer IP packet in the tunnel. For details on this **tunnel tos** command, see the *VPN and Ethernet Services Command Reference for Cisco ASR 9000 Series Routers*.
Step 9	Use the **commit** or **end** command.	**commit** —Saves the configuration changes and remains within the configuration session. **end** —Prompts user to take one of these actions: • **Yes** — Saves configuration changes and exits the configuration session. • **No** —Exits the configuration session without committing the configuration changes. • **Cancel** —Remains in the configuration session, without committing the configuration changes.

IP in IP Tunneling: Examples

The following examples provide configurations for an IPv4 or IPv6 tunnel, with the transport VRF as the default VRF for the following simplified network topology.

Figure 16: IP in IP tunnel network topology



Configuration example for an IPv4 tunnel

PE1 Router Configuration	PE2 Router Configuration

`interface GigabitEthernet0/0/0/0`
`!! Link between PE1-PE2`
`ipv4 address 100.1.1.1/64`
`!`
`interface GigabitEthernet0/0/0/1`
`!! Link between CE1-PE1`
`ipv4 address 20.1.1.1/24`
`ipv6 address 20::1/64`
`!`
`interface tunnel-ip 1`
`ipv4 address 10.1.1.1/24`
`ipv6 address 10::1/64`
`tunnel mode ipv4`
`tunnel source GigabitEthernet0/0/0/0`
`tunnel destination 100.1.1.2`
`!`
 	
`router static`
`address-family ipv4 unicast`
`30.1.1.0/24 tunnel-ip1`
`address-family ipv6 unicast`
`30::0/64 tunnel-ip1`
`!`
`!`
`!`
	`interface GigabitEthernet0/0/0/0`
	`!! Link between PE1-PE2`
	`ipv4 address 100.1.1.2/64`
	`!`
	`interface GigabitEthernet0/0/0/1`
	`!! Link between PE2-CE2`
	`ipv4 address 30.1.1.1/24`
	`ipv6 address 30::1/64`
	`!`
	`interface tunnel-ip 1`
	`ipv4 address 10.1.1.2/24`
	`ipv6 address 10::2/64`
	`tunnel mode ipv4`
	`tunnel source GigabitEthernet0/0/0/0`
	`tunnel destination 100.1.1.1`
	`!`
	
	`router static`
	`address-family ipv4 unicast`
	`20.1.1.0/24 tunnel-ip1`
	`address-family ipv6 unicast`
	`20::0/64 tunnel-ip1`
	`!`
	`!`
	`!`

CE1 Router Configuration	CE2 Router Configuration

`interface GigabitEthernet0/0/0/1`
`!! Link between CE1-PE1`
`ipv4 address 20.1.1.2 255.255.255.0`
`ipv6 address 20::2/64`
`!`
`router static`
`address-family ipv4 unicast`
`30.1.1.0/24 20.1.1.1`
`address-family ipv6 unicast`
`30::0/64 20::1`
`!`
`!`
	`interface GigabitEthernet0/0/0/1`
	`!! Link between CE2-PE2`
	`ipv4 address 30.1.1.2 255.255.255.0`
	`ipv6 address 30::2/64`
	`!`
	`router static`
	`address-family ipv4 unicast`
	`20.1.1.0/24 30.1.1.1`
	`address-family ipv6 unicast`
	`20::0/64 30::1`
	`!`
	`!`

Configuration example for an IPv6 tunnel

PE1 Router Configuration	PE2 Router Configuration

```
interface GigabitEthernet0/0/0/0
 !! Link between PE1-PE2
 ipv6 address 100::1/64
!
interface GigabitEthernet0/0/0/1
 !! Link between CE1-PE1
 vrf RED
 ipv4 address 20.1.1.1/24
 ipv6 address 20::1/64
!
interface tunnel-ip 1
 vrf RED
 ipv4 address 10.1.1.1/24
 ipv6 address 10::1/64
 tunnel mode ipv6
 tunnel source GigabitEthernet0/0/0/0
 tunnel destination 100::2
 !
vrf RED
 address-family ipv6 unicast
  import route-target
   2:1
  !
  export route-target
   2:1
  !
 address-family ipv4 unicast
  import route-target
   2:1
  !
  export route-target
   2:1
  !
router static
vrf RED
  address-family ipv4 unicast
   30.1.1.0/24 tunnel-ip1
  address-family ipv6 unicast
   30::0/64 tunnel-ip1
  !
 !
!
```

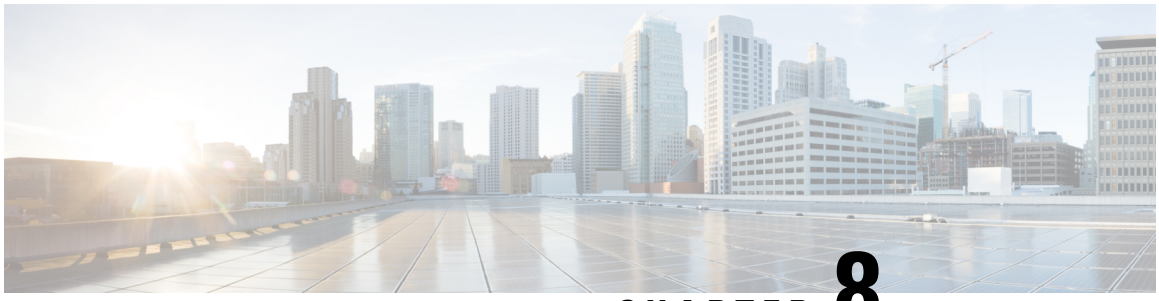
```
interface GigabitEthernet0/0/0/0
!! Link between PE1-PE2
 ipv6 address 100::2/64
!
interface GigabitEthernet0/0/0/1
 !! Link between PE2-CE2
 vrf RED
 ipv4 address 30.1.1.1/24
 ipv6 address 30::1/64
!
interface tunnel-ip 1
 vrf RED
 ipv4 address 10.1.1.2/24
 ipv6 address 10::2/64
 tunnel mode ipv6
 tunnel source GigabitEthernet0/0/0/0
 tunnel destination 100::1
 !
vrf RED
 address-family ipv6 unicast
  import route-target
   2:1
  !
  export route-target
   2:1
  !
 address-family ipv4 unicast
  import route-target
   2:1
  !
  export route-target
   2:1
  !
router static
vrf RED
  address-family ipv4 unicast
   20.1.1.0/24 tunnel-ip1
  address-family ipv6 unicast
   20::0/64 tunnel-ip1
  !
 !
!
```

CE1 Router Configuration

```
interface GigabitEthernet0/0/0/1
!! Link between CE1-PE1
 ipv4 address 20.1.1.2 255.255.255.0
 ipv6 address 20::2/64
!
router static
 address-family ipv4 unicast
  30.1.1.0/24 20.1.1.1
 address-family ipv6 unicast
  30::0/64 20::1
 !
!
```

CE2 Router Configuration

```
interface GigabitEthernet0/0/0/1
!! Link between CE2-PE2
 ipv4 address 30.1.1.2 255.255.255.0
 ipv6 address 30::2/64
!
router static
 address-family ipv4 unicast
  20.1.1.0/24 30.1.1.1
address-family ipv6 unicast
  20::0/64 30::1
 !
!
```



CHAPTER 8

Implementing DCI Layer 3 Gateway between MPLS-VPN and EVPN Data Center

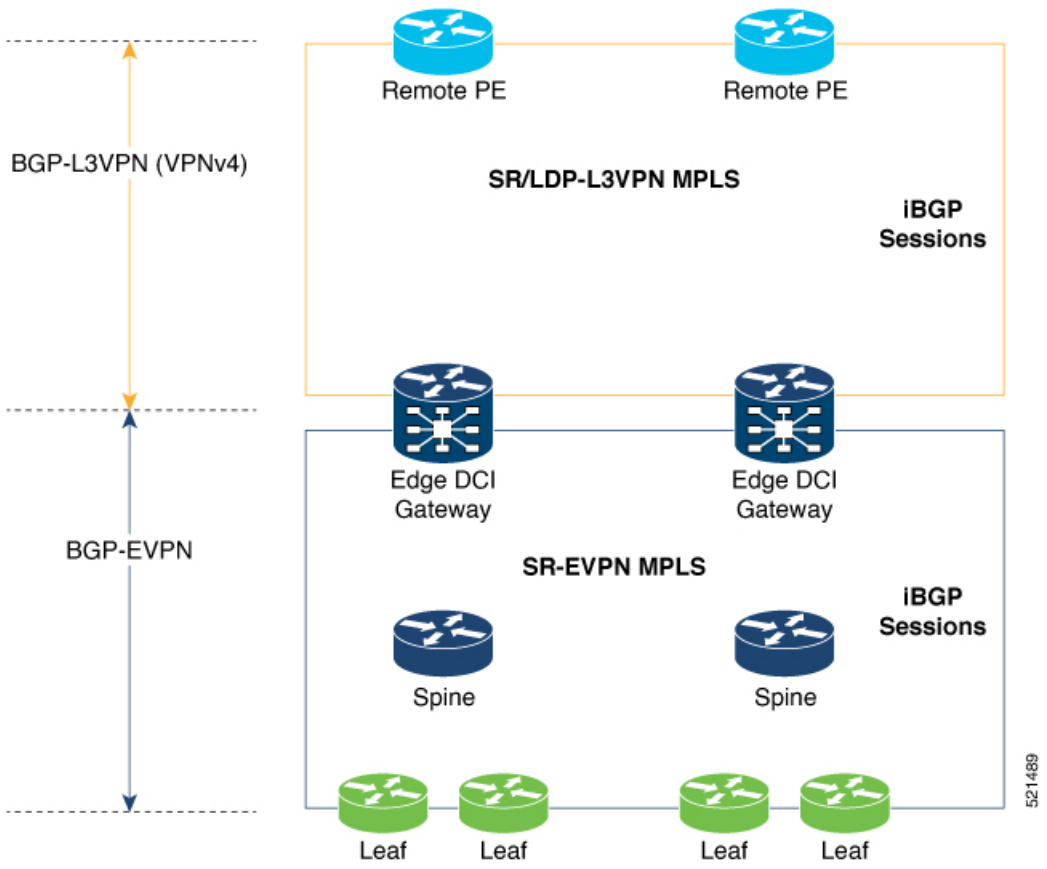
This chapter module provides conceptual and configuration information for Data Center Interconnect (DCI) Layer 3 Gateway between MPLS-VPN and EVPN Data Center.

Data Center Interconnect between MPLS-VPN and EVPN-MPLS

This part provides conceptual and configuration information for Data Center Interconnect (DCI) Layer 3 Gateway with EVPN-MPLS on Cisco ASR 9000 Series Router.

DCI Layer 3 Gateway with EVPN-MPLS

You can use SR-EVPN for Data Center on routers for a spine-leaf architecture with edge devices such as border leaf. DCI L3 stitching allows Data Centers that run SR-EVPN to communicate with legacy and existing MPLS VPN (VPNv4) sites.



In this topology,

Leaf (ToR) – Router acts as both access switch and distributed PE. Leaf establishes BGP EVPN neighborship with Spine route-reflector (RR). This router sends and receives prefixes from the DCI Gateway. Leaf ToR provides the following types of services:

- Regular L3 VRF configuration using subinterfaces to attach some CE devices. Traditional PE-CE scenario without EVPN configuration.
- L3 EVPN VRF using L2VPN configuration to attach multiple Data Centers services.

Leaf sends and receives prefixes from or to the DCI gateway:

- Leaf sends prefixes to DCI: Leaf re-originates local learned VRF subnet route as EVPN Route Type 5 with the EVPN RT (stitching-rt or regular RT), then sends to Spine RR. Spine RR sends prefixes to DCI gateway.
- Leaf receives prefixes from DCI: Leaf receives EVPN Route Type 5 from Spine RR that is re-originated at DCI gateway due to stitching between VPNv4 and EVPN. Leaf imports remote VPNv4 prefixes to local VRF matching VPNv4 RT (stitching-rt or regular RT).

Spine RR: Spine RR establishes BGP EVPN neighborship with Leaf (ToR) and Edge DCI Gateway serving as Route-Reflector for EVPN prefixes between the devices in the Data Center. Leaf and DCI Gateway must be configured as clients of Spine RR.

Edge (DCI gateway): Edge (DCI gateway) acts as an edge router that allows communication between services connected at Leaf and CEs in legacy MPLS network architecture. The edge DCI gateway establishes BGP EVPN neighborship with Spine RR and remote PEs, or RR depending on legacy MPLS network architecture.

The edge DCI gateway sends and receives prefixes from or to the Data Center:

- DCI gateway receives prefixes from legacy MPLS VPNv4 network and sends prefixes to Leaf: DCI gateway receives L3VPN (VPNv4) routes from remote MPLS VPN (VPNv4) PE or RR depending on legacy MPLS network architecture matching the VPNv4 RT (stitching-rt or regular RT). Then re-originate these prefixes as EVPN Route Type 5 with the EVPN RT (stitching-rt or regular RT) advertising to Spine RR due to BGP EVPN neighbor with the Spine.
- DCI gateway receives prefixes from Leaf and sends prefixes to legacy MPLS VPNv4 network: DCI gateway receives EVPN Route Type 5 originated from Leaf (ToR) by Spine RR due to BGP EVPN neighbor with the Spine. Leaf and DCI gateway does not have a direct BGP neighborship. Then import the routes to local VRF matching the EVPN RT (stitching-rt or regular RT) and re-originate this prefix as VPNv4 router with the VPNv4 RT (stitching-rt or regular RT) and advertise to remote MPLS VPN (VPNv4) PE or RR depending on legacy MPLS network architecture.

Remote PE: Remote PE receives traditional MPLS L3VPN prefixes (VPNv4) by DCI Gateway or RR depending on legacy MPLS network architecture. You must have a unique Route-Distinguisher (RD) between remote PEs and DCI gateway to allow stitching re-originate prefixes from VPNv4 to EVPN at DCI Gateway.

Stitching RTs and Regular RTs can be assigned to any side, EVPN or VPNv4, irrespective of the address-family. Consider the following supported scenarios:

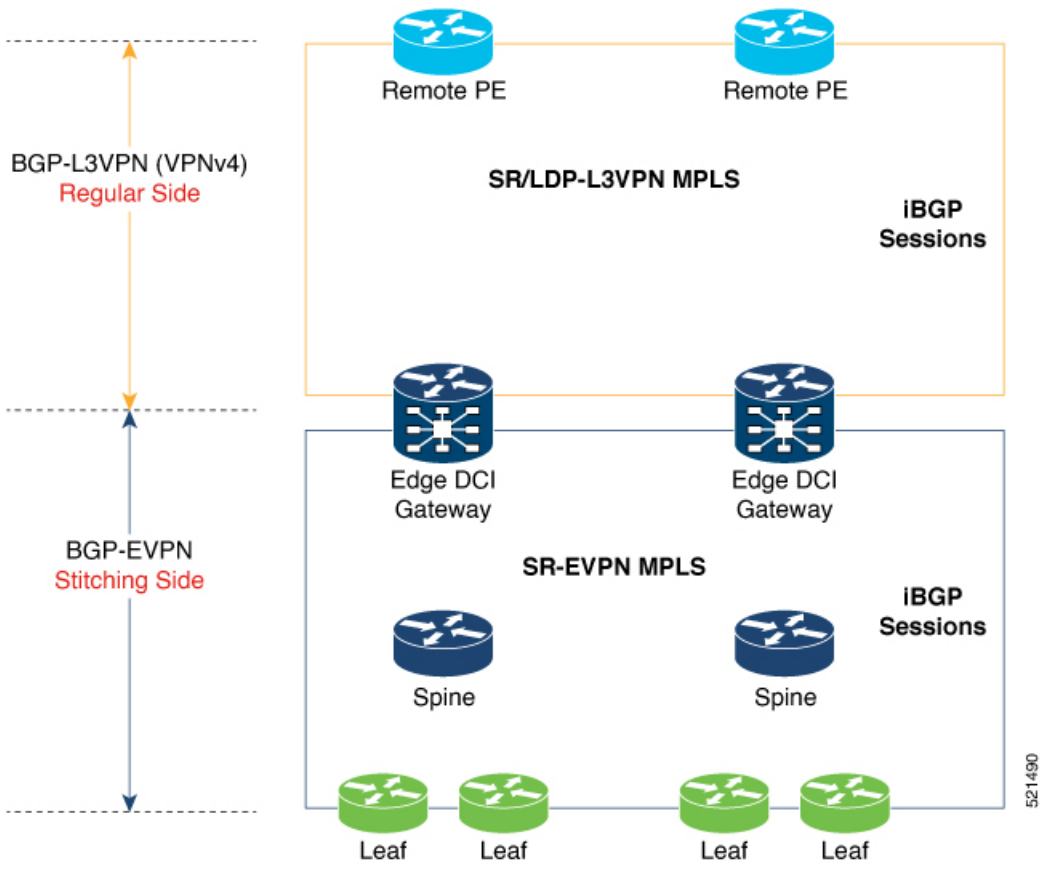
VPNv4-Regular RT and EVPN-Stitching RT

For each VRF on the DCI gateway, there are two sets of manually configured import and export route-targets for VPNv4 as a regular side and EVPN as a stitching side. Consider the following sets:

- Data Center Route-Targets for EVPN associated with EVPN BGP neighbor (Stitching RT).
- MPLS L3VPN Route-Targets for VPNv4 or VPNv6 associated with L3VPN BGP neighbor (Regular RT).

This separation of RTs enables the two sets of RTs to be independently configured. The RTs associated with the EVPN BGP neighbor require **stitching-rt** keyword under VRF configuration. The route-types associated with the L3VPN BGP neighbor do not require the keyword.

The following topology shows regular/normal and stitching side.



Route Targets

The RTs associated with the EVPN BGP neighbor are labelled as stitching RTs. The RTs associated with the L3VPN BGP neighbor are normal RTs.

Route Re-Origination

Consider control plane information propagation by the edge DCI gateway from the L3VPN (regular/normal side) to the Data Center (stitching side). Edge DCI gateway advertises to its BGP EVPN neighbor the routes that are re-originated after importing them from the L3VPN BGP neighbor. For this case of VPNv4 or VPNv6 routes being propagated to the BGP EVPN neighbors (Data Center neighbors), re-originating the routes refers to replacing the normal route-targets with the local route-target values (stitching-rt) associated with the BGP EVPN neighbors.

Route Address-Family and Encoded Address-Family

When an address-family is configured for a BGP neighbor, it means that the specified address-family routes encoded with the NLRI for that address-family are advertised to the neighbor. This does not hold for Data Center BGP neighbors because they use only EVPN address-family. Here, BGP neighbors advertise VPNv4 or VPNv6 unicast routes using the EVPN NLRI encoding. Thus, the encoded address-family and route address family can be possibly different. You can advertise the VPNv4 or VPNv6 address-family using the **advertise vpnv4 unicast** or **advertise vpnv6 unicast** command. For example, an EVPN address-family BGP neighbor configured with the **advertise vpnv4 unicast** command sends VPNv4 unicast routes in an EVPN encoded NLRI.

Local VPNv4 or VPNv6 Route Advertisement

On the edge DCI gateway, the locally sourced VPNv4 or VPNv6 routes (any CE directly connected not using L2VPN with BD/EVI/BVI, using only regular L3 VRF) can be advertised to the BGP EVPN neighbors with the normal route targets (RTs) configured for the VRF or the stitching RTs associated with the BGP EVPN neighbors. By default, these routes are advertised with the normal route targets. You can configure this local VPNv4 or VPNv6 route advertisements to be advertised with stitching RTs to the BGP EVPN neighbors by using the **advertise vpnv4 unicast local stitching-rt** or **advertise vpnv6 unicast local stitching-rt** command as required.

VPNv4 neighbors do not require any additional configuration. By default, these routes are advertised with the normal route-targets to BGP L3VPN neighbors.

Route Distinguishers

The Router Distinguisher (RD) associated per VRF must be unique per PE in the network. There are few available options to keep unique RD per device:

- Manual configuration: You must manually assign a unique value per device in the network. For example, in this scenario:
  - Leaf (ToR) = RD 1
  - Edge DCI Gateway = RD 2
  - Remote PE = RD 3
- Use **rd auto** command under VRF. To assign a unique route distinguisher for each router, you must ensure that each router has a unique BGP router-id. If so, the **rd auto** command assigns a Type 1 route distinguisher to the VRF using the following format: *ip-address:number*. The IP address is specified by the BGP router-id statement and the number (which is derived as an unused index in the 0 to 65535 range) is unique across the VRFs.

Note In a DCI deployment, for route re-originate with stitching-rt for a particular VRF, using the same Route Distinguisher (RD) between edge DCI gateway and MPLS-VPN PE or same RD between edge DCI gateway and Leaf (ToR) is not supported.

Configure VPNv4-Regular RT and EVPN-Stitching RT

This section describes tasks to configure VPNv4-Regular RT and EVPN-Stitching RT. Perform the following tasks to complete the configuration:

- Configure Leaf (ToR)
- Configure Spine-RR (Route Reflector)
- Configure Edge DCI Gateway
- Configure EVPN BGP neighbor and route advertisements
- Configure L3VPN BGP neighbor relationship and route advertisements

Configure Leaf (ToR)

Configure VRF in Leaf (ToR) at BGP-EVPN (Stitching Side) with Stitching-RT.

```
vrf data-center1
 address-family ipv4 unicast
  import route-target
   1:2 stitching                              // BGP - EVPN (Stitching Side)
!
  export route-target
   1:2 stitching                             // BGP - EVPN (Stitching Side)
  !
router bgp 100
  neighbor 10.10.1.1                        // Spine Loopback IP Address
    address-family l2vpn evpn
       advertise vpnv4 unicast
       advertise vpnv6 unicast
  !
```

Note Advertise vpnv4/vpnv6 unicast enables local learned regular L3 VRF prefixes to be advertised as EVPN prefixes to BGP – EVPN neighbor. This means any local prefixes such as PE-CE without L2VPN with BD/EVI/BVI configuration. If all the services are pure EVPN with L2VPN with BD/EVI/BVI configuration these commands are not required.

Configure Spine-RR

Configure Spine RR with Leaf (ToR) and edge DCI gateway as RR client for AFI L2VPN EVPN. VRF configuration is not required.

```
// VRF Config is not required //

router bgp 100
  neighbor 10.10.2.1                 // Leaf (ToR) Loopback IP Address
    address-family l2vpn evpn
       route-reflector-client
   !
  neighbor 10.10.3.1                // Edge DCI Gateway  Loopback IP Address
    address-family l2vpn evpn
       route-reflector-client
!
```

Configure Edge DCI Gateway

You can configure DCI with the same VRF as Leaf (ToR). Use the same RT as remote PE for L3VPN network or the same VRF if that is possible.

Configure VRF and Route Targets Import and Export rules

Perform the following steps to configure VRF and define route targets to be used for import and export of forwarding information.

```
vrf data-center1
 address-family ipv4 unicast
  import route-target
   1:1                              // BGP – L3VPN (Regular/normal Side)
```

```
   1:2 stitching                    // BGP - EVPN (Stitching Side)
!
  export route-target
   1:1                             // BGP - L3VPN (Regular/normal Side)
   1:2 stitching                   // BGP - EVPN (Stitching Side)
!
```

Configure EVPN BGP Neighbor and Route Advertisements

Perform this task on the edge DCI gateway to configure BGP neighbor relationship and route advertisements with the EVPN BGP neighbor.

```
router bgp 100
 addreess-family l2vpn evpn
!
 neighbor 10.10.1.1               // Spine Loopback IP Address
  address-family l2vpn evpn
   import stitching-rt re-originate  //Imp EVPN 1:2, reoriginate VPNv4 RT 1:1
   advertise vpnv4 unicast re-originated stitching-rt //Send routes EVPN 1:2
   advertise vpnv6 unicast re-originated stitching-rt //Send routes EVPN 1:2
!
```

Configure L3VPN BGP Neighbor Relationship and Route Advertisements

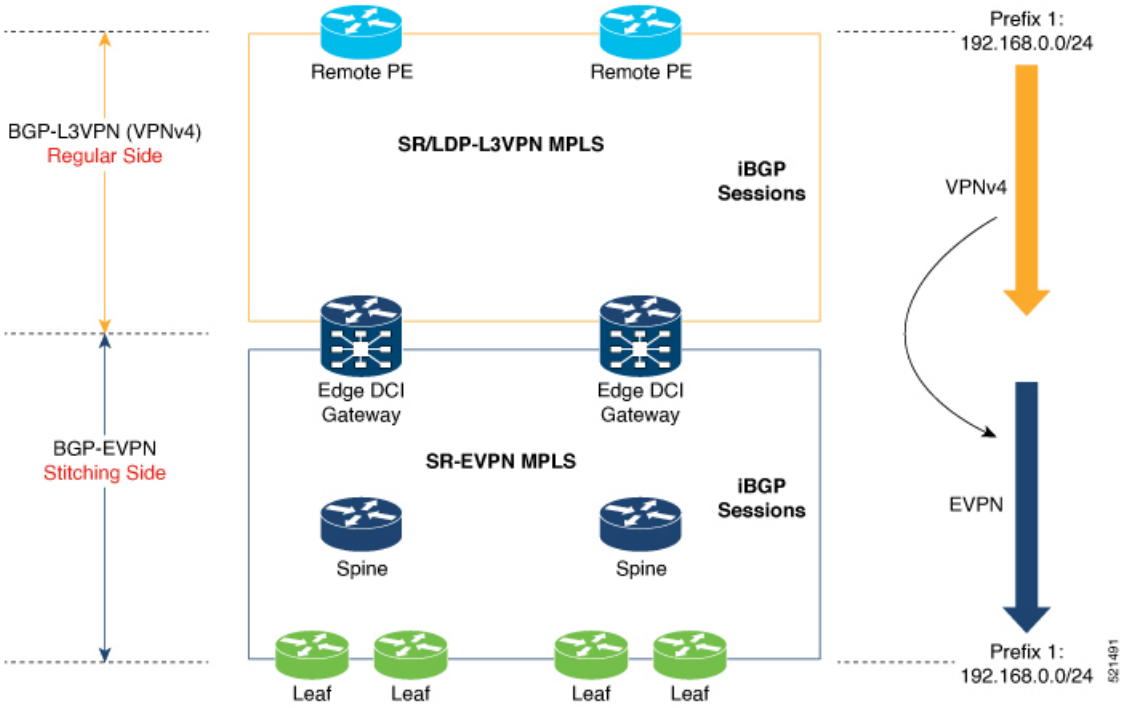
Perform the following steps to configure BGP neighbor relationship and route advertisements with the L3VPN BGP neighbor.

```
router bgp 100
 address-family vpnv4 unicast
!
 neighbor 10.10.1.1               // Spine Loopback IP Address
  address-family vpnv4 unicast      // Same config for VPNv6
   import re-originate stitching-rt // Imp VPNv4 1:1, re-originate EVPN 1:2
   advertise vpnv4 unicast re-originated   // Send routes VPNv4 RT 1:1
!
```

Configuration applies in two directions:

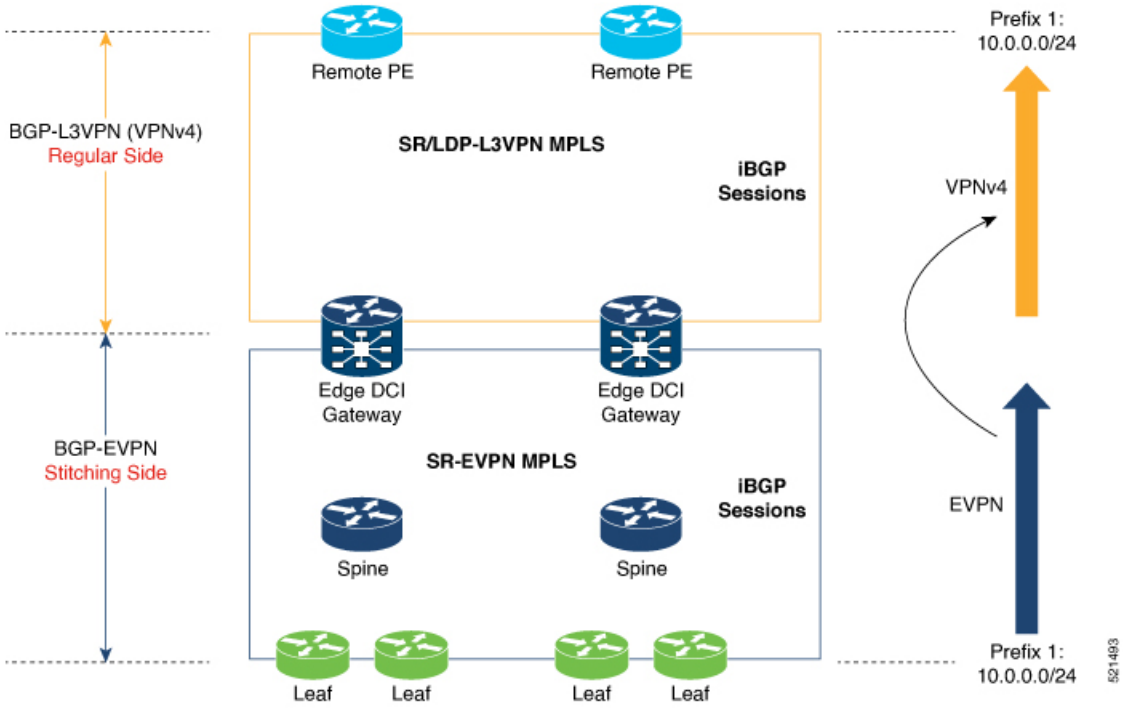
- Stitching from VPNv4 to EVPN routes. Prefixes received from MPLS L3VPN network and re-originated as EVPN prefixes towards Data Center Spine RR and Leaf (ToR).

1. Importing VPNv4 routes with import **re-originate stitching-rt** command under AFI VPNv4 UNICAST. This command imports routes using RT 1:1 and then reoriginate with BGP EVPN 1:2 **stitching-rt**.
2. Advertising re-originated EVPN routes with VPNv4 RT with advertise **vpvn4 unicast re-originated** command under AFI L2VPN EVPN. This command advertises routes from MPLS L3VPN network (VPNv4) to BGP EVPN neighbors inside Data Center (Spine RR and then Leaf (ToR)), re-originating these routes using BGP EVPN 1:2 **stitching-rt**.



- Stitching from EVPN to VPNv4 routes. Prefixes received from BGP-EVPN Data Center and re-originated as MPLS L3VPN prefixes towards VPNv4 RR or remote PE in L3VPN network.

1. Importing EVPN routes with import **stitching-rt re-originate** command under AFI L2VPN EVPN. This command imports routes using RT 1:2 **stitching-rt** and then re-originate with VPNv4 regular/normal VPNv4 RT 1:1.
2. Advertising re-originated EVPN routes with VPNv4 RT with **advertise vpvn4 unicast re-originated** command under AFI VPNv4 UNICAST. This command advertises routes from EVPN Data Center to VPNv4 RR or remote PEs, re-originating these routes using regular/normal VPNv4 RT 1:1.



Verification of Edge DCI Gateway Configuration

```
Router# show bgp l2vpn evpn

Fri Aug 21 00:24:10.773 PDT
BGP router identifier 30.30.30.30, local AS number 100
BGP generic scan interval 60 secs
Non-stop routing is enabled
BGP table state: Active
Table ID: 0x0   RD version: 0
BGP main routing table version 16
BGP NSR Initial initsync version 1 (Reached)
BGP NSR/ISSU Sync-Group versions 16/0
BGP scan interval 60 secs

Status codes: s suppressed, d damped, h history, * valid, > best
     i - internal, r RIB-failure, S stale, N Nexthop-discard
Origin codes: i - IGP, e - EGP, ? - incomplete
   Network            Next Hop            Metric LocPrf Weight Path
Route Distinguisher: 100:1
*>i[2][10000][48][0226.51bd.c81c][32][200::1001]/232
     11.0.0.1                      100      0 i
*>i[2][10000][48][0226.51bd.c81c][32][200:1::1001]/232
     11.0.0.1                      100      0 i
*>i[2][10000][48][0226.51bd.c81c][32][200.1.1.1]/136
        11.0.0.1                      100      0 i
*>i[2][10000][48][0226.51bd.c81c][32][200.1.1.2]/136
     11.0.0.1                      100      0 i
*>i[5][4231][32][100.1.1.1]/80
     11.0.0.1                      100      0 i
*>i[5][4231][32][100.1.1.2]/80
     11.0.0.1                      100      0 i
*>i[5][4231][112][fec0::1001]/176
     11.0.0.1                      100      0 i
```

```
*>i[5][4232][112][fec0::1:1001]/176
     11.0.0.1                 100      0 i

Processed 8 prefixes, 8 paths
```

```
Router# show bgp l2vpn evpn rd 100:1 [5][4231][112][fec0::1001]/176 detail

Fri Aug 21 00:34:43.747 PDT
BGP routing table entry for [5][4231][112][fec0::1001]/176, Route Distinguisher: 100:1
Versions:
  Process           bRIB/RIB  SendTblVer
  Speaker                 5           5
    Flags: 0x04040001+0x00000000;
Last Modified: Aug 21 00:16:58.000 for 00:17:46
Paths: (1 available, best #1)
  Not advertised to any peer
  Path #1: Received by speaker 0
  Flags: 0x4000600025060005, import: 0x3f
  Not advertised to any peer
  Local
    11.0.0.1 (metric 2) from 20.0.0.1 (11.0.0.1)
      Received Label 16001
      Origin IGP, localpref 100, valid, internal, best, group-best, import-candidate,
reoriginate, not-in-vrf
      Received Path ID 0, Local Path ID 1, version 5
      Extended community: Flags 0x6: RT:1:1
      Originator: 11.0.0.1, Cluster list: 20.20.20.20
      EVPN ESI: ffff.ffff.ffff.ffff.ff01, Gateway Address : fec0::254
```

```
Router# show bgp l2vpn evpn neighbors 20.0.0.1 detail
Fri Aug 21 00:25:37.383 PDT

BGP neighbor is 20.0.0.1
 Remote AS 100, local AS 100, internal link
 Remote router ID 20.20.20.20
  BGP state = Established, up for 00:08:58
  NSR State: NSR Ready
  Last read 00:00:34, Last read before reset 00:00:00
  Hold time is 180, keepalive interval is 60 seconds
  Configured hold time: 180, keepalive: 60, min acceptable hold time: 3
  Last write 00:00:36, attempted 19, written 19
  Second last write 00:01:36, attempted 143, written 143
  Last write before reset 00:00:00, attempted 0, written 0
  Second last write before reset 00:00:00, attempted 0, written 0
  Last write pulse rcvd  Aug 21 00:25:03.667 last full not set pulse count 33
  Last write pulse rcvd before reset 00:00:00
  Socket not armed for io, armed for read, armed for write
  Last write thread event before reset 00:00:00, second last 00:00:00
  Last KA expiry before reset 00:00:00, second last 00:00:00
  Last KA error before reset 00:00:00, KA not sent 00:00:00
  Last KA start before reset 00:00:00, second last 00:00:00
  Precedence: internet
  Non-stop routing is enabled
  Entered Neighbor NSR TCP mode:
    TCP Initial Sync :              Aug 21 00:18:07.291
    TCP Initial Sync Phase Two :    Aug 21 00:18:07.319
    TCP Initial Sync Done :         Aug 21 00:18:08.334
  Multi-protocol capability received
  Neighbor capabilities:            Adv         Rcvd
    Route refresh:                  Yes         Yes
    4-byte AS:                      Yes         Yes
    Address family VPNv4 Unicast:   Yes         No
    Address family VPNv6 Unicast:   Yes         No
```

```
  Address family L2VPN EVPN:      Yes         Yes
 Message stats:
  InQ depth: 0, OutQ depth: 0
                 Last_Sent               Sent  Last_Rcvd               Rcvd
  Open:          Aug 21 00:16:38.087        1  Aug 21 00:16:40.123        1
  Notification:  ---                        0  ---                        0
  Update:        Aug 21 00:24:01.421        9  Aug 21 00:24:03.652       13
  Keepalive:     Aug 21 00:25:01.434        8  Aug 21 00:25:03.667        9
  Route_Refresh: Aug 21 00:24:01.377        3  ---                        0
  Total:                                   21                           23
 Minimum time between advertisement runs is 0 secs
 Inbound message logging enabled, 3 messages buffered
 Outbound message logging enabled, 3 messages buffered

For Address Family: VPNv4 Unicast
 BGP neighbor version 35
 Update group: 0.3 Filter-group: 0.1  No Refresh request being processed
 Advertise Reorigination Enabled
 Advertise AFI EoR can be sent
 Route refresh request: received 0, sent 0
 0 accepted prefixes, 0 are bestpaths
 Cumulative no. of prefixes denied: 0.
 Prefix advertised 4, suppressed 0, withdrawn 0
 Maximum prefixes allowed 2097152
 Threshold for warning message 75%, restart interval 0 min
 AIGP is enabled
 An EoR was not received during read-only mode
 Last ack version 35, Last synced ack version 35
 Outstanding version objects: current 0, max 1
 Additional-paths operation: None
 Send Multicast Attributes

For Address Family: VPNv6 Unicast
 BGP neighbor version 29
 Update group: 0.3 Filter-group: 0.1  No Refresh request being processed
 Advertise Reorigination Enabled
 Advertise AFI EoR can be sent
 Route refresh request: received 0, sent 0
 0 accepted prefixes, 0 are bestpaths
 Cumulative no. of prefixes denied: 0.
 Prefix advertised 0, suppressed 0, withdrawn 0
 Maximum prefixes allowed 1048576
 Threshold for warning message 75%, restart interval 0 min
 AIGP is enabled
 An EoR was not received during read-only mode
 Last ack version 29, Last synced ack version 29
 Outstanding version objects: current 0, max 0
 Additional-paths operation: None
 Send Multicast Attributes
 Advertise VPNv4 routes enabled with Reoriginate,Local with stitching-RT option

For Address Family: L2VPN EVPN
 BGP neighbor version 18
 Update group: 0.2 Filter-group: 0.1  No Refresh request being processed
 Route refresh request: received 0, sent 3
 8 accepted prefixes, 8 are bestpaths
 Cumulative no. of prefixes denied: 0.
 Prefix advertised 4, suppressed 0, withdrawn 6
 Maximum prefixes allowed 2097152
 Threshold for warning message 75%, restart interval 0 min
 AIGP is enabled
 An EoR was received during read-only mode
 Last ack version 18, Last synced ack version 18
 Outstanding version objects: current 0, max 2
```

```
  Additional-paths operation: None
  Send Multicast Attributes
  Advertise VPNv4 routes enabled with Reoriginate, option
  Advertise VPNv6 routes is enabled with Reoriginate, option
  Import Stitching is enabled for this neighbor address-family
  Import Reoriginate is enabled for this neighbor address-family

  Connections established 1; dropped 0
  Local host: 30.0.0.1, Local port: 59405, IF Handle: 0x00000000
  Foreign host: 20.0.0.1, Foreign port: 179
  Last reset 00:00:00
```

At the end of each one AFI VPNv4, VPNv6, or L2VPN EVPN, you can see import and advertise information based on the configuration.

```
Router# show bgp sessions

Fri Aug 21 00:25:57.216 PDT

Neighbor        VRF                  Spk    AS   InQ  OutQ  NBRState     NSRState
20.0.0.1        default                0   100     0     0  Established  NSR Ready[PP]
32.0.0.2        default                0   200     0     0  Established  NSR Ready
```

```
Router# show bgp vpnv4 unicast

Fri Aug 21 00:28:41.253 PDT
BGP router identifier 30.30.30.30, local AS number 100
BGP generic scan interval 60 secs
Non-stop routing is enabled
BGP table state: Active
Table ID: 0x0   RD version: 0
BGP main routing table version 39
BGP NSR Initial initsync version 4 (Reached)
BGP NSR/ISSU Sync-Group versions 39/0
BGP scan interval 60 secs

Status codes: s suppressed, d damped, h history, * valid, > best
              i - internal, r RIB-failure, S stale, N Nexthop-discard
Origin codes: i - IGP, e - EGP, ? - incomplete
   Network            Next Hop            Metric LocPrf Weight Path
Route Distinguisher: 1:1
*> 10.0.0.1/8         32.0.0.2                               0 200 300 i
*> 10.0.0.2/8         32.0.0.2                               0 200 300 i
Route Distinguisher: 30.30.30.30:0 (default for vrf foo)
*> 10.0.0.1/8         32.0.0.2                               0 200 300 i
*> 10.0.0.2/8         32.0.0.2                               0 200 300 i
*>i100.1.1.1/32       172.16.0.1                  100        0 i
*>i100.1.1.2/32       172.16.0.1                  100        0 i
*>i200.1.1.1/32       172.16.0.1                  100        0 i
*>i200.1.1.2/32       172.16.0.1                  100        0 i
```

```
Router# show bgp vpnv4 unicast rd 192.168.0.1 10.0.0.1/8 detail

Fri Aug 21 00:28:57.824 PDT
BGP routing table entry for 10.0.0.1/8, Route Distinguisher: 192.168.0.1
Versions:
  Process           bRIB/RIB  SendTblVer
  Speaker                26          26
    Flags: 0x04103001+0x00000000;
Last Modified: Aug 21 00:24:01.000 for 00:04:58
Paths: (1 available, best #1)
  Advertised to peers (in unique update groups):
```

```
    20.0.0.1
  Path #1: Received by speaker 0
  Flags: 0x4000c00005060001, import: 0x80
  Advertised to peers (in unique update groups):
    20.0.0.1
  200 300
    32.0.0.2 from 32.0.0.2 (40.40.40.40)
      Received Label 24001
      Origin IGP, localpref 100, valid, external, best, group-best, import-candidate,
imported, reoriginated with stitching-rt
      Received Path ID 0, Local Path ID 1, version 26
      Extended community: RT: 1:2
      Source AFI: VPNv4 Unicast, Source VRF: default, Source Route Distinguisher: 1:1


Router# show bgp vrf foo

Fri Aug 21 00:24:36.523 PDT
BGP VRF foo, state: Active
BGP Route Distinguisher: 192.168.0.1:0
VRF ID: 0x60000002
BGP router identifier 3192.168.0.1, local AS number 100
Non-stop routing is enabled
BGP table state: Active
Table ID: 0xe0000011   RD version: 35
BGP main routing table version 35
BGP NSR Initial initsync version 4 (Reached)
BGP NSR/ISSU Sync-Group versions 31/0

Status codes: s suppressed, d damped, h history, * valid, > best
              i - internal, r RIB-failure, S stale, N Nexthop-discard
Origin codes: i - IGP, e - EGP, ? - incomplete
   Network            Next Hop            Metric LocPrf Weight Path
Route Distinguisher: 30.30.30.30:0 (default for vrf foo)
*> 10.0.0.1/8         172.16.0.1                             0 200 300 i
*> 10.0.0.2/8         172.16.0.1                             0 200 300 i
*>i100.1.1.1/32       172.16.0.1                  100      0 i
*>i100.1.1.2/32       172.16.0.1                   100      0 i
*>i200.1.1.1/32       172.16.0.1                   100      0 i
*>i200.1.1.2/32       172.16.0.1                  100      0 i

Processed 6 prefixes, 6 paths


Router# show bgp vrf foo ipv4 unicast 100.1.1.1/32 detail

Mon Dec  8 23:24:50.243 PST
BGP routing table entry for 100.1.1.1/32, Route Distinguisher:
192.168.0.1:0
Versions:
  Process           bRIB/RIB  SendTblVer
  Speaker                43          43
    Local Label: 24001 (with rewrite);
    Flags: 0x05081001+0x00000200;
Last Modified: Dec  8 18:04:21.000 for 05:20:30
Paths: (1 available, best #1)
  Advertised to PE peers (in unique update groups):
    32.0.0.2
  Path #1: Received by speaker 0
  Flags: 0x400061000d060005, import: 0x80
  Advertised to PE peers (in unique update groups):
    32.0.0.2
  Local
    172.16.0.1 (metric 2) from 20.0.0.1 (172.16.0.1)
      Received Label 1234
```

```
      Origin IGP, localpref 100, valid, internal, best, group-best, import-candidate,
imported, reoriginated
      Received Path ID 0, Local Path ID 1, version 43
      Extended community: RT:1:2
      Originator: 172.16.0.1, Cluster list: 20.20.20.20
      Source AFI: L2VPN EVPN, Source VRF: default, Source Route Distinguisher: 100:1


Router# show bgp vpnv4 unicast update-group

Fri Aug 21 00:27:57.910 PDT

Update group for VPNv4 Unicast, index 0.1:
  Attributes:
    Outbound policy: pass
    First neighbor AS: 200
    Send communities
    Send GSHUT community if originated
    Send extended communities
    4-byte AS capable
    Send Re-originated VPN routes
    Send multicast attributes
    Minimum advertisement interval: 30 secs
  Update group desynchronized: 0
  Sub-groups merged: 0
  Number of refresh subgroups: 0
  Messages formatted: 8, replicated: 8
  All neighbors are assigned to sub-group(s)
    Neighbors in sub-group: 0.2, Filter-Groups num:1
     Neighbors in filter-group: 0.2(RT num: 0)
      32.0.0.2

Update group for VPNv4 Unicast, index 0.3:
  Attributes:
    Neighbor sessions are IPv4
    Internal
    Common admin
    First neighbor AS: 100
    Send communities
    Send GSHUT community if originated
    Send extended communities
    4-byte AS capable
    Send AIGP
    Send Re-originated VPN routes
    Send multicast attributes
    Minimum advertisement interval: 0 secs
  Update group desynchronized: 0
  Sub-groups merged: 0
  Number of refresh subgroups: 0
  Messages formatted: 2, replicated: 2
  All neighbors are assigned to sub-group(s)
    Neighbors in sub-group: 0.1, Filter-Groups num:1
     Neighbors in filter-group: 0.1(RT num: 0)
      20.0.0.1


Router# show bgp l2vpn evpn update-group

Fri Aug 21 00:27:42.786 PDT

Update group for L2VPN EVPN, index 0.2:
  Attributes:
    Neighbor sessions are IPv4
    Internal
    Common admin
```

```
  First neighbor AS: 100
  Send communities
  Send GSHUT community if originated
  Send extended communities
  4-byte AS capable
  Send AIGP
  Send multicast attributes
  Minimum advertisement interval: 0 secs
Update group desynchronized: 0
Sub-groups merged: 0
Number of refresh subgroups: 0
Messages formatted: 4, replicated: 4
All neighbors are assigned to sub-group(s)
  Neighbors in sub-group: 0.1, Filter-Groups num:1
   Neighbors in filter-group: 0.1(RT num: 0)
    20.0.0.1
```

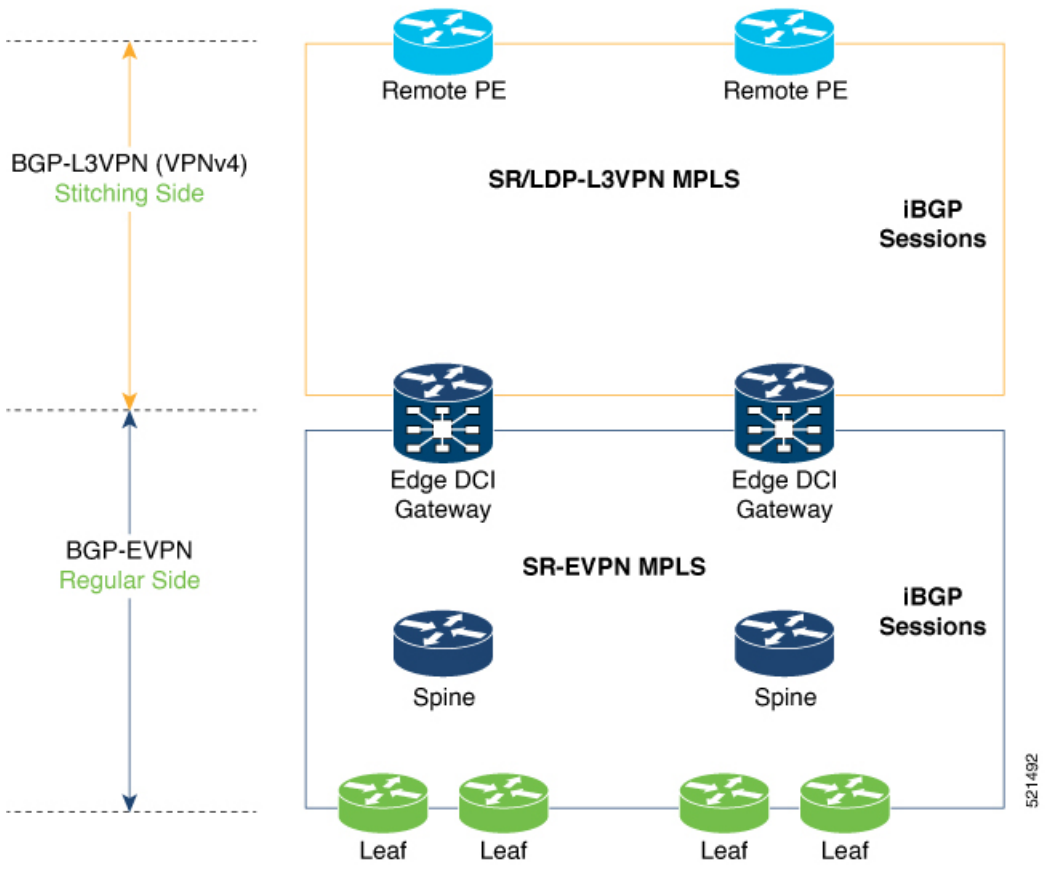
EVPN-Regular RT and VPNv4-Stitching RT

For each VRF on the DCI gateway, there are two sets of manually configured import and export route-targets for EVPN as regular side and VPNv4 as stitching side. Consider the following sets:

- Data Center Route-Targets for EVPN associated with EVPN BGP neighbor (Regular RT)
- MPLS L3VPN Route-Targets for VPNv4 or VPNv6 associated with L3VPN BGP neighbor (Stitching RT)

This separation of RTs enables the two sets of RTs to be independently configured. The RTs associated with the EVPN BGP neighbor does not require the keyword, it remains a normal configuration. The RTs associated with the L3VPN BGP neighbor require **stitching-rt** keyword under VRF configuration.

The following topology shows regular or normal and stitching side.



Route Targets

The RTs associated with the L3VPN BGP neighbor are labelled as stitching RTs. The RTs associated with the EVPN BGP neighbor are normal RTs.

Route Re-Origination

Consider control plane information propagation by the edge DCI gateway from the L3VPN (stitching side) to the Data Center (regular/normal side). Edge DCI gateway advertises to its BGP EVPN neighbor the routes that are re-originated after importing them from the L3VPN BGP neighbor. For this case of VPNv4 or VPNv6 routes being propagated to the BGP EVPN neighbors (Data Center neighbors), re-originating the routes refers to replacing the stitching route-targets with the local route-target values (regular/normal) associated with the BGP EVPN neighbors.

Local VPNv4 or VPNv6 Route Advertisement

On the edge DCI gateway, the locally sourced VPNv4 or VPNv6 routes (any CE directly connected not using L2VPN with BD/EVI/BVI, using only regular L3 VRF) can be advertised to the BGP EVPN neighbors with the normal route targets (RTs) configured for the VRF or the stitching RTs associated with the BGP EVPN neighbors. By default, these routes are advertised with the normal route targets to the BGP EVPN Neighbors (regular/normal side)

VPNv4 neighbors require an additional configuration on the existing legacy VRF to allow these routes to be advertised to VPNv4 RR or remote PEs. Configure **stitching-rt** keyword on existing VRF under import/export RT.

Route Distinguishers

The Router Distinguisher (RD) associated per VRF must be unique per PE in the network. There are few available options to keep unique RD per device:

- Manual configuration: You must manually assign a unique value per device in the network. For example, in this scenario:
  - Leaf (ToR) = RD 1
  - Edge DCI Gateway = RD 2
  - Remote PE = RD 3
- Use **rd auto** command under VRF. To assign a unique route distinguisher for each router, you must ensure that each router has a unique BGP router-id. If so, the **rd auto** command assigns a Type 1 route distinguisher to the VRF using the following format: *ip-address:number*. The IP address is specified by the BGP router-id statement and the number (which is derived as an unused index in the 0 to 65535 range) is unique across the VRFs.

Note In a DCI deployment, for route re-originate with stitching-rt for a particular VRF, using the same Route Distinguisher (RD) between edge DCI gateway and MPLS-VPN PE or same RD between edge DCI gateway and Leaf (ToR) is not supported.

Configure EVPN-Regular RT and VPNv4-Stitching RT

This section describes tasks to configure EVPN-Regular RT and VPNv4-Stitching RT. Perform the following tasks to complete the configuration:

- Configure Leaf (ToR)
- Configure Spine-RR (Route Reflector)
- Configure Edge DCI Gateway
- Configure EVPN BGP neighbor and route advertisements
- Configure L3VPN BGP neighbor relationship and route advertisements

Configure Leaf (ToR)

Configure VRF in Leaf (ToR) at BGP-EVPN (regular/normal side). Note that the **stitching-rt** keyword is not required.

```
vrf data-center1
 address-family ipv4 unicast
  import route-target
   1:2                              // BGP - EVPN (Regular/Normal Side)
!
```

```
  export route-target
   1:2                            // BGP - EVPN (Regular/Normal Side)
  !
router bgp 100
  neighbor 10.10.1.1              // Spine Loopback IP Address
    address-family l2vpn evpn
       advertise vpnv4 unicast
       advertise vpnv6 unicast
!
```

Note Advertise vpnv4/vpnv6 unicast enables local learned regular L3 VRF prefixes to be advertised as EVPN prefixes to BGP-EVPN neighbor. This means any local prefixes such as PE-CE without L2VPN with BD/EVI/BVI configuration. If all the services are pure EVPN with L2VPN with BD/EVI/BVI configuration these commands are not required.

Configure Spine-RR

Configure Spine RR with Leaf (ToR) and edge DCI gateway as RR client for AFI L2VPN EVPN.

```
// VRF Config is not required //
router bgp 100
  neighbor 10.10.2.1                      // Leaf (ToR) Loopback IP Address
    address-family l2vpn evpn
       route-reflector-client
  !
  neighbor 10.10.3.1                      // Edge DCI Gateway  Loopback IP Address
    address-family l2vpn evpn
       route-reflector-client
!
```

Configure Edge DCI Gateway

You can configure DCI with the same VRF as Leaf (ToR). Use the same RT as remote PE for L3VPN network or the same VRF if that is possible.

Configure VRF and Route Targets Import and Export rules

Perform the following steps to configure VRF and define route targets to be used for import and export of forwarding information.

```
vrf data-center1
 address-family ipv4 unicast
  import route-target
   1:1 stitching                  // BGP - L3VPN (Stitching Side)
   1:2                            // BGP - EVPN (Regular/normal Side)
!
  export route-target
   1:1 stitching                  // BGP - L3VPN (Stitching Side)
   1:2                            // BGP - EVPN (Regular/normal Side)
!
```

Configure EVPN BGP Neighbor and Route Advertisements

Perform this task on the edge DCI gateway to configure BGP neighbor relationship and route advertisements with the EVPN BGP neighbor.

```
router bgp 100
 address-family l2vpn evpn
!
 neighbor 10.10.1.1          // Spine Loopback IP Address
  address-family l2vpn evpn
   import re-originate stitching-rt //Imp EVPN RT 1:2, re-originate VPNv4 1:1
   advertise vpnv4 unicast re-originated //Send routes VPNv4 RT 1:1
!
```

Configure L3VPN BGP Neighbor Relationship and Route Advertisements

Perform the following steps to configure BGP neighbor relationship and route advertisements with the L3VPN BGP neighbor.

```
router bgp 100
 address-family vpnv4 unicast
!
 neighbor 10.10.1.1          // Spine Loopback IP Address
  address-family vpnv4 unicast     // Same config for VPNv6
   import stitching-rt re-originate // Imp VPNv4 1:1, reoriginate EVPN 1:2
   advertise vpnv4 unicast re-originated stitching-rt //Send Routes EVPN 1:2
   advertise vpnv6 unicast re-originated stitching-rt //Send Routes EVPN 1:2
!
```

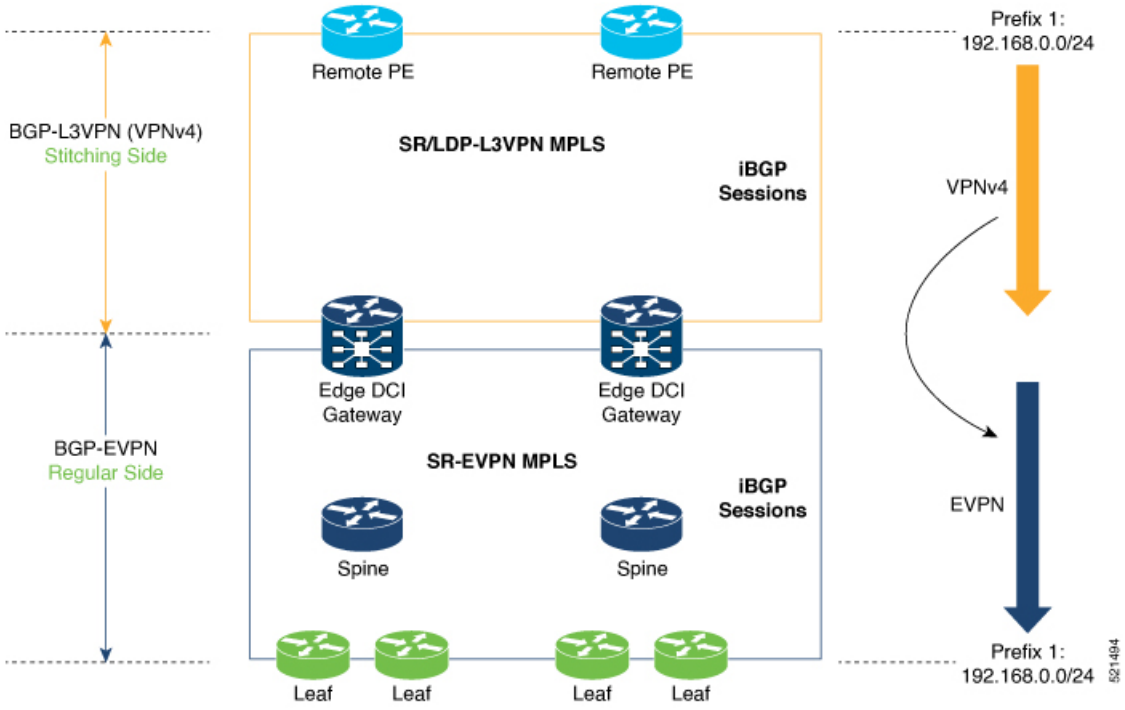
Note The **stitching-rt** applies for L3VPN RT and EVPN RT does not require the **stitching-rt** for this use case.

If there are existing regular local L3 VRF without L2VPN with BD/EVI/BVI in these devices, configure import/export Stitching-RT for existing VRFs to advertise to L3VPN RR or remote PEs.

Configuration applies in two directions:

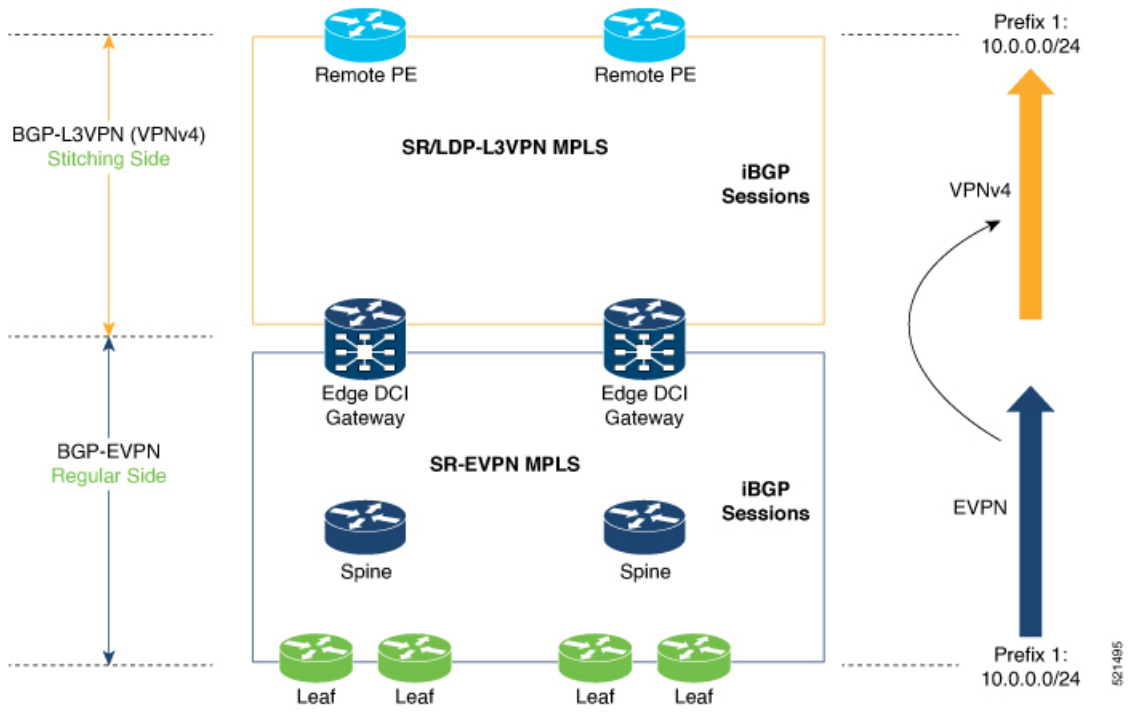
- Stitching from VPNv4 to EVPN routes. Prefixes received from MPLS L3VPN network and re-originated as EVPN prefixes towards Data Center Spine RR and Leaf (ToR)

  1. Importing VPNv4 routes with **import stitching-rt re-originate** command under AFI VPNv4 UNICAST. This command imports routes using RT 1:1 stitching-rt and then re-originate with BGP EVPN 1:2
  2. Advertising re-originated EVPN routes with VPNv4 RT with **advertise vpvn4 unicast re-originated** command under AFI L2VPN EVPN. This command advertises routes from MPLS L3VPN network (VPNv4) to BGP EVPN neighbors inside Data Center (Spine RR and then Leaf (ToR)), re-originating these routes using BGP EVPN 1:2.



- Stitching from EVPN to VPNv4 routes. Prefixes received from BGP-EVPN Data Center and re-originated as MPLS L3VPN prefixes towards VPNv4 RR or remote PE in L3VPN network.

  1. Importing EVPN routes with **import re-originate stitching-rt** command under AFI L2VPN EVPN. This command imports routes using RT 1:2 and then re-originate with VPNv4 RT 1:1 **stitching-rt**.
  2. Advertising re-originated EVPN routes with VPNv4 RT with **advertise vpvn4 unicast re-originated stitching-rt** command under AFI VPNv4 UNICAST. This command advertises routes from EVPN Data Center to VPNv4 RR or remote PEs, re-originating these routes using VPNv4 RT 1:1 **stitching-rt** .



Verification of Edge DCI Gateway Configuration

```
Router# show bgp l2vpn evpn

Fri Aug 21 00:24:10.773 PDT
BGP router identifier 30.30.30.30, local AS number 100
BGP generic scan interval 60 secs
Non-stop routing is enabled
BGP table state: Active
Table ID: 0x0   RD version: 0
BGP main routing table version 16
BGP NSR Initial initsync version 1 (Reached)
BGP NSR/ISSU Sync-Group versions 16/0
BGP scan interval 60 secs

Status codes: s suppressed, d damped, h history, * valid, > best
     i - internal, r RIB-failure, S stale, N Nexthop-discard
Origin codes: i - IGP, e - EGP, ? - incomplete
   Network            Next Hop            Metric LocPrf Weight Path
Route Distinguisher: 100:1
*>i[2][10000][48][0226.51bd.c81c][32][200::1001]/232
     11.0.0.1                       100      0 i
*>i[2][10000][48][0226.51bd.c81c][32][200:1::1001]/232
     11.0.0.1                       100      0 i
*>i[2][10000][48][0226.51bd.c81c][32][200.1.1.1]/136
        11.0.0.1                       100      0 i
*>i[2][10000][48][0226.51bd.c81c][32][200.1.1.2]/136
     11.0.0.1                       100      0 i
*>i[5][4231][32][100.1.1.1]/80
     11.0.0.1                       100      0 i
*>i[5][4231][32][100.1.1.2]/80
     11.0.0.1                       100      0 i
*>i[5][4231][112][fec0::1001]/176
     11.0.0.1                       100      0 i
*>i[5][4232][112][fec0::1:1001]/176
```

```
     11.0.0.1                       100      0 i

Processed 8 prefixes, 8 paths
```

```
Router# show bgp l2vpn evpn rd 100:1 [5][4231][112][fec0::1001]/176 detail
```

```
Fri Aug 21 00:34:43.747 PDT
BGP routing table entry for [5][4231][112][fec0::1001]/176, Route Distinguisher: 100:1
Versions:
  Process           bRIB/RIB  SendTblVer
  Speaker                 5           5
    Flags: 0x04040001+0x00000000;
Last Modified: Aug 21 00:16:58.000 for 00:17:46
Paths: (1 available, best #1)
  Not advertised to any peer
  Path #1: Received by speaker 0
  Flags: 0x4000600025060005, import: 0x3f
  Not advertised to any peer
  Local
    11.0.0.1 (metric 2) from 20.0.0.1 (11.0.0.1)
      Received Label 16001
      Origin IGP, localpref 100, valid, internal, best, group-best, import-candidate,
reoriginate stitching-rt, not-in-vrf
      Received Path ID 0, Local Path ID 1, version 5
      Extended community: Flags 0x6: RT:1:1
      Originator: 11.0.0.1, Cluster list: 20.20.20.20
      EVPN ESI: ffff.ffff.ffff.ffff.ff01, Gateway Address : fec0::254
```

The main difference with scenario 1 is that the prefixes have a **reoriginate stitching-rt** keyword on the output versus scenario 1 having just reoriginate.

```
Router# show bgp l2vpn evpn neighbors 20.0.0.1 detail
```

```
Fri Aug 21 00:25:37.383 PDT

BGP neighbor is 20.0.0.1
 Remote AS 100, local AS 100, internal link
 Remote router ID 20.20.20.20
  BGP state = Established, up for 00:08:58
  NSR State: NSR Ready
  Last read 00:00:34, Last read before reset 00:00:00
  Hold time is 180, keepalive interval is 60 seconds
  Configured hold time: 180, keepalive: 60, min acceptable hold time: 3
  Last write 00:00:36, attempted 19, written 19
  Second last write 00:01:36, attempted 143, written 143
  Last write before reset 00:00:00, attempted 0, written 0
  Second last write before reset 00:00:00, attempted 0, written 0
  Last write pulse rcvd  Aug 21 00:25:03.667 last full not set pulse count 33
  Last write pulse rcvd before reset 00:00:00
  Socket not armed for io, armed for read, armed for write
  Last write thread event before reset 00:00:00, second last 00:00:00
  Last KA expiry before reset 00:00:00, second last 00:00:00
  Last KA error before reset 00:00:00, KA not sent 00:00:00
  Last KA start before reset 00:00:00, second last 00:00:00
  Precedence: internet
  Non-stop routing is enabled
  Entered Neighbor NSR TCP mode:
    TCP Initial Sync :              Aug 21 00:18:07.291
    TCP Initial Sync Phase Two :    Aug 21 00:18:07.319
    TCP Initial Sync Done :         Aug 21 00:18:08.334
  Multi-protocol capability received
  Neighbor capabilities:            Adv         Rcvd
    Route refresh:                  Yes         Yes
    4-byte AS:                      Yes         Yes
    Address family VPNv4 Unicast:   Yes         No
```

```
    Address family VPNv6 Unicast:   Yes         No
    Address family L2VPN EVPN:      Yes         Yes
  Message stats:
    InQ depth: 0, OutQ depth: 0
                    Last_Sent               Sent  Last_Rcvd               Rcvd
    Open:           Aug 21 00:16:38.087        1  Aug 21 00:16:40.123        1
    Notification:   ---                        0  ---                        0
    Update:         Aug 21 00:24:01.421        9  Aug 21 00:24:03.652       13
    Keepalive:      Aug 21 00:25:01.434        8  Aug 21 00:25:03.667        9
    Route_Refresh:  Aug 21 00:24:01.377        3  ---                        0
    Total:                                    21                           23
  Minimum time between advertisement runs is 0 secs
  Inbound message logging enabled, 3 messages buffered
  Outbound message logging enabled, 3 messages buffered

 For Address Family: VPNv4 Unicast
  BGP neighbor version 35
  Update group: 0.3 Filter-group: 0.1  No Refresh request being processed
  Advertise Reorigination Enabled
  Advertise AFI EoR can be sent
  Route refresh request: received 0, sent 0
  0 accepted prefixes, 0 are bestpaths
  Cumulative no. of prefixes denied: 0.
  Prefix advertised 4, suppressed 0, withdrawn 0
  Maximum prefixes allowed 2097152
  Threshold for warning message 75%, restart interval 0 min
  AIGP is enabled
  An EoR was not received during read-only mode
  Last ack version 35, Last synced ack version 35
  Outstanding version objects: current 0, max 1
  Additional-paths operation: None
  Send Multicast Attributes

 For Address Family: VPNv6 Unicast
  BGP neighbor version 29
  Update group: 0.3 Filter-group: 0.1  No Refresh request being processed
  Advertise Reorigination Enabled
  Advertise AFI EoR can be sent
  Route refresh request: received 0, sent 0
  0 accepted prefixes, 0 are bestpaths
  Cumulative no. of prefixes denied: 0.
  Prefix advertised 0, suppressed 0, withdrawn 0
  Maximum prefixes allowed 1048576
  Threshold for warning message 75%, restart interval 0 min
  AIGP is enabled
  An EoR was not received during read-only mode
  Last ack version 29, Last synced ack version 29
  Outstanding version objects: current 0, max 0
  Additional-paths operation: None
  Send Multicast Attributes
  Advertise VPNv4 routes enabled with Reoriginate,Local with stitching-RT option

 For Address Family: L2VPN EVPN
  BGP neighbor version 18
  Update group: 0.2 Filter-group: 0.1  No Refresh request being processed
  Route refresh request: received 0, sent 3
  8 accepted prefixes, 8 are bestpaths
  Cumulative no. of prefixes denied: 0.
  Prefix advertised 4, suppressed 0, withdrawn 6
  Maximum prefixes allowed 2097152
  Threshold for warning message 75%, restart interval 0 min
  AIGP is enabled
  An EoR was received during read-only mode
  Last ack version 18, Last synced ack version 18
```

```
  Outstanding version objects: current 0, max 2
  Additional-paths operation: None
  Send Multicast Attributes
  Advertise VPNv4 routes enabled with Reoriginate, option
  Advertise VPNv6 routes is enabled with Reoriginate, option
  Import Reoriginate is enabled for this neighbor address-family

  Connections established 1; dropped 0
  Local host: 30.0.0.1, Local port: 59405, IF Handle: 0x00000000
  Foreign host: 20.0.0.1, Foreign port: 179
  Last reset 00:00:00
```

At the end of each one AFI VPNv4, VPNv6, or L2VPN EVPN, you can see import and advertise information based on the configuration.

Based on whether stitching-side or regular side, import stitching applies on VPNv4 AFI. In Scenario 1 you can see import stitching under L2VPN EVPN.

```
Router# show bgp sessions

Fri Aug 21 00:25:57.216 PDT

Neighbor        VRF                  Spk    AS   InQ  OutQ  NBRState     NSRState
20.0.0.1        default                0   100     0     0  Established  NSR Ready[PP]
32.0.0.2        default                0   200     0     0  Established  NSR Ready
```

```
Router# show bgp vpnv4 unicast

Fri Aug 21 00:28:41.253 PDT
BGP router identifier 30.30.30.30, local AS number 100
BGP generic scan interval 60 secs
Non-stop routing is enabled
BGP table state: Active
Table ID: 0x0   RD version: 0
BGP main routing table version 39
BGP NSR Initial initsync version 4 (Reached)
BGP NSR/ISSU Sync-Group versions 39/0
BGP scan interval 60 secs

Status codes: s suppressed, d damped, h history, * valid, > best
              i - internal, r RIB-failure, S stale, N Nexthop-discard
Origin codes: i - IGP, e - EGP, ? - incomplete
   Network            Next Hop            Metric LocPrf Weight Path
Route Distinguisher: 1:1
*> 1.1.1.0/24         32.0.0.2                               0 200 300 i
*> 1.1.2.0/24         32.0.0.2                               0 200 300 i
Route Distinguisher: 30.30.30.30:0 (default for vrf foo)
*> 1.1.1.0/24         32.0.0.2                               0 200 300 i
*> 1.1.2.0/24         32.0.0.2                               0 200 300 i
*>i100.1.1.1/32       11.0.0.1                    100        0 i
*>i100.1.1.2/32       11.0.0.1                    100        0 i
*>i200.1.1.1/32       11.0.0.1                    100        0 i
*>i200.1.1.2/32       11.0.0.1                    100        0 i
```

In origin IGP line, you can see that the prefix was reoriginated with regular-RT.

```
Router# show bgp vpnv4 unicast rd 30.30.30.30:0 1.1.1.0/24 detail

Fri Aug 21 00:28:57.824 PDT
BGP routing table entry for 1.1.1.0/24, Route Distinguisher: 30.30.30.30:0
Versions:
  Process           bRIB/RIB  SendTblVer
  Speaker                26          26
    Flags: 0x04103001+0x00000000;
Last Modified: Aug 21 00:24:01.000 for 00:04:58
```

```
Paths: (1 available, best #1)
  Advertised to peers (in unique update groups):
    20.0.0.1
  Path #1: Received by speaker 0
  Flags: 0x4000c00005060001, import: 0x80
  Advertised to peers (in unique update groups):
    20.0.0.1
  200 300
    32.0.0.2 from 32.0.0.2 (40.40.40.40)
      Received Label 24001
      Origin IGP, localpref 100, valid, external, best, group-best, import-candidate,
imported, reoriginated
      Received Path ID 0, Local Path ID 1, version 26
      Extended community: RT: 1:2
      Source AFI: VPNv4 Unicast, Source VRF: default, Source Route Distinguisher: 1:1
```

```
Router# show bgp vrf foo

Fri Aug 21 00:24:36.523 PDT
BGP VRF foo, state: Active
BGP Route Distinguisher: 30.30.30.30:0
VRF ID: 0x60000002
BGP router identifier 30.30.30.30, local AS number 100
Non-stop routing is enabled
BGP table state: Active
Table ID: 0xe0000011   RD version: 35
BGP main routing table version 35
BGP NSR Initial initsync version 4 (Reached)
BGP NSR/ISSU Sync-Group versions 31/0

Status codes: s suppressed, d damped, h history, * valid, > best
              i - internal, r RIB-failure, S stale, N Nexthop-discard
Origin codes: i - IGP, e - EGP, ? - incomplete
   Network            Next Hop            Metric LocPrf Weight Path
Route Distinguisher: 30.30.30.30:0 (default for vrf foo)
*> 1.1.1.0/24         32.0.0.2                               0 200 300 i
*> 1.1.2.0/24         32.0.0.2                               0 200 300 i
*>i100.1.1.1/32       11.0.0.1                    100        0 i
*>i100.1.1.2/32       11.0.0.1                    100        0 i
*>i200.1.1.1/32       11.0.0.1                    100        0 i
*>i200.1.1.2/32       11.0.0.1                    100        0 i

Processed 6 prefixes, 6 paths
```

```
Router# show bgp vrf foo ipv4 unicast 100.1.1.1/32 detail

Mon Dec  8 23:24:50.243 PST
BGP routing table entry for 100.1.1.1/32, Route Distinguisher:
30.30.30.30:0
Versions:
  Process           bRIB/RIB  SendTblVer
  Speaker                43          43
    Local Label: 24001 (with rewrite);
    Flags: 0x05081001+0x00000200;
Last Modified: Dec  8 18:04:21.000 for 05:20:30
Paths: (1 available, best #1)
  Advertised to PE peers (in unique update groups):
    32.0.0.2
  Path #1: Received by speaker 0
  Flags: 0x400061000d060005, import: 0x80
  Advertised to PE peers (in unique update groups):
    32.0.0.2
  Local
    11.0.0.1 (metric 2) from 20.0.0.1 (11.0.0.1)
      Received Label 1234
```

```
      Origin IGP, localpref 100, valid, internal, best, group-best, import-candidate,
imported, reoriginated with stitching-rt
      Received Path ID 0, Local Path ID 1, version 43
      Extended community: RT:1:2
      Originator: 11.0.0.1, Cluster list: 20.20.20.20
      Source AFI: L2VPN EVPN, Source VRF: default, Source Route Distinguisher: 100:1v
```

```
Router# show bgp vpnv4 unicast update-group

Fri Aug 21 00:27:57.910 PDT

Update group for VPNv4 Unicast, index 0.1:
  Attributes:
    Outbound policy: pass
    First neighbor AS: 200
    Send communities
    Send GSHUT community if originated
    Send extended communities
    4-byte AS capable
    Send Re-originated VPN routes
    Send multicast attributes
    Minimum advertisement interval: 30 secs
  Update group desynchronized: 0
  Sub-groups merged: 0
  Number of refresh subgroups: 0
  Messages formatted: 8, replicated: 8
  All neighbors are assigned to sub-group(s)
    Neighbors in sub-group: 0.2, Filter-Groups num:1
     Neighbors in filter-group: 0.2(RT num: 0)
      32.0.0.2

Update group for VPNv4 Unicast, index 0.3:
  Attributes:
    Neighbor sessions are IPv4
    Internal
    Common admin
    First neighbor AS: 100
    Send communities
    Send GSHUT community if originated
    Send extended communities
    4-byte AS capable
    Send AIGP
    Send Re-originated VPN routes
    Send multicast attributes
    Minimum advertisement interval: 0 secs
  Update group desynchronized: 0
  Sub-groups merged: 0
  Number of refresh subgroups: 0
  Messages formatted: 2, replicated: 2
  All neighbors are assigned to sub-group(s)
    Neighbors in sub-group: 0.1, Filter-Groups num:1
     Neighbors in filter-group: 0.1(RT num: 0)
      20.0.0.1
```

```
Router# show bgp l2vpn evpn update-group

Fri Aug 21 00:27:42.786 PDT

Update group for L2VPN EVPN, index 0.2:
  Attributes:
    Neighbor sessions are IPv4
    Internal
    Common admin
    First neighbor AS: 100
    Send communities
```

```
  Send GSHUT community if originated
  Send extended communities
  4-byte AS capable
  Send AIGP
  Send multicast attributes
  Minimum advertisement interval: 0 secs
Update group desynchronized: 0
Sub-groups merged: 0
Number of refresh subgroups: 0
Messages formatted: 4, replicated: 4
All neighbors are assigned to sub-group(s)
  Neighbors in sub-group: 0.1, Filter-Groups num:1
   Neighbors in filter-group: 0.1(RT num: 0)
    20.0.0.1
```

EVPN Default VRF Route Leaking

The EVPN Default VRF Route Leaking feature leak routes between EVPN address-family and IPv4/IPv6 unicast address-family (Default-VRF), enabling the data center hosts to access the Internet. This feature is an extension of Border Gateway Protocol (BGP) VRF Dynamic route leaking feature that provides connectivity between non-default VRF hosts and Default VRF hosts by exchanging routes between the non-default VRF and Default VRF. EVPN Default VRF Route Leaking feature extends the BGP VRF Dynamic leaking feature, by allowing EVPN/L3VPN hosts to communicate with Default VRF hosts.

The import process installs the Internet route in a VRF table or a VRF route in the Internet table, providing connectivity.

The BGP VRF Dynamic route leaking feature is enabled by:

- Importing from default-VRF to non-default-VRF using the following command in VRF address-family configuration mode.

  import from default-vrf route-policy *route-policy-name* **[advertise-as-vpn]**

  If the **advertise-as-vpn** keyword is used, the paths imported from the default-VRF to the non-default-VRF are advertised to the (EVPN/L3VPN) PEs as well as to the CEs. If the **advertise-as-vpn** keyword is not used, the paths imported from the default-VRF to the non-default-VRF are not advertised to the PEs. However, the paths are still advertised to the CEs.

  The EVPN Default VRF Route Leaking feature with **advertise-as-vpn** keyword, enables to advertise the paths imported from default-VRF to non-default VRFs to EVPN PE peers as well.

  A new command **advertise vpnv4/vpnv6 unicast imported-from-default-vrf disable** is added under neighbor address-family configuration mode for EVPN and VPNv4/VPNv6 unicast to disable advertisement of Default-VRF leaked routes to that neighbor.

- Importing from non-default-VRF to default-VRF using the following command in VRF address-family configuration mode.

  export to default-vrf route-policy *route-policy-name* **[advertise-as-vpn]**

  The Dynamic Route Leaking feature enables leaking of local and CE routes to Default-VRF.

  A new optional keyword **allow-imported-vpn** is added to the above command, when configured, enables the leaking of EVPN and L3VPN imported/re-originated routes to the Default-VRF.

A route-policy is mandatory to filter the imported routes. This reduces the risk of unintended import of routes between the Internet table and the VRF tables and the corresponding security issues. There is no hard limit

on the number of prefixes that can be imported. The import creates a new prefix in the destination VRF, which increases the total number of prefixes and paths.

Note Each VRF importing global routes adds workload equivalent to a neighbor receiving the global table. This is true even if the user filters out all but a few prefixes.

Scale Limitation of Default Route Leaking

Default VRF route leaking uses Dynamic Route Leaking feature to leak prefixes between the default VRF and the DC VRF. Do not use Dynamic Route Leaking feature to leak default VRF prefixes to large number of DC VRFs, even if you filter out all prefixes except a few that are to be leaked.

The following are the key factors that affect the performance:

- The default VRF prefix scale, which is approximately 0.7 million internet prefixes.
- The number of DC VRFs the default VRF prefixes that are to be imported.

To improve the scale, either the prefix scale or the number of VRFs whose prefixes that are to be imported must be reduced.

To manage the scale limitation, Cisco recommends you to do the following:

- Host the Internet prefixes on an adjacent PE with IPv4 unicast peering with DCI, and advertise a default route towards the DCI. On the DCI, import the default route from default VRF to DC VRFs.
- Host the Internet prefixes on an adjacent PE with IPv4 unicast peering with DCI. On the DCI, configure a static default route in the DC VRF with the next hop of the default VRF pointing to the adjacent PE address.
- Configure the static default route 0.0.0.0/0 on DC VRF with nexthop as “vrf default”.

Note If the static routes are re-distributed to BGP, make sure it is not unintentionally advertised out.

EVPN Default VRF Route Leaking on the DCI for Internet Connectivity

The EVPN Default VRF Route Leaking feature leak routes between the Default-VRF and Data Center-VRF on the DCI to provide Internet access to data center hosts.

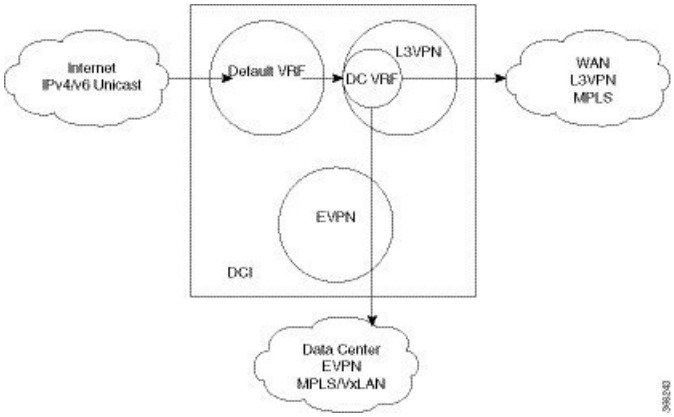
This feature is enabled by:

- Leaking routes from Default-VRF to Data Center-VRF
- Leaking routes to Default-VRF from Data Center-VRF

Leaking Routes from Default-VRF to Data Center-VRF

This section explains the process of leaking Default-VRF routes to Data Center-VRF.

Figure 17: Leaking Routes from Default-VRF to Data Center-VRF



Step 1 The Internet routes are present in the Default-VRF on the DCI.

Note A static default-route (0/0) can be configured under Default-VRF router static address-family configuration and redistributed to BGP.

Step 2 A route-policy is configured to select the routes to be leaked from Default-VRF to Data Center-VRF.

Example:

```
route-policy import-from-default-policy
  if destination in (100.10.0.0/16, 100.20.0.0/16) then
    pass
  endif
end-policy
!

route-policy import-from-default-policy-v6
  if destination in (100:10::0/64, 100:20::0/64) then
    pass
  endif
end-policy
!
```

Note Instead of leaking the internet routes, you can leak the default-route 0/0 from Default-VRF to Data Center-VRF using the following policy.

```
route-policy import-from-default-policy
  if destination in (0.0.0.0/0) then
    pass
  endif
end-policy
!

route-policy import-from-default-policy-v6
  if destination in (0::0/0) then
    pass
  endif
end-policy
!
```

Step 3 Leak Default-VRF routes specified in the route-policy to Data Center-VRF by configuring **import from default-vrf route-policy import-from-default-policy(-v6)** under Data Center VRF address-family configuration mode.

Example:

```
vrf data-center-vrf
 address-family ipv4 unicast
  import from default-vrf route-policy import-from-default-policy
!
address-family ipv6 unicast
  import from default-vrf route-policy import-from-default-policy-v6
!
```

Step 4 Advertise the leaked (Default-VRF) routes in the Data Center-VRF as EVPN routes towards Data Center routers by configuring **advertise-as-vpn** option.

Example:

```
vrf data-center-vrf
 address-family ipv4 unicast
  import from default-vrf route-policy import-from-default-policy advertise-as-vpn
!
address-family ipv6 unicast
  import from default-vrf route-policy import-from-default-policy-v6 advertise-as-vpn
!
```

Note To advertise any routes from L3VPN address-family to EVPN peers, use **advertise vpnv4/vpnv6 unicast re-originated [stitching-rt]** command under neighbor address-family L2VPN EVPN.

EVPN Default-originate

Instead of advertising the Default-VRF routes towards Data Center routers, default-originate can be configured under the EVPN neighbor address-family to advertise the default route. When default-originate is configured under the neighbor address-family for EVPN/L3VPN, there is no need to advertise the Default-VRF leaked routes to the data center and **advertise-as-vpn** need not be configured.

Example:

```
router bgp 100
  neighbor 40.0.0.1
    address-family l2vpn evpn
      default-originate

 vrf data-center-vrf
  rd auto
  address-family ipv4 unicast
    allow vpn default-originate
 !
  address-family ipv6 unicast
    allow vpn default-originate
```

Step 5 To block advertisement of the Default-VRF leaked routes towards a particular EVPN/L3VPN peer, use **advertise vpnv4/vpnv6 unicast imported-from-default-vrf disable** command under respective neighbor address-family.

Example:

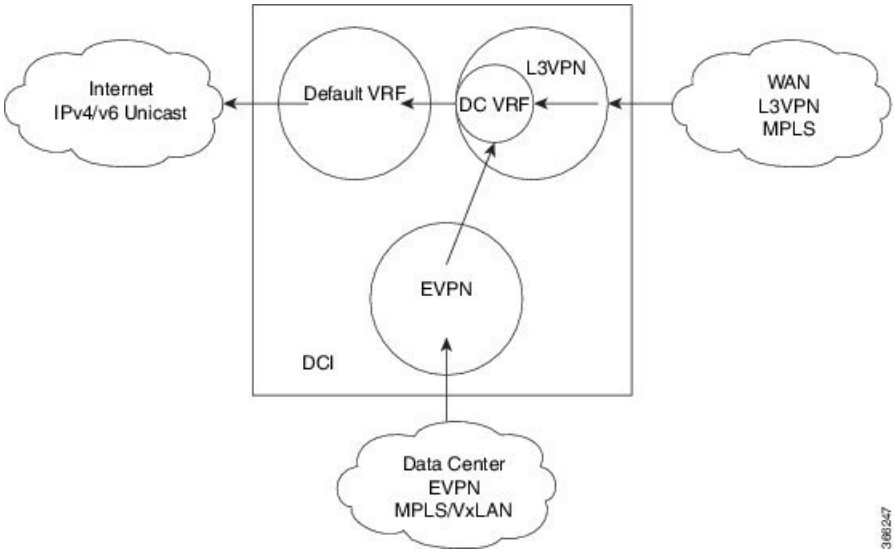
```
router bgp 100
  neighbor 40.0.0.1
    address-family l2vpn evpn
```

```
      advertise vpnv4 unicast imported-from-default-vrf disable
      advertise vpnv6 unicast imported-from-default-vrf disable
!
router bgp 100
  neighbor 60.0.0.1
    address-family vpnv4 unicast
      advertise vpnv4 unicast imported-from-default-vrf disable
    address-family vpnv6 unicast
      advertise vpnv6 unicast imported-from-default-vrf disable
```

Leaking Routes to Default-VRF from Data Center-VRF

This section explains the process of leaking Data Center-VRF routes to Default-VRF.

Figure 18: Leaking Routes to Default-VRF from Data Center-VRF



Step 1 Data Center routes are received on the DCI as EVPN Route-type 2 and Route-type 5 NLRI and imported to the Data Center VRFs.

Step 2 A route-policy is configured to select the routes to be leaked from Data Center-VRF to Default-VRF.

Example:

```
route-policy export-to-default-policy
  if destination in (200.47.0.0/16, 200.168.0.0/16) then
    pass
  endif
end-policy
!

route-policy export-to-default-policy-v6
  if destination in (200:47::0/64, 200:168::0/64) then
    pass
  endif
end-policy
!
```

Step 3 Leak Data Center-VRF routes specified in the above policy to Default-VRF by configuring **export to default-vrf route-policy export-to-default-policy(-v6) [allow-imported-vpn]** under Data Center-VRF address-family configuration mode.

Normally only local and CE VRF routes are allowed to be leaked to the Default-VRF, but **allow-imported-vpn** configuration enables leaking of EVPN/L3VPN imported routes to the Default-VRF.

Example:

```
vrf data-center-vrf
  address-family ipv4 unicast
    export to default-vrf route-policy export-to-default-policy [allow-imported-vpn]
  !
  address-family ipv6 unicast
    export to default-vrf route-policy export-to-default-policy-v6 [allow-imported-vpn]
  !
```

Step 4 The Leaked routes in the Default VRF are advertised to the Internet.

Note Instead of advertising the leaked routes to the Internet, an aggregate can be configured and advertised to the Internet.

Sample Router Configuration

The following sample configuration specifies how EVPN Default VRF Route Leaking feature is configured on a DCI router to provide Internet access to the data center hosts.

```
vrf data-center-vrf
  address-family ipv4 unicast
    import from default-vrf route-policy import-from-default-policy advertise-as-vpn
    export to default-vrf route-policy export-to-default-policy allow-imported-vpn
  !
  address-family ipv6 unicast
    import from default-vrf route-policy import-from-default-policy-v6 advertise-as-vpn
    export to default-vrf route-policy export-to-default-policy-v6 allow-imported-vpn
  !

route-policy import-from-default-policy
  if destination in (100.10.0.0/16, 100.20.0.0/16) then
    pass
  endif
end-policy
!

route-policy import-from-default-policy-v6
  if destination in (100:10::0/64, 100:20::0/64) then
    pass
  endif
end-policy
!

route-policy export-to-default-policy
  if destination in (200.47.0.0/16, 200.168.0.0/16) then
    pass
```

```
  endif
end-policy
!

route-policy export-to-default-policy-v6
  if destination in (200:47::0/64, 200:168::0/64) then
    pass
  endif
end-policy
!

router bgp 100
  neighbor 40.0.0.1
    address-family l2vpn evpn
      import stitching-rt re-originate
      advertise vpnv4 unicast re-originated stitching-rt
      advertise vpnv6 unicast re-originated stitching-rt

  neighbor 60.0.0.1
    address-family vpnv4 unicast
      import re-originate stitching-rt
      advertise vpnv4 unicast re-originated
      advertise vpnv4 unicast imported-from-default-vrf disable

    address-family vpnv6 unicast
      import re-originate stitching-rt
      advertise vpnv6 unicast re-originated
      advertise vpnv6 unicast imported-from-default-vrf disable
```

Sample Router Configuration: with default-originate

The following sample configuration specifies how EVPN Default VRF Route Leaking feature is configured along with default-originate on a DCI router to provide Internet access to data center hosts.

```
vrf data-center-vrf
  address-family ipv4 unicast
    import from default-vrf route-policy import-from-default-policy <= Remove
advertise-as-vpn=>
    export to default-vrf route-policy export-to-default-policy allow-imported-vpn
  !
  address-family ipv6 unicast
    import from default-vrf route-policy import-from-default-policy-v6 <= Remove
advertise-as-vpn=>
    export to default-vrf route-policy export-to-default-policy-v6 allow-imported-vpn
  !
route-policy import-from-default-policy
  if destination in (100.10.0.0/16, 100.20.0.0/16) then
    pass
  endif
end-policy
!
route-policy import-from-default-policy-v6
  if destination in (100:10::0/64, 100:20::0/64) then
    pass
  endif
end-policy
!
route-policy export-to-default-policy
  if destination in (200.47.0.0/16, 200.168.0.0/16) then
    pass
  endif
end-policy
!
```

```
route-policy export-to-default-policy-v6
  if destination in (200:47::0/64, 200:168::0/64) then
    pass
  endif
end-policy
!
router bgp 100
  neighbor 40.0.0.1
    address-family l2vpn evpn
      import stitching-rt re-originate
      advertise vpnv4 unicast re-originated stitching-rt
      advertise vpnv6 unicast re-originated stitching-rt
      default-originate <= Added=>

  neighbor 60.0.0.1
    address-family vpnv4 unicast
      import re-originate stitching-rt
      advertise vpnv4 unicast re-originated
      advertise vpnv4 unicast imported-from-default-vrf disable

    address-family vpnv6 unicast
      import re-originate stitching-rt
      advertise vpnv6 unicast re-originated
      advertise vpnv6 unicast imported-from-default-vrf disable

 vrf data-center-vrf
  rd auto
  address-family ipv4 unicast
    allow vpn default-originate <= Added=>
  !
  address-family ipv6 unicast
    allow vpn default-originate <= Added=>
```

EVPN Service VRF Route Leaking

The EVPN Service VRF Route Leaking feature enables connectivity to the services in the Service VRF to customers in EVPN Data Center VRF. The Service VRF and Data Center VRF routes can be IPv4 and/or IPv6 addresses. The Services VRF is any L3 VRF providing services reachable through connected, static, re-distributed IGP or BGP routes.

This feature leaks routes between Data Center VRF and Service VRF, enabling the EVPN/L3VPN hosts to access the Services in the Service VRF. This feature rely on Border Gateway Protocol (BGP) VRF extranet feature that imports routes between two VRFs.

The import process installs the Data Center VRF routes in a Service VRF table or a Service VRF routes in the Data Center VRF table, providing connectivity.

The BGP Service VRF route leaking feature is enabled by:

- Importing routes from Service VRF to Data Center VRF and advertising it as EVPN/L3VPN route from Data Center VRF.
    - Importing Service VRF routes to Data Center VRF by attaching Data Center VRF import RTs to Service VRF routes.

      This can be achieved by configuring one or more Data Center VRF import RTs as export RT of Service VRF, or configuring a Service VRF export route-policy to attach import RT EXTCOMM to Service VRF routes matching the import RTs of Data Center VRF using the following command in Service VRF address-family configuration mode.

      export route-policy service-vrf-export-route-policy-name

Where the route-policy "service-vrf-export-route-policy-name" attaches the RT EXTCOMM matching the one or more import RTs of Data Center VRF to Service VRF routes.

- Advertising Data Center VRF imported routes that are exported from Service VRFs as EVPN/L3VPN NLRI from Data Center VRF using the following command in Data Center VRF address-family configuration mode.

  import from vrf advertise-as-vpn

  If the **advertise-as-vpn** keyword is used, the paths imported from the Service VRF to the Data Center VRF are advertised to the (EVPN/L3VPN) PEs as well as to the CEs. If the **advertise-as-vpn** keyword is not used, the paths imported from the Service VRF to the Data Center VRF are not advertised to the PEs. However, the paths are still advertised to the CEs.

- Block advertising Data Center VRF leaked routes from being advertised to a neighbor using the following command in neighbor address-family configuration mode.

  advertise vpnv4/vpnv6 unicast imported-from-vrf disable

  A new command **advertise vpnv4/vpnv6 unicast imported-from-vrf disable** is added under neighbor address-family configuration mode for EVPN and VPNv4/VPNv6 unicast to disable advertisement of VRF to VRF leaked routes to that neighbor.

- Importing EVPN/L3VPN routes from Data Center VRF to Service VRF
  - Importing EVPN/L3VPN routes from Data Center VRF to Service VRF by attaching Service VRF import RTs.

    This can be achieved by configuring one or more Service VRF import RTs as export RT of Data Center VRF, or configuring a Data Center VRF export route-policy to attach import RT EXTCOMM to Data Center VRF routes matching the import RTs of Service VRF using the following command in Data Center VRF address-family configuration mode.

    export route-policy data-center-vrf-export-route-policy-name

    The route-policy "data-center-vrf-export-route-policy-name" attaches the RT EXTCOMM matching one or more import RTs of Service VRF.

  - Allow leaking of Data Center VRF routes to Service VRF by using the following command in Data Center VRF address-family configuration mode.

    export to vrf allow-imported-vpn

Note In order to prevent un-intended import of routes to VRFs, select unique RT's to import routes between Service VRF and Data Center VRF, which are not used for normal import of VPN/EVPN routes to Data Center VRFs.

The Extranet Route Leaking feature enables leaking of local and CE routes from one VRF to another VRF. A new command **export to vrf allow-imported-vpn** is added to enable the leaking of EVPN and L3VPN imported/re-originated Data Center VRF routes to the Service VRF.

Note A route-policy is preferred to filter the imported routes. This reduces the risk of unintended import of routes between the Data Center VRF and the Service VRF, and the corresponding security issues. There is no hard limit on the number of prefixes that can be imported. The import creates a new prefix in the destination VRF, which increases the total number of prefixes and paths.

Note This feature does not advertise EVPN/L3VPN PE routes imported to Data Center VRF and leaked to Service VRF as EVPN/L3VPN PE route.

EVPN Service VRF Route Leaking on the DCI for Service Connectivity

The EVPN Service VRF Route Leaking feature leaks routes between the Service VRF and Data Center VRF on the DCI to provide access to Services to data center hosts.

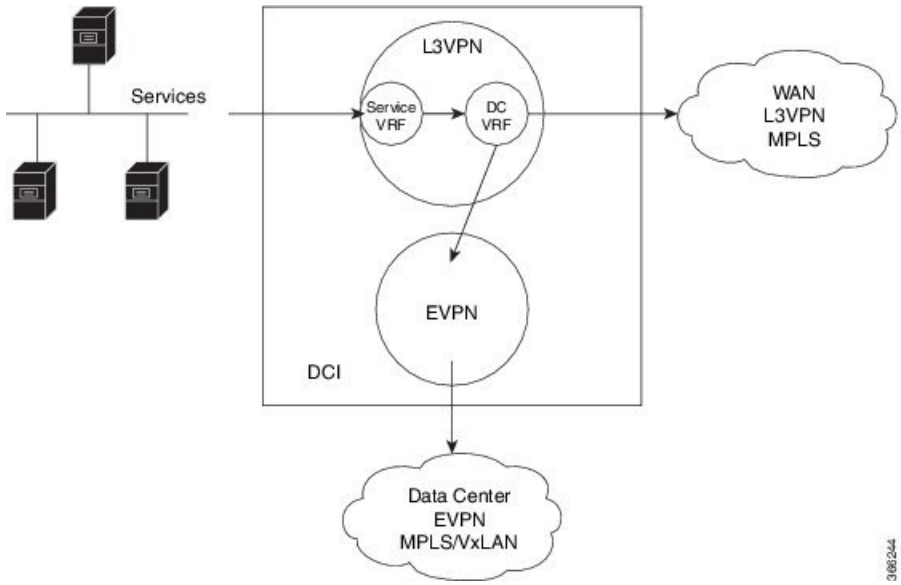
This feature is enabled by:

- Leaking routes from Service VRF to Data Center VRF
- Leaking routes to Service VRF from Data Center VRF

Leaking Routes from Service VRF to Data Center VRF

This section explains the process of leaking Service VRF routes to Data Center VRF.

Figure 19: Leaking Routes from Service VRF to Data Center VRF



Step 1 The Service routes are present in the Service VRF on the DCI.

Step 2 A route-policy is configured to select the routes to be leaked from Service VRF to Data Center VRF.

Example:

```
route-policy service-vrf-export-policy
  if destination in (100.10.0.0/16, 100.20.0.0/16) then
    set extcommunity rt (1:1) additive <--- matches import RT of Data Center-VRF
  endif
end-policy
!
route-policy service-vrf-export-policy-v6
  if destination in (100:10::0/64, 100:20::0/64) then
    set extcommunity rt (1:1) additive <--- matches import RT of Data Center-VRF
  endif
end-policy
!
```

Step 3 Leak Service VRF routes specified in the route-policy to Data Center VRF by configuring **export route-policy service-vrf-export-policy(-v6)** under Service VRF address-family configuration mode.

Example:

```
vrf service-vrf
  address-family ipv4 unicast
    import route-target
      3:1
      4:1 stitching
    export route-policy service-vrf-export-policy
    export route-target
      3:1
      4:1 stitching
   !
  address-family ipv6 unicast
    import route-target
      3:1
      4:1 stitching
    export route-policy service-vrf-export-policy-v6
    export route-target
      3:1
      4:1 stitching
   !
```

Step 4 Advertise the leaked (Service VRF) routes in the Data Center VRF as EVPN/L3VPN routes towards Data Center routers by configuring **import from vrf advertise-as-vpn** under Data Center VRF address-family configuration mode..

Example:

```
vrf data-center-vrf
  address-family ipv4 unicast
    import from vrf advertise-as-vpn
    import route-target
      1:1
      100:1
      200:1 stitching
    export route-target
      100:1
      200:1 stitching
  !
  address-family ipv6 unicast
    import from vrf advertise-as-vpn
    import route-target
      1:1
      100:1
```

```
      200:1 stitching
    export route-target
      100:1
      200:1 stitching
  !
```

Note To advertise any routes from L3VPN address-family to EVPN peers, use **advertise vpnv4/vpnv6 unicast re-originated [stitching-rt]** command under neighbor address-family L2VPN EVPN.

EVPN Default-originate

Instead of advertising the Service VRF routes towards Data Center routers, default-originate can be configured under the EVPN neighbor address-family to advertise the default route. When **allow vpn default-originate** is configured under the Data Center VRF, there is no need to advertise the Service VRF leaked routes to the data center and **advertise-as-vpn** need not be configured.

Example:

```
router bgp 100
  neighbor 40.0.0.1
    address-family l2vpn evpn
      default-originate

 vrf data-center-vrf
  rd auto
  address-family ipv4 unicast
    allow vpn default-originate
 !
  address-family ipv6 unicast
    allow vpn default-originate
```

Step 5 To block advertisement of the Service VRF leaked routes towards a particular EVPN/L3VPN peer, use **advertise vpnv4/vpnv6 unicast imported-from-vrf disable** command under respective neighbor address-family.

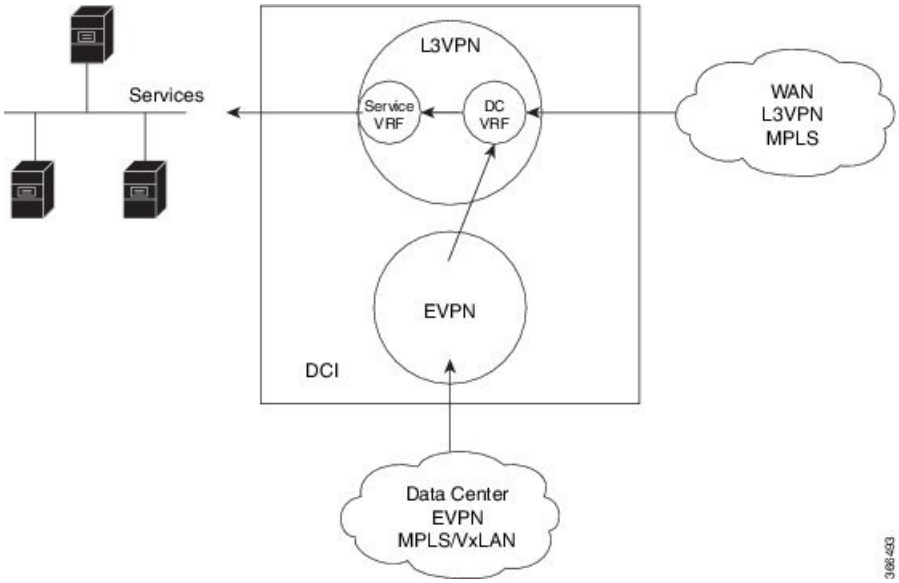
Example:

```
router bgp 100
  neighbor 40.0.0.1
    address-family l2vpn evpn
      import stitching-rt re-originate
      advertise vpnv4 unicast re-originated stitching-rt
      advertise vpnv4 unicast imported-from-vrf disable
      advertise vpnv6 unicast re-originated stitching-rt
      advertise vpnv6 unicast imported-from-vrf disable
    !
router bgp 100
  neighbor 60.0.0.1
    address-family vpnv4 unicast
      import re-originate stitching-rt
      advertise vpnv4 unicast re-originated
      advertise vpnv4 unicast imported-from-vrf disable
    address-family vpnv6 unicast
      import re-originate stitching-rt
      advertise vpnv6 unicast re-originated
      advertise vpnv6 unicast imported-from-vrf disable
```

Leaking Routes to Service VRF from Data Center VRF

This section explains the process of leaking Data Center VRF routes to Service VRF.

Figure 20: Leaking Routes to Service VRF from Data Center VRF



Step 1 Data Center routes are received on the DCI as EVPN Route-type 2 and Route-type 5 NLRI and imported to the Data Center VRFs.

Step 2 A route-policy is configured to select the routes to be leaked from Data Center VRF to Service VRF.

The policy attaches RT EXTCOMM to Data Center VRF routes matching one or more import RT of the Service VRF.

Example:

```
route-policy data-center-vrf-export-policy
  if destination in (200.47.0.0/16) then <--- EVPN PE route
    set extcommunity rt (4:1) additive <--- matches import stitching-RT of service-VRF
  if destination in (200.168.0.0/16) then <--- VPNv4 PE route
    set extcommunity rt (3:1) additive <--- matches import RT of service-VRF
  endif
end-policy
!
route-policy data-center-vrf-export-policy-v6
  if destination in (200:47::0/64) then <--- EVPN PE route
    set extcommunity rt (4:1) additive <--- matches import stitching-RT of service-VRF
  elseif destination in (200:168::0/64) then <--- VPNv6 PE route
    set extcommunity rt (3:1) additive <--- matches import RT of service-VRF
  endif
end-policy
!
```

Note An EVPN/L3VPN route received from a neighbor configured locally with "import stitching-rt re-originate" is imported to Data Center VRF if the route's RT EXTCOMM matches with one or more Data Center VRF import stitching RTs, and is leaked to Service VRF if the Data Center VRF route's RT EXTCOMM matches with one or more Service VRF import stitching RTs.

Step 3 Leak Data Center VRF routes specified in the above policy to Service VRF by configuring **export route-policy data-center-vrf-export-policy(-v6)** under Data Center VRF address-family configuration mode.

Normally only local and CE VRF routes are allowed to be leaked to the Service VRF, but **allow-imported-vpn** configuration enables leaking of EVPN/L3VPN imported routes to the Service VRF.

Example:

```
vrf data-center-vrf
  address-family ipv4 unicast
    import from vrf advertise-as-vpn
    import route-target
      1:1
      100:1
      200:1 stitching
    export route-policy data-center-vrf-export-policy
    export to vrf allow-imported-vpn
    export route-target
      100:1
      200:1 stitching
  !
  address-family ipv6 unicast
    import from vrf advertise-as-vpn
    import route-target
      1:1
      100:1
      200:1 stitching
    export route-policy data-center-vrf-export-policy-v6
    export to vrf allow-imported-vpn
    export route-target
      100:1
      200:1 stitching
  !
```

Step 4 The Data Center VRF leaked routes in the Service VRF are advertised to Service VRF CE peers.

Sample Router Configuration

The following sample configuration specifies how EVPN Service VRF Route Leaking feature is configured on a DCI router providing access to data center hosts to Services in the Service VRF.

```
vrf data-center-vrf
  address-family ipv4 unicast
    import from vrf advertise-as-vpn
    import route-target
      1:1
      100:1
      200:1 stitching
    export route-policy data-center-vrf-export-policy
    export to vrf allow-imported-vpn
    export route-target
      100:1
      200:1 stitching
  !
  address-family ipv6 unicast
    import from vrf advertise-as-vpn
    import route-target
      1:1
      100:1
      200:1 stitching
```

```
    export route-policy data-center-vrf-export-policy-v6
    export to vrf allow-imported-vpn
    export route-target
      100:1
      200:1 stitching
  !

vrf service-vrf
  address-family ipv4 unicast
    import route-target
      3:1
      4:1 stitching
    export route-policy service-vrf-export-policy
    export route-target
      3:1
      4:1 stitching
   !
  address-family ipv6 unicast
    import route-target
      3:1
      4:1 stitching
    export route-policy service-vrf-export-policy-v6
    export route-target
      3:1
      4:1 stitching
   !

route-policy data-center-vrf-export-policy
  if destination in (200.47.0.0/16) then
    set extcommunity rt (4:1) additive
  if destination in (200.168.0.0/16)
    set extcommunity rt (3:1) additive
  endif
end-policy
!

route-policy data-center-vrf-export-policy-v6
  if destination in (200:47::0/64) then
    set extcommunity rt (4:1) additive
  elseif destination in (200:168::0/64)
    set extcommunity rt (3:1) additive
  endif
end-policy
!

route-policy service-vrf-export-policy
  if destination in (100.10.0.0/16, 100.20.0.0/16) then
    set extcommunity rt (1:1) additive
  endif
end-policy
!

route-policy service-vrf-export-policy-v6
  if destination in (100:10::0/64, 100:20::0/64) then
    set extcommunity rt (1:1) additive
  endif
end-policy
!

route-policy pass-all
  pass
end-policy
!
```

```
router bgp 100
  neighbor 40.0.0.1
    remote-as 100
    address-family l2vpn evpn
      import stitching-rt re-originate
      advertise vpnv4 unicast re-originated stitching-rt
      advertise vpnv6 unicast re-originated stitching-rt
    !
  neighbor 60.0.0.1
    remote-as 200
    address-family vpnv4 unicast
      import re-originate stitching-rt
      route-policy pass-all in
      route-policy pass-all out
      advertise vpnv4 unicast re-originated
      advertise vpnv4 unicast imported-from-vrf disable
    address-family vpnv6 unicast
      import re-originate stitching-rt
      route-policy pass-all in
      route-policy pass-all out
      advertise vpnv6 unicast re-originated
      advertise vpnv6 unicast imported-from-vrf disable
```

Sample Router Configuration: with default-originate

The following sample configuration specifies how EVPN Service VRF Route Leaking feature is configured along with default-originate on a DCI router to provide data center hosts access to Services in the Service VRF..

```
vrf data-center-vrf
  address-family ipv4 unicast
    import from vrf advertise-as-vpn
    import route-target
      1:1
      100:1
      200:1 stitching
    export route-policy data-center-vrf-export-policy
    export to vrf allow-imported-vpn
    export route-target
      100:1
      200:1 stitching
  !
  address-family ipv6 unicast
    import from vrf advertise-as-vpn
    import route-target
      1:1
      100:1
      200:1 stitching
    export route-policy data-center-vrf-export-policy-v6
    export to vrf allow-imported-vpn
    export route-target
      100:1
      200:1 stitching
  !

vrf service-vrf
  address-family ipv4 unicast
    import route-target
      3:1
      4:1 stitching
    export route-policy service-vrf-export-policy
    export route-target
      3:1
```

```
      4:1 stitching
   !
  address-family ipv6 unicast
    import route-target
      3:1
      4:1 stitching
    export route-policy service-vrf-export-policy-v6
    export route-target
      3:1
      4:1 stitching
   !

route-policy data-center-vrf-export-policy
  if destination in (200.47.0.0/16) then
    set extcommunity rt (4:1) additive
  if destination in (200.168.0.0/16) then
    set extcommunity rt (3:1) additive
  endif
end-policy
!

route-policy data-center-vrf-export-policy-v6
  if destination in (200:47::0/64) then
    set extcommunity rt (4:1) additive
  elseif destination in (200:168::0/64) then
    set extcommunity rt (3:1) additive
  endif
end-policy
!

route-policy service-vrf-export-policy
  if destination in (100.10.0.0/16, 100.20.0.0/16) then
    set extcommunity rt (1:1) additive
  endif
end-policy
!

route-policy service-vrf-export-policy-v6
  if destination in (100:10::0/64, 100:20::0/64) then
    set extcommunity rt (1:1) additive
  endif
end-policy
!

route-policy pass-all
  pass
end-policy
!

router bgp 100
  neighbor 40.0.0.1
    remote-as 100
    address-family l2vpn evpn
      import stitching-rt re-originate
      advertise vpnv4 unicast re-originated stitching-rt
      advertise vpnv4 unicast imported-from-vrf disable
      advertise vpnv6 unicast re-originated stitching-rt
      advertise vpnv6 unicast imported-from-vrf disable
      default-originate <= Added=>
    !
  neighbor 60.0.0.1
    remote-as 200
    address-family vpnv4 unicast
      import re-originate stitching-rt
```

```
    route-policy pass-all in
    route-policy pass-all out
    advertise vpnv4 unicast re-originated
    advertise vpnv4 unicast imported-from-vrf disable
    default-originate <= Added=>
  address-family vpnv6 unicast
    import re-originate stitching-rt
    route-policy pass-all in
    route-policy pass-all out
    advertise vpnv6 unicast re-originated
    advertise vpnv6 unicast imported-from-vrf disable
    default-originate <= Added=>

vrf data-center-vrf
  rd auto
  address-family ipv4 unicast
    allow vpn default-originate <= Added=>
  !
  address-family ipv6 unicast
    allow vpn default-originate <= Added=>
```

Data Center Interconnect between MPLS-VPN and EVPN-VxLAN

This part provides conceptual and configuration information for Data Center Interconnect (DCI) VXLAN Layer 3 Gateway on the router.

Release	Modification
Release 5.3.2	This feature was introduced.
Release 6.1.x	OpFlex
Release 6.6.x	EVPN VxLAN VRF leaking

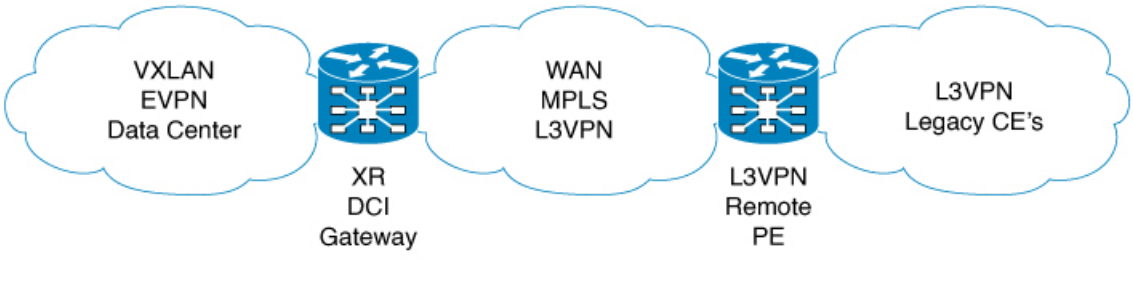
Data Center Interconnect VXLAN Layer 3 Gateway

Router can serve as a Data Center Interconnect (DCI) L3 Gateway using stitching technology between VPNv4/v6 and EVPN-VXLAN. The DCI provides a solution for a new EVPN-VXLAN Data Center that needs to communicate with legacy and existing traditional MPLS VPN networks (VPNv4) having PE-CE architecture.

The DCI L3 gateway provides the following functions:

- **IP connectivity between multi-tenant remote Data Center sites**: Consider the following network topology that has two Data Center sites connected through the intermediate service provider network. The multi-tenant Data Centers use VXLAN encapsulation to carry separate tenant IP traffic. The VXLAN-enabled Data Center sites use the MP-BGP EVPN control plane for distributing both Layer-2 and Layer-3 forwarding information within the site. The router uses MPLS L3VPN application service over the service provider network to provide L3 connectivity between the two Data Center sites. Making this translation between EVPN-VXLAN to VPNv4 overlay.

- **IP Connectivity between Data Center and remote PEs in a legacy network**: Consider the following network topology that has one new Data Center site connected through the intermediate service provider network. The multi-tenant Data Center uses VXLAN encapsulation to carry separate tenant IP traffic. The VXLAN-enabled Data Center site uses the MP-BGP EVPN control plane for distributing both Layer-2 and Layer-3 forwarding information within the site. The router uses MPLS L3VPN application service over the service provider network to provide L3 connectivity between the Data Center services and the legacy CEs using VPNv4 to communicate with services placed inside the Data Center. Making this translation between EVPN-VXLAN to VPNv4 overlay.



Note

- DCI gateway does not provide layer 2 inter-connectivity across Data Centers.
- In a DCI deployment, for route reoriginate with stitching-rt for a particular VRF, using the same Route Distinguisher (RD) between DCI and MPLS-VPN PE or same RD between DCI and VxLAN Top of Rack (ToR) is not supported.

Route Targets

For each VRF on the DCI router, there are two sets of manually configured import and export route-targets. One set of import and export route-targets is associated with the Data Center BGP neighbor that uses EVPN address-family to exchange L3 information; the other set of import and export route-targets is associated with the L3VPN BGP neighbor that use VPNv4 or VPNv6 unicast address-family to exchange L3 information. This separation of route targets (RTs) enables the two sets of RTs to be independently configured. The DCI router effectively stitches the two set of RTs. The RTs associated with the EVPN BGP neighbor are labelled as stitching RTs. The RTs associated with the L3VPN BGP neighbor are normal RTs.

Route Re-origination

Consider the case of control plane information propagation by the DCI from the L3VPN side to the Data Center side. Here, instead of advertising the remote Data Center's original BGP EVPN routes, you can configure the DCI router to advertise to its BGP EVPN neighbor the routes that are re-originated after importing them

from the L3VPN BGP neighbor. For this case of VPNv4 or VPNv6 routes being propagated to the BGP EVPN neighbors (Data Center neighbors), re-originating the routes refers to replacing the normal route-targets with the local route-target values associated with the BGP EVPN neighbors . The converse holds true for the routing information traffic propagation from the BGP EVPN control plane to BGP L3VPN control plane. You can configure this re-origination by using the re-originate keyword in the import re-originate command.. Configuring this command, by default, also enables advertisement of L2VPN EVPN prefixes to the EVPN BGP neighbors. You can suppress native L2VPN EVPN address-family NLRI advertisements towards the EVPN Neighbor using the advertise l2vpn evpn disable command under the EVPN BGP address-family configuration mode.

Route Address-Family and Encoded Address-Family

When an address-family is configured for a BGP neighbor, it means that the specified address-family routes encoded with the NLRI for that address-family is advertised to the neighbor. This does not hold for data center BGP neighbors because they use only EVPN address-family. Here, BGP neighbors advertise VPNv4 or VPNv6 unicast routes using the EVPN NLRI encoding. Thus, here the encoded address-family and route address family can be possibly different. You can advertise the VPNv4 or VPNv6 address-family using the advertise vpnv4 unicast or advertise vpnv6 unicast command. For example, a EVPN address-family BGP neighbor configured with the advertise vpnv4 unicast command sends VPNv4 unicast routes in an EVPN encoded NLRI.

Local VPNv4 or VPNv6 Routes Advertisement

On the DCI router, the locally sourced VPNv4 or VPNv6 routes can be advertised to the BGP EVPN neighbors with the normal route targets (RTs) configured for the VRF or the stitching RTs associated with the BGP EVPN neighbors. By default, these routes are advertised with the normal route targets. You can configure these local VPNv4 or VPNv6 route advertisements to be advertised with stitching RTs to the BGP EVPN neighbors by using the advertise vpnv4 unicast local stitching-rt or advertise vpnv6 unicast local stitching-rt command as required.

Data Center VXLAN with Support for MP-BGP

The Data Center VXLAN uses MP-BGP for control-plane learning of end-host Layer 2 and Layer 3 reachability information. The DCI router is configured with a VXLAN Tunnel EndPoint (VTEP). For VTEP configuration details, see the chapter Implementing Layer 3 VXLAN Gateway. You also need to run the host-reachabilty protocol bgp command to specify that control-plane learning within Data center site is through BGP routing protocol.

The DCI Gateway router and the EVPN BGP neighbor (Data Center BGP neighbor) exchange BGP EVPN NLRIs of route type 5 that carry L3 routing information and associated VXLAN encapsulation information. Some of the VXLAN information is carried in the EVPN NLRI and the rest is carried in RFC 5512 Tunnel Type Encapsulation EXTCOMM and Router MAC EXTCOMM defined in draft-ietf-bess-evpn-inter-subnet-forwarding-00. BGP downloads VXLAN encapsulation as RIB remote next hop opaque attribute to L3RIB.

Default-Originate Forwarding to BGP EVPN Neighbor

Instead of advertising the specific networks available in the remote Data Center, you can configure the DCI gateway to advertise a default route to the directly connected Data Center neighbor. To send the default route for a VRF instance to the Data Center BGP EVPN neighbor, the VPN default-originate information that is typically forwarded to the L3VPN BGP neighbor, is also configured to be forwarded to the BGP EVPN neighbor in the Data Center. To do so, you need to configure allow vpn default-originate command in the BGP VRF configuration mode and also configure default-originate command under EVPN BGP neighbor in L2VPN EVPN address-family configuration mode. This configures BGP to forward only one default route

information for a VRF instance from the DCI Gateway to the BGP neighbor that has L2VPN EVPN address-family. This default route information is encoded in the EVPN "IP Prefix Route" NLRI.

With the advertisement of a default route to the connected Data Center, the DCI Gateway should not advertise specific prefixes of the remote Data Center to the BGP EVPN neighbor. To prevent forwarding of VRF prefixes, you need to configure the DCI gateway with a EVPN BGP neighbor policy that drops forwarding of all prefixes.

Configure Data Center Interconnect Router

Perform the following steps tto configure the Data Center Interconnect (DCI) router:

- Configure VRF and route targets import/export rules
- Configure Bridge Domain for DCI Gateway
- Configure VTEP.
- Configure EVPN BGP neighbor and route advertisement
- Configure L3VPN BGP neighbor relationship and route advertisements

Configure VRF and route targets import/export rules

Perform the following to configure VRF and define route targets to be used for import and export of forwarding information.

```
Router# configure
Router(config)# vrf data-center-10
Router(config-vrf-af)# address-family ipv4 unicast
Router(config-vrf-af)# import route-target 1:1
Router(config-vrf-af)# export route-target 1:2
Router(config-vrf-af)# import route-target 10:1 stitching
Router(config-vrf-af)# export route-target 10:2 stitching
Router(config-vrf-af)# commit
```

Configure Bridge Domain for DCI Gateway

Perform the following to configure the bridge domain on the DCI Gateway.

Note For DCI VxLAN L3 Gateway, only routed interface BVI and member vni can be configured in the bridge-domain. All other L2 services such as EVI, PW, or AC are not supported in the bridge-domain.

```
Router# configure
Router(config)# interface bvi 1
Router(config-if)# vrf cust1
Router(config-if)# ipv4 address 40.1.1.1 255.255.255.254
Router(config)# exit
Router(config)# l2vpn
Router(config-l2vpn)# bridge group bg1
Router(config-l2vpn-bg)# bridge-domain bd1
Router(config-l2vpn-bg-bd)# routed interface BVI1
Router(config-l2vpn-bg-bd)# member vni 5001
Router(config-l2vpn-bg-bd)# commit
```

Configure VTEP (VxLAN Terminal EndPoint) on the DCI Gateway.

Perform the following to configure VTEP (VxLAN Terminal EndPoint) on the DCI Gateway.

```
Router# configure
Router(config)# interface loopback 0
Router(config-if)# ipv4 address 40.1.1.1 255.255.255.255
Router(config)# exit
Router(config)# interface nve 1
Router(config-if)# source interface loopback 0
Router(config-if)# member vni 5001
Router(config-nve-vni)# vrf cust1
Router(config-nve-vni)# host reachability protocol bgp
Router(config-nve-vni)# commit
```

Configure EVPN BGP neighbor and route advertisements

Perform the following on the DCI router to configure BGP neighbor relationship and route advertisements with the EVPN BGP neighbor.

```
Router# configure
Router(config)# router bgp 100
Router(config-bgp)# address-family l2vpn evpn
Router(config-bgp-af)# exit
Router(config-bgp)# neighbor 1.1.1.1
Router(config-bgp-nbr)# remote-as 100
Router(config-bgp-nbr)# address-family l2vpn evpn
Router(config-bgp-nbr)# default-originate /*optional configuration*/
Router(config-bgp-nbr-af)# import stitching-rt reoriginate
Router(config-bgp-nbr-af)# advertise vpnv4 unicast re-originated
Router(config-bgp-nbr-af)# advertise vpnv6 unicast re-originated
Router(config-bgp-nbr-af)# advertise l2vpn evpn disable/*optional configuration*/
Router(config-bgp-nbr-af)# commit
```

Configure L3VPN BGP neighbor relationship and route advertisements

Perform the following to configure BGP neighbor relationship and route advertisements with the L3VPN BGP neighbor.

```
Router# configure
Router(config)# router bgp 100
Router(config-bgp-nbr)# address-family vpnv4
Router(config-bgp-af)# exit
Router(config-bgp)# neighbor 1.1.1.1
Router(config-bgp-nbr)# remote-as 100
Router(config-bgp-nbr)# address-family vpnv4
Router(config-bgp-nbr-af)# import reoriginate stitching-rt
Router(config-bgp-nbr-af)# advertise vpnv4 unicast re-originated


Router# configure
Router(config)# router bgp 100
Router(config-bgp-nbr)# address-family vpnv6
Router(config-bgp-af)# exit
Router(config-bgp)# neighbor 1.1.1.1
Router(config-bgp-nbr)# remote-as 100
Router(config-bgp-nbr)# address-family vpnv6
Router(config-bgp-nbr-af)# import reoriginate stitching-rt
Router(config-bgp-nbr-af)# advertise vpnv6 unicast re-originated
```

Verification

You can use the folllowing show commands to verify the DCI Gateway configurations:

```
Router# show bgp l2vpn evpn

BGP router identifier 30.30.30.30, local AS number 100
BGP generic scan interval 60 secs
Non-stop routing is enabled
BGP table state: Active
Table ID: 0x0   RD version: 0
BGP main routing table version 16
BGP NSR Initial initsync version 1 (Reached)
BGP NSR/ISSU Sync-Group versions 16/0
BGP scan interval 60 secs

Status codes: s suppressed, d damped, h history, * valid, > best
              i - internal, r RIB-failure, S stale, N Nexthop-discard
Origin codes: i - IGP, e - EGP, ? - incomplete
   Network            Next Hop            Metric LocPrf Weight Path
Route Distinguisher: 100:1
*>i[2][10000][48][0226.51bd.c81c][32][200::1001]/232
                      11.0.0.1                 100      0 i
*>i[2][10000][48][0226.51bd.c81c][32][200:1::1001]/232
                      11.0.0.1                 100      0 i
*>i[2][10000][48][0226.51bd.c81c][32][200.1.1.1]/136
                      11.0.0.1                 100      0 i
*>i[2][10000][48][0226.51bd.c81c][32][200.1.1.2]/136
                      11.0.0.1                 100      0 i
*>i[5][4231][32][100.1.1.1]/80
                      11.0.0.1                 100      0 i
*>i[5][4231][32][100.1.1.2]/80
                      11.0.0.1                 100      0 i
*>i[5][4231][112][fec0::1001]/176
                      11.0.0.1                 100      0 i
*>i[5][4232][112][fec0::1:1001]/176
                      11.0.0.1                 100      0 i

Processed 8 prefixes, 8 paths


Router#  show bgp l2vpn evpn rd 100:1 [5][4231][112][fec0::1001]/176 detail

BGP routing table entry for [5][4231][112][fec0::1001]/176, Route Distinguisher: 100:1
Versions:
  Process           bRIB/RIB  SendTblVer
  Speaker                 5           5
    Flags: 0x04040001+0x00000000;
Last Modified: Aug 21 00:16:58.000 for 00:17:46
Paths: (1 available, best #1)
  Not advertised to any peer
  Path #1: Received by speaker 0
  Flags: 0x4000600025060005, import: 0x3f
  Not advertised to any peer
  Local
    11.0.0.1 (metric 2) from 20.0.0.1 (11.0.0.1)
      Received Label 16001
      Origin IGP, localpref 100, valid, internal, best, group-best, import-candidate,
reoriginate, not-in-vrf
      Received Path ID 0, Local Path ID 1, version 5
     Extended community: Flags 0x2: Encapsulation Type:8 Router MAC:aabb.ccdd.eeff RT:65540:1
 RT:40.40.40.40:1 RT:100:1
```

```
      Originator: 11.0.0.1, Cluster list: 20.20.20.20
      EVPN ESI: ffff.ffff.ffff.ffff.ff01, Gateway Address : fec0::254

Router# show bgp l2vpn evpn neighbors 20.0.0.1 detail

BGP neighbor is 20.0.0.1
 Remote AS 100, local AS 100, internal link
 Remote router ID 20.20.20.20
  BGP state = Established, up for 00:08:58
  NSR State: NSR Ready
  Last read 00:00:34, Last read before reset 00:00:00
  Hold time is 180, keepalive interval is 60 seconds
  Configured hold time: 180, keepalive: 60, min acceptable hold time: 3
  Last write 00:00:36, attempted 19, written 19
  Second last write 00:01:36, attempted 143, written 143
  Last write before reset 00:00:00, attempted 0, written 0
  Second last write before reset 00:00:00, attempted 0, written 0
  Last write pulse rcvd  Aug 21 00:25:03.667 last full not set pulse count 33
  Last write pulse rcvd before reset 00:00:00
  Socket not armed for io, armed for read, armed for write
  Last write thread event before reset 00:00:00, second last 00:00:00
  Last KA expiry before reset 00:00:00, second last 00:00:00
  Last KA error before reset 00:00:00, KA not sent 00:00:00
  Last KA start before reset 00:00:00, second last 00:00:00
  Precedence: internet
  Non-stop routing is enabled
  Entered Neighbor NSR TCP mode:
    TCP Initial Sync :              Aug 21 00:18:07.291
    TCP Initial Sync Phase Two :    Aug 21 00:18:07.319
    TCP Initial Sync Done :         Aug 21 00:18:08.334
  Multi-protocol capability received
  Neighbor capabilities:            Adv         Rcvd
    Route refresh:                  Yes         Yes
    4-byte AS:                      Yes         Yes
    Address family VPNv4 Unicast:   Yes         No
    Address family VPNv6 Unicast:   Yes         No
    Address family L2VPN EVPN:      Yes         Yes
  Message stats:
    InQ depth: 0, OutQ depth: 0
                    Last_Sent               Sent  Last_Rcvd               Rcvd
    Open:           Aug 21 00:16:38.087        1  Aug 21 00:16:40.123        1
    Notification:   ---                        0  ---                        0
    Update:         Aug 21 00:24:01.421        9  Aug 21 00:24:03.652       13
    Keepalive:      Aug 21 00:25:01.434        8  Aug 21 00:25:03.667        9
    Route_Refresh:  Aug 21 00:24:01.377        3  ---                        0
    Total:                                    21                           23
  Minimum time between advertisement runs is 0 secs
  Inbound message logging enabled, 3 messages buffered
  Outbound message logging enabled, 3 messages buffered

 For Address Family: VPNv4 Unicast
  BGP neighbor version 35
  Update group: 0.3 Filter-group: 0.1  No Refresh request being processed
  Advertise Reorigination Enabled
  Advertise AFI EoR can be sent
  Route refresh request: received 0, sent 0
  0 accepted prefixes, 0 are bestpaths
  Cumulative no. of prefixes denied: 0.
  Prefix advertised 4, suppressed 0, withdrawn 0
  Maximum prefixes allowed 2097152
  Threshold for warning message 75%, restart interval 0 min
  AIGP is enabled
  An EoR was not received during read-only mode
  Last ack version 35, Last synced ack version 35
```

```
  Outstanding version objects: current 0, max 1
  Additional-paths operation: None
  Send Multicast Attributes

 For Address Family: VPNv6 Unicast
  BGP neighbor version 29
  Update group: 0.3 Filter-group: 0.1  No Refresh request being processed
  Advertise Reorigination Enabled
  Advertise AFI EoR can be sent
  Route refresh request: received 0, sent 0
  0 accepted prefixes, 0 are bestpaths
  Cumulative no. of prefixes denied: 0.
  Prefix advertised 0, suppressed 0, withdrawn 0
  Maximum prefixes allowed 1048576
  Threshold for warning message 75%, restart interval 0 min
  AIGP is enabled
  An EoR was not received during read-only mode
  Last ack version 29, Last synced ack version 29
  Outstanding version objects: current 0, max 0
  Additional-paths operation: None
  Send Multicast Attributes
  Advertise VPNv4 routes enabled with Reoriginate,Local with stitching-RT option

 For Address Family: L2VPN EVPN
  BGP neighbor version 18
  Update group: 0.2 Filter-group: 0.1  No Refresh request being processed
  Route refresh request: received 0, sent 3
  8 accepted prefixes, 8 are bestpaths
  Cumulative no. of prefixes denied: 0.
  Prefix advertised 4, suppressed 0, withdrawn 6
  Maximum prefixes allowed 2097152
  Threshold for warning message 75%, restart interval 0 min
  AIGP is enabled
  An EoR was received during read-only mode
  Last ack version 18, Last synced ack version 18
  Outstanding version objects: current 0, max 2
  Additional-paths operation: None
  Send Multicast Attributes
  Advertise VPNv4 routes enabled with Reoriginate, option
  Advertise VPNv6 routes is enabled with Reoriginate, option
  Import Stitching is enabled for this neighbor address-family
  Import Reoriginate is enabled for this neighbor address-family

  Connections established 1; dropped 0
  Local host: 30.0.0.1, Local port: 59405, IF Handle: 0x00000000
  Foreign host: 20.0.0.1, Foreign port: 179
  Last reset 00:00:00
```

```
Router# show bgp sessions


Neighbor        VRF                  Spk    AS   InQ  OutQ  NBRState     NSRState
20.0.0.1        default                0   100     0     0  Established  NSR Ready[PP]
32.0.0.2        default                0   200     0     0  Established  NSR Ready
```

```
Router#  show bgp vpnv4 unicast
BGP router identifier 30.30.30.30, local AS number 100
BGP generic scan interval 60 secs
Non-stop routing is enabled
BGP table state: Active
Table ID: 0x0   RD version: 0
BGP main routing table version 39
BGP NSR Initial initsync version 4 (Reached)
BGP NSR/ISSU Sync-Group versions 39/0
```

```
BGP scan interval 60 secs

Status codes: s suppressed, d damped, h history, * valid, > best
              i - internal, r RIB-failure, S stale, N Nexthop-discard
Origin codes: i - IGP, e - EGP, ? - incomplete
   Network            Next Hop            Metric LocPrf Weight Path
Route Distinguisher: 1:1
*> 1.1.1.0/24         32.0.0.2                               0 200 300 i
*> 1.1.2.0/24         32.0.0.2                               0 200 300 i
Route Distinguisher: 30.30.30.30:0 (default for vrf foo)
*> 1.1.1.0/24         32.0.0.2                               0 200 300 i
*> 1.1.2.0/24         32.0.0.2                               0 200 300 i
*>i100.1.1.1/32       11.0.0.1                    100        0 i
*>i100.1.1.2/32       11.0.0.1                    100        0 i
*>i200.1.1.1/32       11.0.0.1                    100        0 i
*>i200.1.1.2/32       11.0.0.1                    100        0 i


Router# show bgp vpnv4 unicast rd 30.30.30.30:0 1.1.1.0/24 detail

BGP routing table entry for 1.1.1.0/24, Route Distinguisher: 30.30.30.30:0
Versions:
  Process           bRIB/RIB  SendTblVer
  Speaker                26          26
    Flags: 0x04103001+0x00000000;
Last Modified: Aug 21 00:24:01.000 for 00:04:58
Paths: (1 available, best #1)
  Advertised to peers (in unique update groups):
    20.0.0.1
  Path #1: Received by speaker 0
  Flags: 0x4000c00005060001, import: 0x80
  Advertised to peers (in unique update groups):
    20.0.0.1
  200 300
    32.0.0.2 from 32.0.0.2 (40.40.40.40)
      Received Label 24001
      Origin IGP, localpref 100, valid, external, best, group-best, import-candidate,
imported, reoriginated with stitching-rt
      Received Path ID 0, Local Path ID 1, version 26
      Extended community: RT:100:2
      Source AFI: VPNv4 Unicast, Source VRF: default, Source Route Distinguisher: 1:1


Router# show bgp vrf foo


BGP VRF foo, state: Active
BGP Route Distinguisher: 30.30.30.30:0
VRF ID: 0x60000002
BGP router identifier 30.30.30.30, local AS number 100
Non-stop routing is enabled
BGP table state: Active
Table ID: 0xe0000011   RD version: 35
BGP main routing table version 35
BGP NSR Initial initsync version 4 (Reached)
BGP NSR/ISSU Sync-Group versions 31/0

Status codes: s suppressed, d damped, h history, * valid, > best
              i - internal, r RIB-failure, S stale, N Nexthop-discard
Origin codes: i - IGP, e - EGP, ? - incomplete
   Network            Next Hop            Metric LocPrf Weight Path
Route Distinguisher: 30.30.30.30:0 (default for vrf foo)
*> 1.1.1.0/24         32.0.0.2                               0 200 300 i
*> 1.1.2.0/24         32.0.0.2                               0 200 300 i
*>i100.1.1.1/32       11.0.0.1                    100        0 i
```

```
*>i100.1.1.2/32       11.0.0.1                 100      0 i
*>i200.1.1.1/32       11.0.0.1                 100      0 i
*>i200.1.1.2/32       11.0.0.1                 100      0 i

Processed 6 prefixes, 6 paths


Router#  show bgp vrf foo ipv4 unicast 100.1.1.1/32 detail

BGP routing table entry for 100.1.1.1/32, Route Distinguisher:
30.30.30.30:0
Versions:
  Process           bRIB/RIB  SendTblVer
  Speaker               43          43
    Local Label: 24001 (with rewrite);
    Flags: 0x05081001+0x00000200;
Last Modified: Dec  8 18:04:21.000 for 05:20:30
Paths: (1 available, best #1)
  Advertised to PE peers (in unique update groups):
    32.0.0.2
  Path #1: Received by speaker 0
  Flags: 0x400061000d060005, import: 0x80
  Advertised to PE peers (in unique update groups):
    32.0.0.2
  Local
    11.0.0.1 (metric 2) from 20.0.0.1 (11.0.0.1)
      Received Label 1234
      Origin IGP, localpref 100, valid, internal, best, group-best, import-candidate,
imported, reoriginated
      Received Path ID 0, Local Path ID 1, version 43
      Extended community: Encapsulation Type:8 Router MAC:aabb.ccdd.eeff RT:1:2
      Originator: 11.0.0.1, Cluster list: 20.20.20.20
      RIB RNH: table_id 0xe0000011, Encap 8, VNI 1234, MAC Address: aabb.ccdd.eeff, IP
Address: 11.0.0.1, IP table_id 0xe0000000
      Source AFI: L2VPN EVPN, Source VRF: default, Source Route
Distinguisher: 100:1

Router# show bgp vpnv4 unicast update-group



Update group for VPNv4 Unicast, index 0.1:
  Attributes:
    Outbound policy: pass
    First neighbor AS: 200
    Send communities
    Send GSHUT community if originated
    Send extended communities
    4-byte AS capable
    Send Re-originated VPN routes
    Send multicast attributes
    Minimum advertisement interval: 30 secs
  Update group desynchronized: 0
  Sub-groups merged: 0
  Number of refresh subgroups: 0
  Messages formatted: 8, replicated: 8
  All neighbors are assigned to sub-group(s)
    Neighbors in sub-group: 0.2, Filter-Groups num:1
     Neighbors in filter-group: 0.2(RT num: 0)
      32.0.0.2

Update group for VPNv4 Unicast, index 0.3:
  Attributes:
    Neighbor sessions are IPv4
    Internal
```

```
    Common admin
    First neighbor AS: 100
    Send communities
    Send GSHUT community if originated
    Send extended communities
    4-byte AS capable
    Send AIGP
    Send Re-originated VPN routes
    Send multicast attributes
    Minimum advertisement interval: 0 secs
  Update group desynchronized: 0
  Sub-groups merged: 0
  Number of refresh subgroups: 0
  Messages formatted: 2, replicated: 2
  All neighbors are assigned to sub-group(s)
    Neighbors in sub-group: 0.1, Filter-Groups num:1
     Neighbors in filter-group: 0.1(RT num: 0)
      20.0.0.1
```

```
Router# show bgp l2vpn evpn update-group
```

```
Update group for L2VPN EVPN, index 0.2:
  Attributes:
    Neighbor sessions are IPv4
    Internal
    Common admin
    First neighbor AS: 100
    Send communities
    Send GSHUT community if originated
    Send extended communities
    4-byte AS capable
    Send AIGP
    Send multicast attributes
    Minimum advertisement interval: 0 secs
  Update group desynchronized: 0
  Sub-groups merged: 0
  Number of refresh subgroups: 0
  Messages formatted: 4, replicated: 4
  All neighbors are assigned to sub-group(s)
    Neighbors in sub-group: 0.1, Filter-Groups num:1
     Neighbors in filter-group: 0.1(RT num: 0)
      20.0.0.1
```

Example for configuring Data Center Interconnection Layer 3 Gateway

The following configurations provide an example Data Center Interconnection (DCI) Layer 3 Gateway configuration.

VTEP-related configuration:

```
interface Loopback1
 ipv4 address 40.1.1.1 255.255.255.255
!

interface nve1
 source-interface Loopback1
 member vni 1
  vrf cust1
  host-reachabilty protocol bgp
 !
```

```
interface BVI1
 vrf cust1
 ipv4 address 10.99.1.30 255.255.255.0
 ipv6 address 10:99:1::30/64
!

l2vpn
 bridge group bg1
  bridge-domain bd1
   routed interface BVI1
   member vni 1
  !
 !
```

VRF-related configuration

```
vrf data-center-10
  import route-target 1:1
  export route-target 1:2
  import route-target 10:10 stitching
  export route-target 10:20 stitching
```

Data Center EVPN BGP neighbor-related configuration

```
router bgp 1
 neighbor 1.1.1.1
   address-family l2vpn evpn
     import stitching-rt reoriginate
     advertise vpnv4 unicast reoriginated
     advertise vpnv6 unicast reoriginated
     advertise vpnv4 unicast local stitching-rt
     advertise vpnv6 unicast local stitching-rt
     advertise l2vpn evpn disable
```

L3VPN BGP neighbor-related configuration

```
router bgp 2
neighbor 10.10.10.10
  address-family vpnv4
    import reoriginate stitching-rt
    advertise vpnv4 unicast reoriginated
```

The following example configuration shows how to configure the DCI router to forward default route to its Data Center neighbor.

```
router bgp 1
 address-family vpnv4 unicast
 address-family vpnv6 unicast
 address-family l2vpn evpn
 exit
 neighbor 1.1.1.1
  address-family l2vpn evpn
  default-originate
  exit
 vrf foo
  rd 2:1
  address-family ipv4 unicast
  allow vpn default-originate
  exit
 address-family ipv6 unicast
 allow vpn default-originate
 exit
exit
!
```

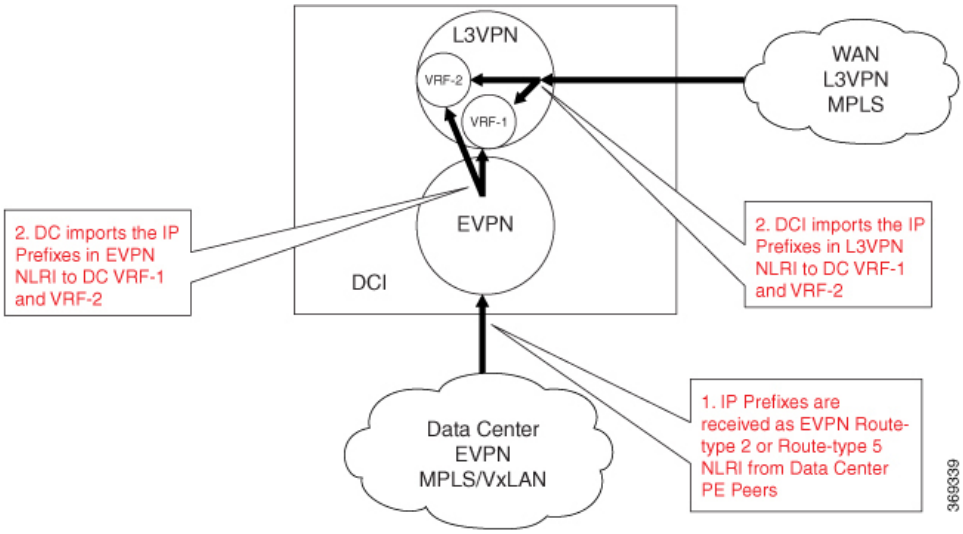
EVPN VxLAN VRF Route Leaking

The EVPN VxLAN VRF Route Leaking feature enables you to import IP prefixes to more than one VRF and leaks IP prefixes from source VRF to destination VRF. This feature supports the following functionalities:

Import L3VPN and EVPN IP Prefixes to more than one VRF

This functionality allows you to import L3VPN and EVPN IP prefixes to more than one VRF when the RT EXTCOMM in the prefix matches import RT of more than one VRF. Starting from Cisco IOS XR Software Release 6.6.2, this functionality is supported for prefixes received with VxLAN tunnel attributes.

Figure 21: Import IP Prefixes to DC VRF-1 and DC VRF-2



Extranet Route Leaking

This functionality allows you to attach the RT EXTCOMM to the prefix that matches the import RT of another VRF. This enables you to leak redistributed or CE VRF IP prefix to another VRF. It is an existing L3VPN functionality to leak redistributed and CE prefixes between VRFs.

Dynamic Route Leaking

This functionality enables you to leak routes between default-VRF and L3VPN VRFs.

For more information about extranet route leaking, see the *Implementing BGP* chapter in the *Routing Configuration Guide for Cisco ASR 9000 Series Routers*.

Leak L3VPN and EVPN Imported IP Prefixes to another VRF

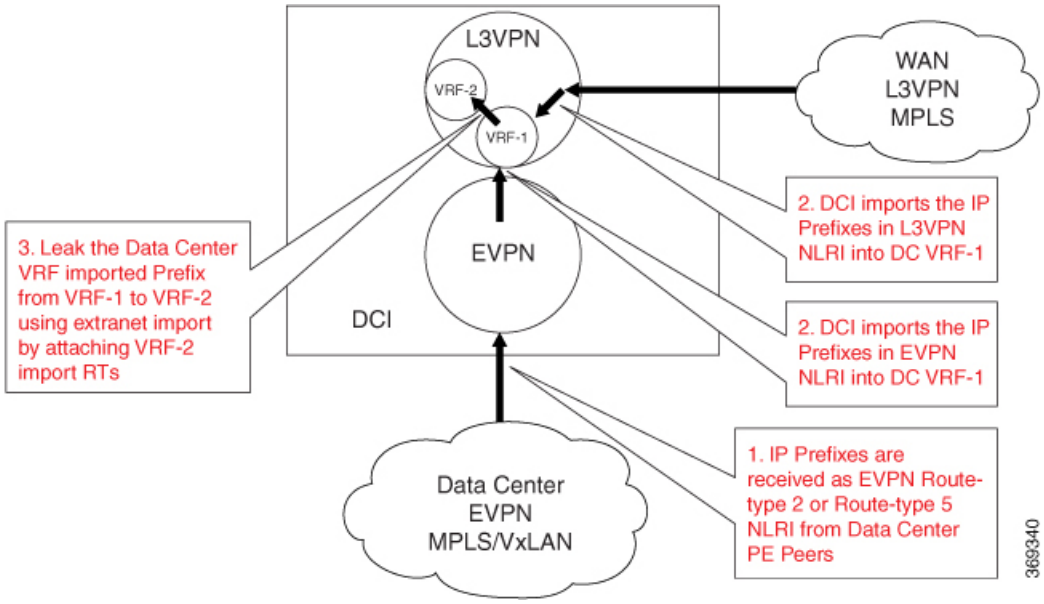
This functionality allows you to attach the RT EXTCOMM to the imported prefix that matches the import RT of another VRF. This enables you to leak L3VPN and EVPN IP prefixes to another VRF.

Configure the following under global VRF address-family mode to enable this functionality:

- Configure the VRF export route-policy in source VRF to attach the import RT of the destination VRF.
- Use the **export to vrf allow-imported-vpn** command under source VRF to enable the leaking of imported prefix from source VRF to destination VRF.

Note Starting from Cisco IOS XR Software Release 6.6.2, this functionality allows you to leak imported L3VPN and EVPN IP prefixes with VxLAN tunnel attributes.

Figure 22: Leak from DC VRF-1 to DC VRF-2



Configuration Example

In this example, the IP address (203.0.113.1/32, 2001:DB8::1/128) in EVPN route-type 2 and prefix (203.0.113.0/24, 2001:DB8::/32) in route-type 5 are imported to VRF-1 and leaked to VRF-2. The RT EXTCOMM (100:1) in NLRI matches one or more import stitching-RTs of VRF-1, and VRF-1 export policy which matches the NLRI IP address and prefix attaches the import stitching-RT (300:1) of VRF-2.

```
RP/0/RSP0/CPU0:router# configure
RP/0/RSP0/CPU0:router(config)# route-policy vrf-leak-from-vrf-1-to-vrf-2
RP/0/RSP0/CPU0:router(config-rpl)# if destination in (203.0.113.1/32, 203.0.113.0/24) then
RP/0/RSP0/CPU0:router(config-rpl-if)# set extcommunity rt (300:1) additive
RP/0/RSP0/CPU0:router(config-rpl-if)# endif
```

```
RP/0/RSP0/CPU0:router(config-rpl)# end-policy
!
RP/0/RSP0/CPU0:router# configure
RP/0/RSP0/CPU0:router(config)# route-policy vrf-leak-from-vrf-1-to-vrf-2-v6
RP/0/RSP0/CPU0:router(config-rpl)# if destination in (2001:DB8::1/128, 2001:DB8::/32) then
RP/0/RSP0/CPU0:router(config-rpl-if)# set extcommunity rt (300:1) additive
RP/0/RSP0/CPU0:router(config-rpl-if)# endif
RP/0/RSP0/CPU0:router(config-rpl)# end-policy
!
RP/0/RSP0/CPU0:router(config)# vrf VRF-1
RP/0/RSP0/CPU0:router(config-vrf)# address-family ipv4 unicast
RP/0/RSP0/CPU0:router(config-vrf-af)# import route-target
RP/0/RSP0/CPU0:router(config-vrf-import-rt)# 1:1
RP/0/RSP0/CPU0:router(config-vrf-import-rt)# 100:1 stitching
!
RP/0/RSP0/CPU0:router(config-vrf-af)# export route-policy vrf-leak-from-vrf-1-to-vrf-2
RP/0/RSP0/CPU0:router(config-vrf-af)# export to vrf allow-imported-vpn
RP/0/RSP0/CPU0:router(config-vrf-af)# export route-target
RP/0/RSP0/CPU0:router(config-vrf-export-rt)#  1:1
RP/0/RSP0/CPU0:router(config-vrf-export-rt)#  100:1 stitching
!
RP/0/RSP0/CPU0:router(config-vrf)# address-family ipv6 unicast
RP/0/RSP0/CPU0:router(config-vrf-af)# import route-target
RP/0/RSP0/CPU0:router(config-vrf-import-rt)#  1:1
RP/0/RSP0/CPU0:router(config-vrf-import-rt)#  100:1 stitching
!
RP/0/RSP0/CPU0:router(config-vrf)# export route-policy vrf-leak-from-vrf-1-to-vrf-2-v6
RP/0/RSP0/CPU0:router(config-vrf-af)# export to vrf allow-imported-vpn
RP/0/RSP0/CPU0:router(config-vrf-af)# export route-target
RP/0/RSP0/CPU0:router(config-vrf-export-rt)# 1:1
RP/0/RSP0/CPU0:router(config-vrf-export-rt)# 100:1 stitching
!
RP/0/RSP0/CPU0:router(config)# vrf VRF-2
RP/0/RSP0/CPU0:router(config-vrf)# address-family ipv4 unicast
RP/0/RSP0/CPU0:router(config-vrf-af)# import route-target
RP/0/RSP0/CPU0:router(config-vrf-import-rt)# 2:1
RP/0/RSP0/CPU0:router(config-vrf-import-rt)# 200:1 stitching
RP/0/RSP0/CPU0:router(config-vrf-import-rt)# 300:1 stitching
!
RP/0/RSP0/CPU0:router(config-vrf-af)# export route-target
RP/0/RSP0/CPU0:router(config-vrf-export-rt)#  2:1
RP/0/RSP0/CPU0:router(config-vrf-export-rt)#  200:1 stitching
!
RP/0/RSP0/CPU0:router(config-vrf)# address-family ipv6 unicast
RP/0/RSP0/CPU0:router(config-vrf-af)# import route-target
RP/0/RSP0/CPU0:router(config-vrf-import-rt)#  2:1
RP/0/RSP0/CPU0:router(config-vrf-import-rt)# 200:1 stitching
RP/0/RSP0/CPU0:router(config-vrf-import-rt)# 300:1 stitching
!
RP/0/RSP0/CPU0:router(config-vrf-af)# export route-target
RP/0/RSP0/CPU0:router(config-vrf-export-rt)# 2:1
RP/0/RSP0/CPU0:router(config-vrf-export-rt)# 200:1 stitching
!
RP/0/RSP0/CPU0:router(config)# router bgp 200
RP/0/RSP0/CPU0:router(config-bgp)# bgp router-id 209.165.200.22
RP/0/RSP0/CPU0:router(config-bgp)# address-family vpnv4 unicast
!
RP/0/RSP0/CPU0:router(config-bgp)# address-family vpnv6 unicast
!
RP/0/RSP0/CPU0:router(config-bgp)# address-family l2vpn evpn
RP/0/RSP0/CPU0:router(config-bgp-af)# neighbor 10.40.0.1
RP/0/RSP0/CPU0:router(config-bgp-nbr)# remote-as 100
RP/0/RSP0/CPU0:router(config-bgp-nbr)# update-source Loopback1
```

```
RP/0/RSP0/CPU0:router(config-bgp-nbr)# address-family l2vpn evpn
RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# import stitching-rt re-originate
RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# route-policy pass in
RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# encapsulation-type vxlan
RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# route-policy pass-all out
RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# advertise vpnv4 unicast re-originated stitching-rt
RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# advertise vpnv6 unicast re-originated stitching-rt
RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# commit
```

Running Configuration

This section shows the running configuration.

```
route-policy vrf-leak-from-vrf-1-to-vrf-2
  if destination in (203.0.113.1/32, 203.0.113.0/24) then
    set extcommunity rt (300:1) additive
  endif
end-policy
!
route-policy vrf-leak-from-vrf-1-to-vrf-2-v6
  if destination in (2001:DB8::1/128, 2001:DB8::/32) then
    set extcommunity rt (300:1) additive
  endif
end-policy
!


vrf VRF-1
 address-family ipv4 unicast
  import route-target
   1:1
   100:1 stitching
  !
  export route-policy vrf-leak-from-vrf-1-to-vrf-2
  export to vrf allow-imported-vpn
  export route-target
   1:1
   100:1 stitching
  !
 !
 address-family ipv6 unicast
  import route-target
   1:1
   100:1 stitching
  !
  export route-policy vrf-leak-from-vrf-1-to-vrf-2-v6
  export to vrf allow-imported-vpn
  export route-target
   1:1
   100:1 stitching


vrf VRF-2
 address-family ipv4 unicast
  import route-target
   2:1
   200:1 stitching
   300:1 stitching
  !
  export route-target
   2:1
   200:1 stitching
  !
```

```
 !
 address-family ipv6 unicast
  import route-target
   2:1
   200:1 stitching
   300:1 stitching
  !
  export route-target
   2:1
   200:1 stitching
  !

router bgp 200
 bgp router-id 209.165.200.22
!
 address-family vpnv4 unicast
!
 address-family vpnv6 unicast
 !
 address-family l2vpn evpn
!
 neighbor 10.40.0.1
  remote-as 100
  update-source Loopback1
  address-family l2vpn evpn
   import stitching-rt re-originate
   route-policy pass in
   encapsulation-type vxlan
   route-policy pass out
   advertise vpnv4 unicast re-originated stitching-rt
   advertise vpnv6 unicast re-originated stitching-rt
  !
 !
```

Advertise Leaked Prefix

This functionality enables you to advertise L3VPN and EVPN prefixes that are leaked from source VRF to destination VRF as L3VPN and EVPN NLRI.

Use the **import from vrf advertise-as-vpn** command under destination VRF address-family, either IPv4 or IPv6 unicast to advertise leaked (CE, redistributed and imported L3VPN and EVPN) prefixes to L3VPN and EVPN PE peers.

Use the **advertise vpnv4 unicast imported-from-vrf disable** or the **advertise vpnv6 unicast imported-from-vrf disable** command under neighbor address-family, either EVPN or L3VPN, to disable advertisement of leaked prefixes from destination VRF to EVPN or L3VPN default VRF neighbors.

Note When the router advertises from VRF-2, it advertises the leaked prefix with VRF-2 route distinguisher, and VRF-2 export route targets (replacing the paths of the existing RTs).

Note The originator loop check prevents the originator from accepting the prefix.

Figure 23: Advertise EVPN Leaked Prefix

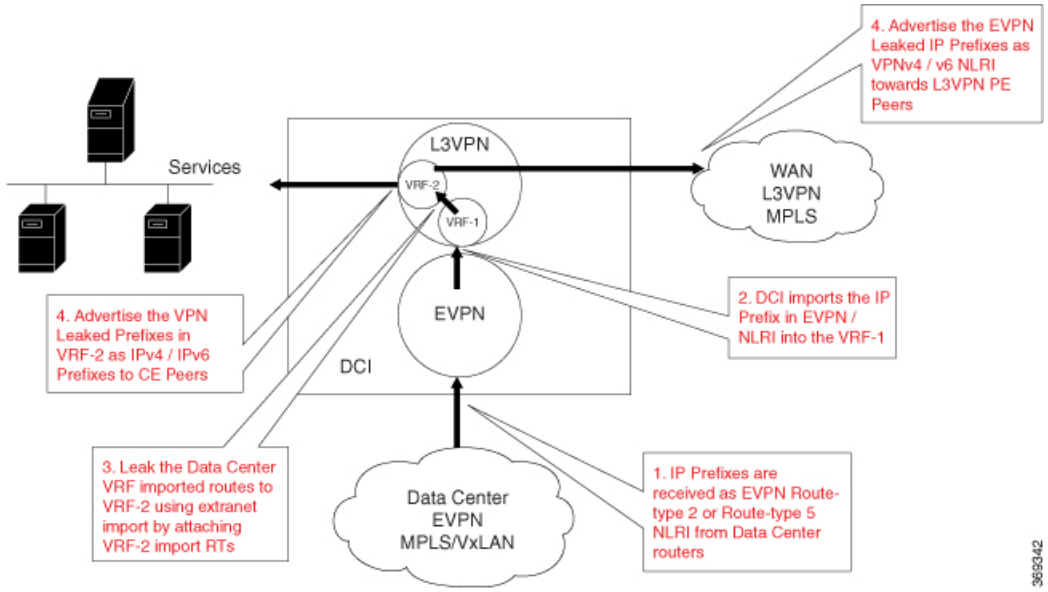
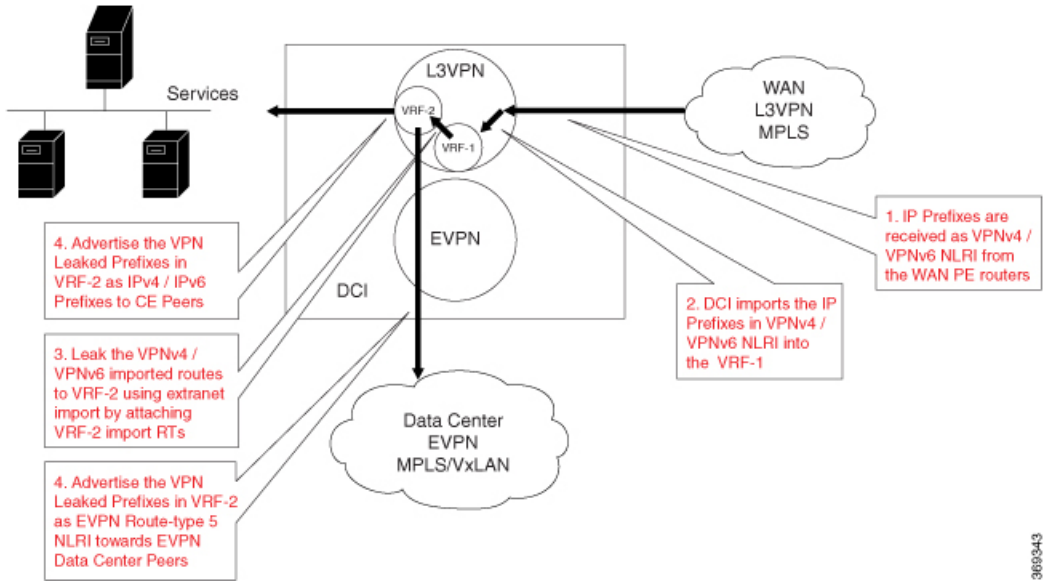


Figure 24: Advertise L3VPN Leaked Prefix



Configuration Example

In this example, the IP address (203.0.113.1/32, 2001:DB8::1/128) in EVPN route-type 2 and prefix (203.0.113.0/24, 2001:DB8::/32) in route-type 5 are imported to VRF-1 and leaked to VRF-2. The RT EXTCOMM (100:1) in NLRI matches one or more import stitching-RTs of VRF-1, and VRF-1 export policy which matches the NLRI IP address and the prefix attaches the import stitching-RT (300:1) of VRF-2.

The router imports prefixes (203.0.113.1/32, 2001:DB8::1/128, 203.0.113.0/24, 2001:DB8::/32) to VRF-1, leaks them from VRF-1 to VRF-2, and advertises them from VRF-2 to L3VPN and EVPN peers.

Use the **advertise vpnv4 unicast imported-from-vrf disable** command under neighbor 10.70.0.1 address-family VPNv4 unicast to block the advertisement of the leaked routes to neighbor 10.70.0.1.

```
RP/0/RSP0/CPU0:router# configure
RP/0/RSP0/CPU0:router(config)# route-policy vrf-leak-from-vrf-1-to-vrf-2
RP/0/RSP0/CPU0:router(config-rpl)# if destination in (203.0.113.1/32, 203.0.113.0/24) then
RP/0/RSP0/CPU0:router(config-rpl-if)# set extcommunity rt (300:1) additive
RP/0/RSP0/CPU0:router(config-rpl-if)# endif
RP/0/RSP0/CPU0:router(config-rpl)# end-policy
!
RP/0/RSP0/CPU0:router# configure
RP/0/RSP0/CPU0:router(config)# route-policy vrf-leak-from-vrf-1-to-vrf-2-v6
RP/0/RSP0/CPU0:router(config-rpl)# if destination in (2001:DB8::1/128, 2001:DB8::/32) then
RP/0/RSP0/CPU0:router(config-rpl-if)# set extcommunity rt (300:1) additive
RP/0/RSP0/CPU0:router(config-rpl-if)# endif
RP/0/RSP0/CPU0:router(config-rpl)# end-policy
!
RP/0/RSP0/CPU0:router(config)# vrf VRF-1
RP/0/RSP0/CPU0:router(config-vrf)# address-family ipv4 unicast
RP/0/RSP0/CPU0:router(config-vrf-af)# import route-target
RP/0/RSP0/CPU0:router(config-vrf-import-rt)# 1:1
RP/0/RSP0/CPU0:router(config-vrf-import-rt)# 100:1 stitching
!
RP/0/RSP0/CPU0:router(config-vrf-af)# export route-policy vrf-leak-from-vrf-1-to-vrf-2
RP/0/RSP0/CPU0:router(config-vrf-af)# export to vrf allow-imported-vpn
RP/0/RSP0/CPU0:router(config-vrf-af)# export route-target
RP/0/RSP0/CPU0:router(config-vrf-export-rt)#  1:1
RP/0/RSP0/CPU0:router(config-vrf-export-rt)#  100:1 stitching
!
RP/0/RSP0/CPU0:router(config-vrf)# address-family ipv6 unicast
RP/0/RSP0/CPU0:router(config-vrf-af)# import route-target
RP/0/RSP0/CPU0:router(config-vrf-import-rt)#  1:1
RP/0/RSP0/CPU0:router(config-vrf-import-rt)#  100:1 stitching
!
RP/0/RSP0/CPU0:router(config-vrf-af)# export route-policy vrf-leak-from-vrf-1-to-vrf-2-v6
RP/0/RSP0/CPU0:router(config-vrf-af)# export to vrf allow-imported-vpn
RP/0/RSP0/CPU0:router(config-vrf-af)# export route-target
RP/0/RSP0/CPU0:router(config-vrf-export-rt)# 1:1
RP/0/RSP0/CPU0:router(config-vrf-export-rt)# 100:1 stitching
!
RP/0/RSP0/CPU0:router(config)# vrf VRF-2
RP/0/RSP0/CPU0:router(config-vrf)# address-family ipv4 unicast
RP/0/RSP0/CPU0:router(config-vrf)# import from vrf advertise-as-vpn
RP/0/RSP0/CPU0:router(config-vrf-af)# import route-target
RP/0/RSP0/CPU0:router(config-vrf-import-rt)# 2:1
RP/0/RSP0/CPU0:router(config-vrf-import-rt)# 200:1 stitching
RP/0/RSP0/CPU0:router(config-vrf-import-rt)# 300:1 stitching
!
RP/0/RSP0/CPU0:router(config-vrf-af)# export route-target
RP/0/RSP0/CPU0:router(config-vrf-export-rt)#  2:1
RP/0/RSP0/CPU0:router(config-vrf-export-rt)#  200:1 stitching
!
RP/0/RSP0/CPU0:router(config-vrf)# address-family ipv6 unicast
RP/0/RSP0/CPU0:router(config-vrf-af)# import from vrf advertise-as-vpn
RP/0/RSP0/CPU0:router(config-vrf-af)# import route-target
RP/0/RSP0/CPU0:router(config-vrf-import-rt)#  2:1
RP/0/RSP0/CPU0:router(config-vrf-import-rt)# 200:1 stitching
RP/0/RSP0/CPU0:router(config-vrf-import-rt)# 300:1 stitching
!
RP/0/RSP0/CPU0:router(config-vrf-af)# export route-target
RP/0/RSP0/CPU0:router(config-vrf-export-rt)# 2:1
```

```
RP/0/RSP0/CPU0:router(config-vrf-export-rt)# 200:1 stitching
!
RP/0/RSP0/CPU0:router(config)# router bgp 200
RP/0/RSP0/CPU0:router(config-bgp)# bgp router-id 172.16.0.1
RP/0/RSP0/CPU0:router(config-bgp)# address-family vpnv4 unicast
!
RP/0/RSP0/CPU0:router(config-bgp)# address-family vpnv6 unicast
!
RP/0/RSP0/CPU0:router(config-bgp)# address-family l2vpn evpn
RP/0/RSP0/CPU0:router(config-bgp-af)# neighbor 10.40.0.1
RP/0/RSP0/CPU0:router(config-bgp-nbr)# remote-as 100
RP/0/RSP0/CPU0:router(config-bgp-nbr)# update-source Loopback1
RP/0/RSP0/CPU0:router(config-bgp-nbr)# address-family l2vpn evpn
RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# import stitching-rt re-originate
RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# route-policy pass in
RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# encapsulation-type vxlan
RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# route-policy pass out
RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# advertise vpnv4 unicast re-originated stitching-rt
RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# advertise vpnv6 unicast re-originated stitching-rt
!

RP/0/RSP0/CPU0:router(config-bgp-af)# neighbor 10.60.0.1
RP/0/RSP0/CPU0:router(config-bgp-nbr)# remote-as 200
RP/0/RSP0/CPU0:router(config-bgp-nbr)# update-source Loopback1
RP/0/RSP0/CPU0:router(config-bgp-nbr)# address-family vpnv4 unicast
RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# import re-originate stitching-rt
RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# advertise vpnv4 unicast re-originated
!

RP/0/RSP0/CPU0:router(config-bgp-nbr)# address-family vpnv6 unicast
RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# import re-originate stitching-rt
RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# advertise vpnv6 unicast re-originated
!

RP/0/RSP0/CPU0:router(config-bgp-af)# neighbor 10.70.0.1
RP/0/RSP0/CPU0:router(config-bgp-nbr)# remote-as 200
RP/0/RSP0/CPU0:router(config-bgp-nbr)# update-source Loopback1
RP/0/RSP0/CPU0:router(config-bgp-nbr)# address-family vpnv4 unicast
RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# import stitching-rt re-originate
RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# advertise vpnv4 unicast re-originated
RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# advertise vpnv4 unicast imported-from-vrf disable
!

RP/0/RSP0/CPU0:router(config-bgp-nbr)# address-family vpnv6 unicast
RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# import re-originate stitching-rt
RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# advertise vpnv6 unicast re-originated
RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# advertise vpnv6 unicast imported-from-vrf disable
RP/0/RSP0/CPU0:router(config-bgp-nbr-af)# commit
```

Running Configuration

This section shows the running configuration.

```
route-policy vrf-leak-from-vrf-1-to-vrf-2
  if destination in (203.0.113.1/32, 203.0.113.0/24) then
    set extcommunity rt (300:1) additive
  endif
end-policy
!
route-policy vrf-leak-from-vrf-1-to-vrf-2-v6
  if destination in (2001:DB8::1/128, 2001:DB8::/32) then
    set extcommunity rt (300:1) additive
```

```
  endif
end-policy
!

vrf VRF-1
 address-family ipv4 unicast
  import route-target
   1:1
   100:1 stitching
  !
  export route-policy vrf-leak-from-vrf-1-to-vrf-2
  export to vrf allow-imported-vpn
  export route-target
   1:1
   100:1 stitching
  !
 !
 address-family ipv6 unicast
  import route-target
   1:1
   100:1 stitching
  !
  export route-policy vrf-leak-from-vrf-1-to-vrf-2-v6
  export to vrf allow-imported-vpn
  export route-target
   1:1
   100:1 stitching


vrf VRF-2
 address-family ipv4 unicast
  import from vrf advertise-as-vpn
  import route-target
   2:1
   200:1 stitching
   300:1 stitching
  !
  export route-target
   2:1
   200:1 stitching
  !
 !
 address-family ipv6 unicast
  import from vrf advertise-as-vpn
  import route-target
   2:1
   200:1 stitching
   300:1 stitching
  !
  export route-target
   2:1
   200:1 stitching
  !

router bgp 200
 bgp router-id 172.16.0.1

!
 address-family vpnv4 unicast
!
 address-family vpnv6 unicast
 !
 address-family l2vpn evpn
!
```

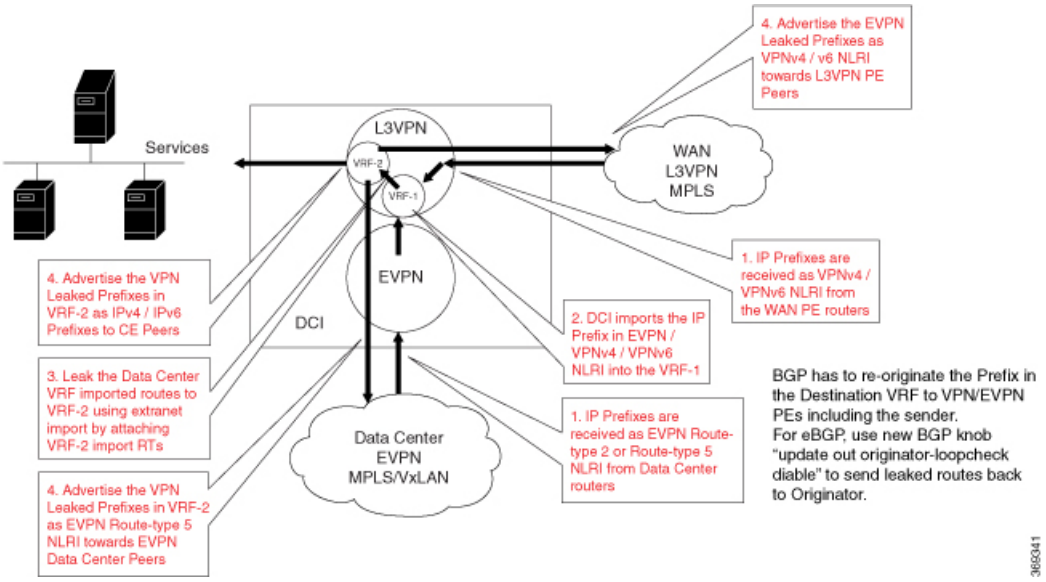
```
 neighbor 10.40.0.1

  remote-as 100
  update-source Loopback1
  address-family l2vpn evpn
   import stitching-rt re-originate
   route-policy pass in
   encapsulation-type vxlan
   route-policy pass out
   advertise vpnv4 unicast re-originated stitching-rt
   advertise vpnv6 unicast re-originated stitching-rt
  !
 !
!
neighbor 10.60.0.1
  remote-as 200
  update-source Loopback1
  address-family vpnv4 unicast
   import re-originate stitching-rt
   advertise vpnv4 unicast re-originated
  !
  address-family vpnv6 unicast
   import re-originate stitching-rt
   advertise vpnv6 unicast re-originated
  !
neighbor 10.70.0.1
  remote-as 200
  update-source Loopback1
  address-family vpnv4 unicast
   import re-originate stitching-rt
   advertise vpnv4 unicast re-originated
   advertise vpnv4 unicast imported-from-vrf disable
  !
  address-family vpnv6 unicast
   import re-originate stitching-rt
   advertise vpnv6 unicast re-originated
   advertise vpnv4 unicast imported-from-vrf disable
!
!
```

Advertise Leaked Prefix Back to Originator

The router advertises leaked L3VPN and EVPN prefixes back to the originator from source VRF to destination VRF as L3VPN or EVPN NLRI.

To provide connectivity between hosts in multiple tenant VRFs, this functionality enables you to leak prefixes between tenant VRFs and advertise the leaked prefixes back to the originator. When source VRF and destination VRF are on the same data center and possibly on the same ToR and Leaf, the leaked prefixes are advertised back to the originator from the DCI.

Figure 25: Advertise Back to Originator



BGP has built-in mechanisms to prevent routing loops, and hence blocks advertisement of prefixes back to the originator. To enable advertisement back to the originator, the prefix update must override sender-side split-horizon check and receiver-side loop checks. BGP has knobs to override sender-side and receiver-side loop checks for iBGP and eBGP.

Service providers prefer eBGP peering in the data center. When the DCI has eBGP peering with the SPINE and the SPINE acting as a route reflector. The SPINE has iBGP peering with TORs in the data center. The following existing configurations impact advertisement of routes from DCI back to originating TOR:

- **as-override** configuration on the DCI router overrides the autonomous system number (ASN) of a data center SPINE with the ASN of the DCI. This configuration disables split-horizon check when the SPINE neighbor is in its own unique update group.
- **as-path-loopcheck out disable** configuration on the DCI router disables AS path loop checking for outbound updates when the SPINE neighbor is in its own unique update group.
- **allowas-in** <*x*> configuration on the SPINE to allow an AS path of data center SPINE autonomous system number (ASN) for a specified number of times. This is needed when the DCI advertises routes back to originator through the SPINE and the **as-override** command is not configured on the DCI.

The above mentioned existing knobs are not adequate to advertise the data center routes from the DCI back to the originating TOR.

Use the **update out originator-loopcheck disable** command under the neighbor configuration to advertise routes back to the originator.

To disable the originator loop checking for outbound updates, use the **update out originator-loopcheck disable** command in the BGP neighbor configuration mode. To re-enable the default originator loop checking, use the **no form** of this command.

Use one of the following configurations to advertise routes received from TOR back to the originating TOR:

- eBGP between DCI and SPINE; and iBGP between the SPINE and TOR1 and TOR2

  Configure the following commands on the DCI:

- **update out originator-loopcheck disable** under SPINE neighbor configuration mode.
- **as-override** under SPINE neighbor address-family configuration mode.

- eBGP between DCI and SPINE; and iBGP between SPINE and TOR1 and TOR2

  Configure the following commands on the DCI:

  - **update out originator-loopcheck disable** under SPINE neighbor configuration mode.
  - **as-path-loopcheck out disable** under global address-family configuration mode.

  Configure the following command on the SPINE:

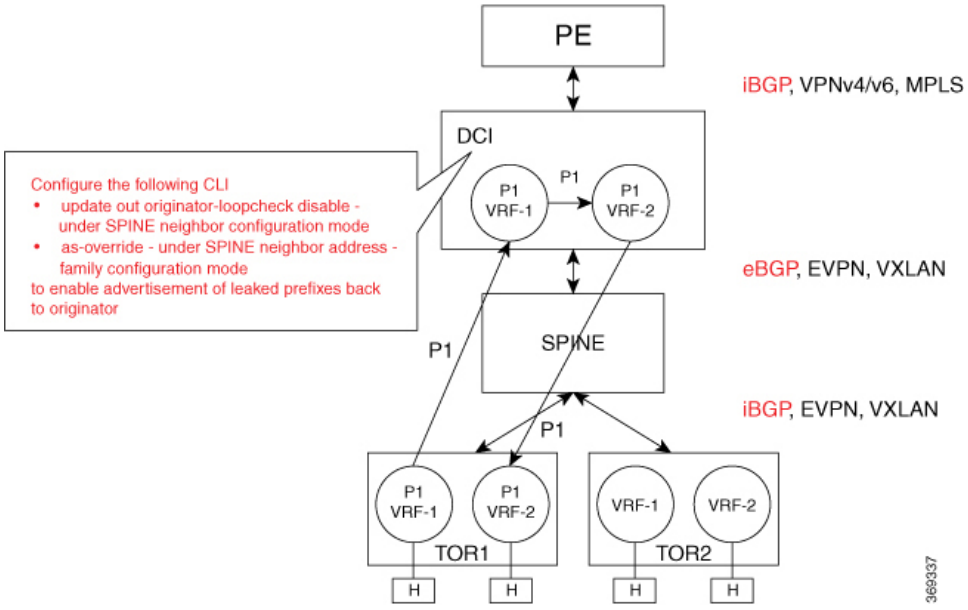
  allowas-in <*x*> command under neighbor address-family configuration mode.

- iBGP between DCI and SPINE; and iBGP between the SPINE and TOR1 and TOR2

  Configure **update out originator-loopcheck disable** command under SPINE neighbor configuration mode.

Configuration Example

Figure 26: Advertise Back to Originator using eBGP



In this example, TOR prefixes are received from the SPINE neighbor 10.40.0.1 on the DCI. The prefixes are imported to VRF-1 and then leaked to VRF-2. The leaked prefixes are advertised from VRF-2 back to the originating neighbor 10.40.0.1.

The example in the previous section *Advertise Leaked Prefix* explains how L3VPN and EVPN prefixes are imported to VRF-1 and how they are leaked to VRF-2 and advertised to L3VPN or EVPN peers, except back to the originator. To advertise the prefix back to the originating neighbor 10.40.0.1, use the **update out originator-loopcheck disable** command under eBGP neighbor configuration mode and **as-override** command under the eBGP neighbor address-family L2VPN EVPN.

```
RP/0/0/CPU0:router#(config)#router bgp 200
RP/0/0/CPU0:router#(config-bgp)#bgp router-id 172.16.0.1
RP/0/0/CPU0:router#(config-bgp)#address-family vpnv4 unicast
!
RP/0/0/CPU0:router#(config-bgp)#address-family vpnv6 unicast
!
RP/0/0/CPU0:router#(config-bgp)#address-family l2vpn evpn
RP/0/0/CPU0:router#(config-bgp-af)#exit
RP/0/0/CPU0:router#(config-bgp)#neighbor 10.40.0.1
RP/0/0/CPU0:router#(config-bgp-nbr)#remote-as 100
RP/0/0/CPU0:router#(config-bgp-nbr)#ebgp-multihop 4
RP/0/0/CPU0:router#(config-bgp-nbr)#update-source Loopback1
RP/0/0/CPU0:router#(config-bgp-nbr)#address-family l2vpn evpn
RP/0/0/CPU0:router#(config-bgp-nbr-af)#update out originator-loopcheck disable
RP/0/0/CPU0:router#(config-bgp-nbr-af)#as-override
RP/0/0/CPU0:router#(config-bgp-nbr-af)#import stitching-rt re-originate
RP/0/0/CPU0:router#(config-bgp-nbr-af)#route-policy pass in
RP/0/0/CPU0:router#(config-bgp-nbr-af)#encapsulation-type vxlan
RP/0/0/CPU0:router#(config-bgp-nbr-af)#route-policy pass out
RP/0/0/CPU0:router#(config-bgp-nbr-af)#advertise vpnv4 unicast re-originated stitching-rt
RP/0/0/CPU0:router#(config-bgp-nbr-af)#advertise vpnv6 unicast re-originated stitching-rt
RP/0/0/CPU0:router#(config-bgp-nbr-af)#commit
```

Running Configuration

This section shows the running configuration.

```
router bgp 200
bgp router-id 172.16.0.1
!
address-family vpnv4 unicast
!
address-family vpnv6 unicast
!
address-family l2vpn evpn
exit
neighbor 10.40.0.1
  remote-as 100
  ebgp-multihop 4
  update-source Loopback1
   address-family l2vpn evpn
    update out originator-loopcheck disable
   as-override
   import stitching-rt re-originate
   route-policy pass in
   encapsulation-type vxlan
   route-policy pass out
   advertise vpnv4 unicast re-originated stitching-rt
   advertise vpnv6 unicast re-originated stitching-rt
  !
!
```

Lookup in Source VRF

Lookup in Source VRF functionality is enabled for an IP prefix when:

- IP prefix is leaked from the source VRF to destination VRF.
- a forwarding lookup for the prefix in destination VRF requires a second look up in the source VRF.

Lookup in source VRF functionality is used when:

- only summary prefix or default-route (0.0.0.0, ::) is leaked from source VRF to destination VRF.
- the longest match prefixes or /32 or /128 host routes are not leaked and are present in the source VRF.
- forwarding requires a match against a longest prefix or /32 or /128 host routes.

This functionality is used in the following scenarios:

- Internet access to the tenant VRF hosts – Where only the default-route (0.0.0.0, ::) is leaked from default VRF to tenant VRF. Internet prefixes are kept in the default VRF. The summary prefixes advertised to the Internet are leaked from tenant VRF to default VRF, the host routes are kept in the tenant VRF.
- Merging two VRFs – Where only the summary prefixes in the two VRFs are leaked to the other VRF.

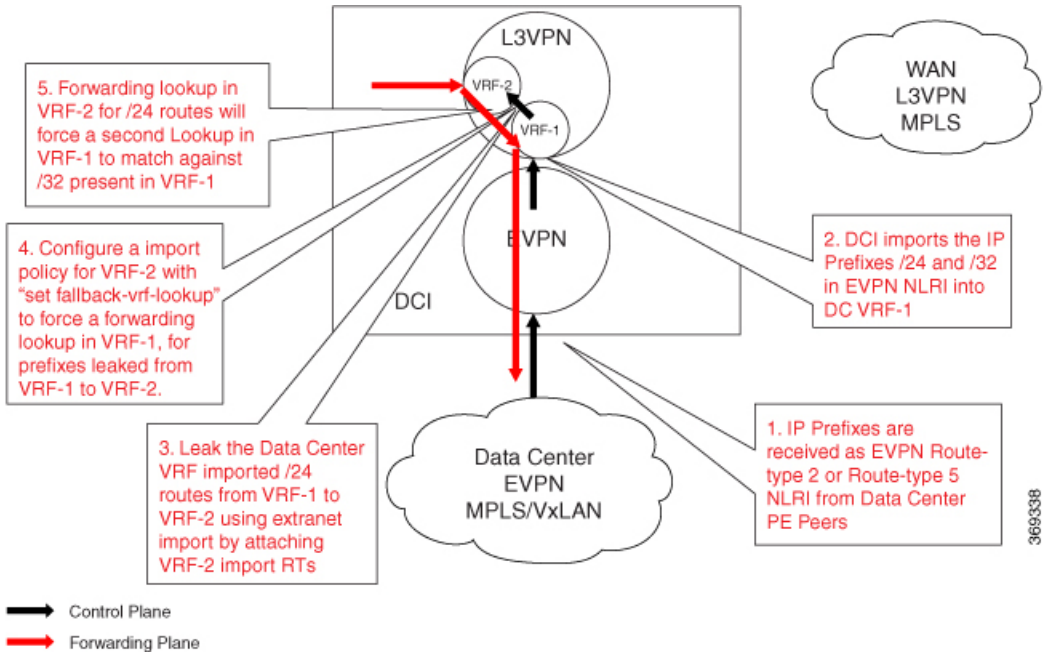
Use the **set fallback-vrf-lookup** command to enable this functionality in the following route-policy attach points:

- VRF import
- VRF import from default-vrf
- VRF export to default-vrf

The route-policy is executed when the prefix is imported to the destination VRF. Use the **set fallback-vrf-lookup** command to select the leaked prefixes that are programmed to force a second look up in the source VRF.

Configuration Example

Figure 27: Force Lookup in Source VRF



The example in the earlier section *Advertise Leaked Prefix* explains how L3VPN and EVPN prefixes are imported to VRF-1, leaked to VRF-2, and advertised to L3VPN and EVPN peers.

Without configuring the Lookup in Source VRF functionality, when traffic is received for the advertised prefix in VRF-2, a lookup is done for the prefix in VRF-2 and traffic is forwarded towards the prefix next hop.

When Lookup in Source VRF functionality is configured for L3VPN and EVPN, prefixes are imported to VRF-1 and leaked to VRF-2. And, when the **set fallback-vrf-lookup** command is used, the prefixes are advertised to L3VPN and EVPN peers. When traffic is received from the opposite direction in VRF-2, first a lookup is performed in VRF-2, which forces a second lookup in VRF-1. To force this second lookup in VRF-1, configure the following import route-policy in VRF-2.

```
RP/0/0/CPU0:router#(config)#route-policy vrf-2-import-policy
RP/0/0/CPU0:router#(config-rpl)#if destination in (203.0.113.0/24) then
RP/0/0/CPU0:router#(config-rpl-if)#set fallback-vrf-lookup
RP/0/0/CPU0:router#(config-rpl-if)#endif
RP/0/0/CPU0:router#(config-rpl)#end-policy
RP/0/0/CPU0:router#(config)#route-policy vrf-2-import-policy-v6
RP/0/0/CPU0:router#(config-rpl)#if destination in (2001:DB8::/32) then
RP/0/0/CPU0:router#(config-rpl-if)#set fallback-vrf-lookup
RP/0/0/CPU0:router#(config-rpl-if)#endif
RP/0/0/CPU0:router#(config-rpl)#end-policy
RP/0/0/CPU0:router#(config)#vrf VRF-2
RP/0/0/CPU0:router#(config-vrf)#address-family ipv4 unicast
RP/0/0/CPU0:router#(config-vrf-af)#import route-policy vrf-2-import-policy
RP/0/0/CPU0:router#(config-vrf-af)#import from vrf advertise-as-vpn
RP/0/0/CPU0:router#(config-vrf-af)#import route-target
RP/0/0/CPU0:router#(config-vrf-import-rt)#2:1
RP/0/0/CPU0:router#(config-vrf-import-rt)#200:1 stitching
RP/0/0/CPU0:router#(config-vrf-import-rt)#300:1 stitching
!
RP/0/0/CPU0:router#(config-vrf-import-rt)#export route-target
RP/0/0/CPU0:router#(config-vrf-export-rt)#2:1
RP/0/0/CPU0:router#(config-vrf-export-rt)#200:1 stitching
!
RP/0/0/CPU0:router#(config-vrf-export-rt)#address-family ipv6 unicast
RP/0/0/CPU0:router#(config-vrf-af)#import route-policy vrf-2-import-policy-v6
RP/0/0/CPU0:router#(config-vrf-af)#import from vrf advertise-as-vpn
RP/0/0/CPU0:router#(config-vrf-af)#import route-target
RP/0/0/CPU0:router#(config-vrf-import-rt)#2:1
RP/0/0/CPU0:router#(config-vrf-import-rt)#200:1 stitching
RP/0/0/CPU0:router#(config-vrf-import-rt)#300:1 stitching
!
RP/0/0/CPU0:router#(config-vrf-import-rt)#export route-target
RP/0/0/CPU0:router#(config-vrf-export-rt)#2:1
RP/0/0/CPU0:router#(config-vrf-export-rt)#200:1 stitching
RP/0/0/CPU0:router#(config-vrf-export-rt)#commit
```

Running Configuration

This section shows the running configuration.

```
route-policy vrf-2-import-policy
  if destination in (203.0.113.0/24) then
    set fallback-vrf-lookup
  endif
end-policy

route-policy vrf-2-import-policy-v6
  if destination in (2001:DB8::/32) then
    set fallback-vrf-lookup
  endif
```

```
end-policy


vrf VRF-2
address-family ipv4 unicast
  import route-policy vrf-2-import-policy
  import from vrf advertise-as-vpn
  import route-target
   2:1
   200:1 stitching
   300:1 stitching
  !
  export route-target
   2:1
   200:1 stitching
  !
!
address-family ipv6 unicast
  import route-policy vrf-2-import-policy-v6
  import from vrf advertise-as-vpn
  import route-target
   2:1
   200:1 stitching
   300:1 stitching
  !
  export route-target
   2:1
   200:1 stitching
  !
```

Enable Services using EVPN VxLAN VRF Route Leaking

Service providers can use the Cisco VxLAN solution to provide the following additional services to its customers:

- Internet access
- Service VRF
- Internet full feed to customer edge devices from DCI
- Inter-VRF routing

Internet Access

The Internet access provides Internet access to tenant hosts in the data center.

For more information, see the *EVPN Default VRF Route Leaking on the DCI for Internet Connectivity* section.

Service VRF

The Service VRF enables connectivity to the services in the Service VRF to customers in the EVPN data center VRF.

For more information, see the *EVPN Service VRF Route Leaking* section.

Internet Full Feed to Customer Edge Devices from DCI

The Internet Full Feed to Customer Edge Devices from DCI service downloads the Internet prefixes on the DCI default VRF to customer edge (CE) device and provides Internet access to CE network hosts through the data center fabric.

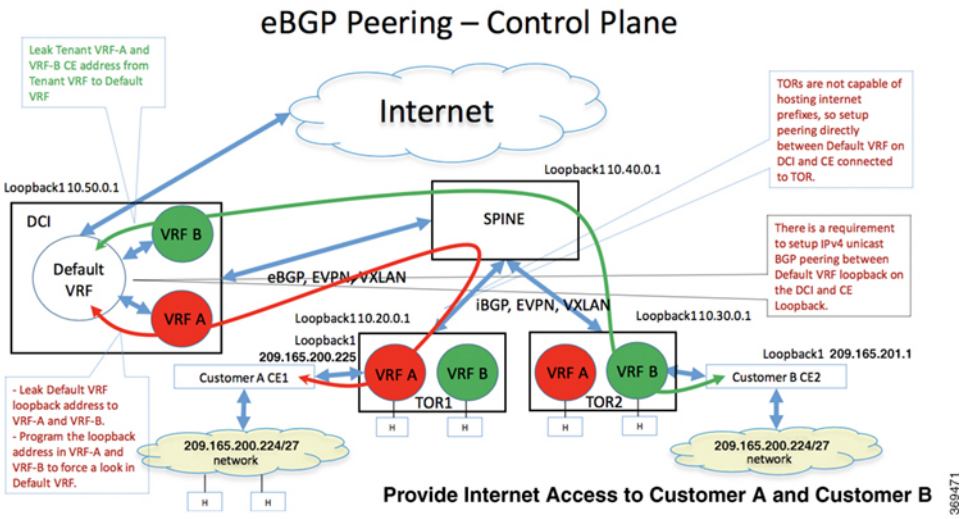
To download Internet prefixes to CE router and to provide Internet access to CE network hosts through the data center fabric, perform the followin tasks:

- Set up reachability between DCI default VRF loopback address and CE loopback address.
- Configure eBGP session between default VRF loopback address on the DCI and CE loopback address.
- Enable exchange of routes between DCI default VRF and CE.
- Provide Internet connectivity to CE network hosts through the data center fabric.

Setup the Reachability between DCI Default VRF Loopback Address and CE Loopback Address

This example shows the configuration required to set up the reachability between default VRF loopback1 address 10.50.0.1 on the DCI, and CE1 loopback1 address 209.165.200.225 connected to VRF-A on the TOR1.

Figure 28: Setup the Reachability between DCI Default VRF Loopback Address and CE Loopback Address



Propagate Reachability of DCI Default VRF Loopback Address to CE1

This section explains the configuration required to propagate the reachability of DCI default VRF loopback address to CE1 router.

DCI Configuration

- Redistribute the default VRF Looback1 address 10.50.0.1 into BGP default VRF.
- Leak the default VRF loopback address 10.50.0.1 to VRF-A, and configure the loopback address in VRF-A to force a fallback VRF lookup in the default VRF.
- Advertise DCI default VRF loopback address 10.50.0.1 in VRF-A towards TOR1 through the SPINE.

```
RP/0/0/CPU0:router(config)# interface Loopback1
RP/0/0/CPU0:router(config-if)#ipv4 address 10.50.0.1 255.255.255.255
RP/0/0/CPU0:router(config-if)# exit
RP/0/0/CPU0:router(config)# vrf VRF-A
RP/0/0/CPU0:router(config-vrf)# address-family ipv4 unicast
RP/0/0/CPU0:router(config-vrf-af)# import from default-vrf route-policy
vrf-a-default-vrf-import-policy advertise-as-vpn
RP/0/0/CPU0:router(config-vrf-af)# import route-target
RP/0/0/CPU0:router(config-vrf-import-rt)# 1:1
RP/0/0/CPU0:router(config-vrf-import-rt)# 100:1 stitching
!
RP/0/0/CPU0:router(config-vrf-af)# export to default-vrf route-policy
vrf-a-default-vrf-export-policy allow-imported-vpn
RP/0/0/CPU0:router(config-vrf-af)# export route-target
RP/0/0/CPU0:router(config-vrf-export-rt)# 1:1
RP/0/0/CPU0:router(config-vrf-export-rt)# 100:1 stitching
!
RP/0/0/CPU0:router(config)# route-policy vrf-a-default-vrf-import-policy
RP/0/0/CPU0:router(config-rpl)# if destination in (10.50.0.1/32, 0.0.0.0/0) then ← DCI
Default VRF loopback is leaked to VRF-A
RP/0/0/CPU0:router(config-rpl-if)# set fallback-vrf-lookup ← Look up in VRF-A forces a
second look up in Default VRF
RP/0/0/CPU0:router(config-rpl-if)# pass
RP/0/0/CPU0:router(config-rpl-if)# endif
RP/0/0/CPU0:router(config-rpl)# end-policy
RP/0/0/CPU0:router(config)# router bgp 300
RP/0/0/CPU0:router(config-bgp)# bgp router-id 172.16.0.1
!
RP/0/0/CPU0:router(config-bgp)# address-family ipv4 unicast
RP/0/0/CPU0:router(config-bgp-af)# redistribute connected
!
RP/0/0/CPU0:router(config-bgp-af)# neighbor 10.40.0.1 ← SPINE
RP/0/0/CPU0:router(config-bgp-nbr)# remote-as 100
RP/0/0/CPU0:router(config-bgp-nbr)# ebgp-multihop 4
RP/0/0/CPU0:router(config-bgp-nbr)# update-source Loopback1
RP/0/0/CPU0:router(config-bgp-nbr)# address-family l2vpn evpn
RP/0/0/CPU0:router(config-bgp-nbr-af)# import stitching-rt re-originate
RP/0/0/CPU0:router(config-bgp-nbr-af)# route-policy pass in
RP/0/0/CPU0:router(config-bgp-nbr-af)# encapsulation-type vxlan
RP/0/0/CPU0:router(config-bgp-nbr-af)# route-policy pass out
RP/0/0/CPU0:router(config-bgp-nbr-af)# advertise vpnv4 unicast re-originated stitching-rt
RP/0/0/CPU0:router(config-bgp-nbr-af)# advertise vpnv6 unicast re-originated stitching-rt
!
RP/0/0/CPU0:router(config-bgp-nbr-af)# vrf VRF-A
RP/0/0/CPU0:router(config-bgp-vrf)# rd auto
RP/0/0/CPU0:router(config-bgp-vrf)# address-family ipv4 unicast
RP/0/0/CPU0:router(config-bgp-vrf-af)# !
```

Note Configure the default VRF loopback address 10.50.0.1 using **set fallback-vrf-lookup** command while importing to VRF-A. With this configuration, a forwarding lookup for default VRF loopback address 10.50.0.1 in VRF-A forces a second forwarding lookup in default VRF. This results in the TCP and BGP packets to hit the default VRF LPTS entries to be steered towards the TCP and BGP process in the default VRF context.

Running Configuration

This section shows the DCI running configuration.

```
interface Loopback1
ipv4 address 10.50.0.1 255.255.255.255
exit
vrf VRF-A
address-family ipv4 unicast
  import from default-vrf route-policy vrf-a-default-vrf-import-policy advertise-as-vpn
  import route-target
   1:1
   100:1 stitching
  !
  export to default-vrf route-policy vrf-a-default-vrf-export-policy allow-imported-vpn
  export route-target
   1:1
   100:1 stitching
!
route-policy vrf-a-default-vrf-import-policy
  if destination in (10.50.0.1/32, 0.0.0.0/0) then ← DCI Default VRF loopback is leaked to
 VRF-A
    set fallback-vrf-lookup ← Look up in VRF-A forces a second look up in default VRF
    pass
  endif
end-policy

router bgp 300
bgp router-id 172.16.0.1
!
address-family ipv4 unicast
   redistribute connected
!
neighbor 10.40.0.1 ← SPINE
  remote-as 100
  ebgp-multihop 4
  update-source Loopback1
  address-family l2vpn evpn
   import stitching-rt re-originate
   route-policy pass in
   encapsulation-type vxlan
   route-policy pass out
   advertise vpnv4 unicast re-originated stitching-rt
   advertise vpnv6 unicast re-originated stitching-rt
  !
!
vrf VRF-A
  rd auto
  address-family ipv4 unicast
  !
!
```

SPINE Configuration

The SPINE acts as a route reflector and reflects the EVPN routes between DCI and TORs.

```
RP/0/0/CPU0:router(config)# interface Loopback1
RP/0/0/CPU0:router(config-if)# ipv4 address 10.40.0.1 255.255.255.255
!
RP/0/0/CPU0:router(config)# router bgp 100
RP/0/0/CPU0:router(config-bgp)# bgp router-id 192.168.0.2
!
RP/0/0/CPU0:router(config-bgp)# address-family l2vpn evpn
RP/0/0/CPU0:router(config-bgp-af)# neighbor 10.50.0.1 ← DCI
RP/0/0/CPU0:router(config-bgp-nbr)# remote-as 300
RP/0/0/CPU0:router(config-bgp-nbr)# ebgp-multihop 4
```

```
RP/0/0/CPU0:router(config-bgp-nbr)# update-source Loopback1
RP/0/0/CPU0:router(config-bgp-nbr)# address-family l2vpn evpn
RP/0/0/CPU0:router(config-bgp-nbr-af)# route-reflector-client
RP/0/0/CPU0:router(config-bgp-nbr-af)# encapsulation-type vxlan
RP/0/0/CPU0:router(config-bgp-nbr-af)# route-policy pass in
RP/0/0/CPU0:router(config-bgp-nbr-af)# route-policy pass out
RP/0/0/CPU0:router(config-bgp-nbr-af)# next-hop-unchanged
!
RP/0/0/CPU0:router(config-bgp-af)# neighbor 10.20.0.1 ← TOR1
RP/0/0/CPU0:router(config-bgp-nbr)# remote-as 100
RP/0/0/CPU0:router(config-bgp-nbr)# update-source Loopback1
RP/0/0/CPU0:router(config-bgp-nbr)# address-family l2vpn evpn
RP/0/0/CPU0:router(config-bgp-nbr-af)# route-reflector-client
RP/0/0/CPU0:router(config-bgp-nbr-af)# encapsulation-type vxlan
!
RP/0/0/CPU0:router(config-bgp-af)# neighbor 10.30.0.1 ← TOR2
RP/0/0/CPU0:router(config-bgp-nbr)# remote-as 100
RP/0/0/CPU0:router(config-bgp-nbr)# update-source Loopback1
RP/0/0/CPU0:router(config-bgp-nbr)# address-family l2vpn evpn
RP/0/0/CPU0:router(config-bgp-nbr-af)# route-reflector-client
RP/0/0/CPU0:router(config-bgp-nbr-af)# encapsulation-type vxlan
```

Running Configuration

This section shows the SPINE running configuration.

```
interface Loopback1
ipv4 address 10.40.0.1 255.255.255.255
!
router bgp 100
bgp router-id 192.168.0.2
!
address-family l2vpn evpn
!
neighbor 10.50.0.1 ← DCI
  remote-as 300
  ebgp-multihop 4
  update-source Loopback1
  address-family l2vpn evpn
   route-reflector-client
   encapsulation-type vxlan
   route-policy pass in
   route-policy pass out
   next-hop-unchanged
  !
!
neighbor 10.20.0.1 ← TOR1
  remote-as 100
  update-source Loopback1
  address-family l2vpn evpn
   route-reflector-client
   encapsulation-type vxlan
  !
!
neighbor 10.30.0.1 ← TOR2
  remote-as 100
  update-source Loopback1
  address-family l2vpn evpn
   route-reflector-client
   encapsulation-type vxlan
  !
```

```
!
!
```

TOR1 Configuration

In this example, advertise DCI default VRF loopback address 10.50.0.1 in VRF-A to CE1 through the eBGP session between TOR1 and CE1, with TOR1 VRF-A loopback address as 10.20.0.2 as the next hop.

```
RP/0/0/CPU0:router(config)# interface Loopback2
RP/0/0/CPU0:router(config-if)# vrf foo
RP/0/0/CPU0:router(config-if)# ipv4 address 10.20.0.2 255.255.255.255
RP/0/0/CPU0:router(config-if)# router bgp 100
RP/0/0/CPU0:router(config-bgp)# bgp router-id 172.16.0.2
!
RP/0/0/CPU0:router(config-bgp)# address-family vpnv4 unicast
!
RP/0/0/CPU0:router(config-bgp)# address-family vpnv6 unicast
!
RP/0/0/CPU0:router(config-bgp)# address-family l2vpn evpn
RP/0/0/CPU0:router(config-bgp-af)# neighbor 10.40.0.1 ← SPINE
RP/0/0/CPU0:router(config-bgp-nbr)# remote-as 100
RP/0/0/CPU0:router(config-bgp-nbr)# update-source Loopback1
RP/0/0/CPU0:router(config-bgp-nbr)# address-family l2vpn evpn
RP/0/0/CPU0:router(config-bgp-nbr-af)# encapsulation-type vxlan
RP/0/0/CPU0:router(config-bgp-nbr-af)# advertise vpnv4 unicast re-originated
RP/0/0/CPU0:router(config-bgp-nbr-af)# advertise vpnv6 unicast re-originated
!
RP/0/0/CPU0:router(config-bgp-nbr-af)# vrf VRF-A
RP/0/0/CPU0:router(config-bgp-vrf)# rd auto
RP/0/0/CPU0:router(config-bgp-vrf)# address-family ipv4 unicast
RP/0/0/CPU0:router(config-bgp-vrf-af)# redistribute connected
!
RP/0/0/CPU0:router(config-bgp-vrf)# neighbor 209.165.200.225 ← CE1
RP/0/0/CPU0:router(config-bgp-vrf-nbr)# remote-as 500
RP/0/0/CPU0:router(config-bgp-vrf-nbr)# ebgp-multihop 4
RP/0/0/CPU0:router(config-bgp-vrf-nbr)# update-source Loopback2
RP/0/0/CPU0:router(config-bgp-vrf-nbr)# address-family ipv4 unicast
RP/0/0/CPU0:router(config-bgp-vrf-nbr-af)# route-policy pass-all in
RP/0/0/CPU0:router(config-bgp-vrf-nbr-af)# route-policy pass-all out
RP/0/0/CPU0:router(config-bgp-vrf-nbr-af)# commit
```

Running Configuration

This section shows the TOR1 running configuration.

```
interface Loopback2
vrf foo
ipv4 address 10.20.0.2 255.255.255.255

router bgp 100
bgp router-id 172.16.0.2
!
address-family vpnv4 unicast
!
address-family vpnv6 unicast
!
address-family l2vpn evpn
!
neighbor 10.40.0.1 ← SPINE
  remote-as 100
  update-source Loopback1
```

```
  address-family l2vpn evpn
   encapsulation-type vxlan
   advertise vpnv4 unicast re-originated
   advertise vpnv6 unicast re-originated
  !
!
vrf VRF-A
  rd auto
  address-family ipv4 unicast
   redistribute connected
  !
  neighbor 209.165.200.225 ← CE1
   remote-as 500
   ebgp-multihop 4
   update-source Loopback2
   address-family ipv4 unicast
    route-policy pass-all in
    route-policy pass-all out
   !
  !
```

CE1 Configuration

In this example, the DCI default VRF loopback address 10.50.0.1 is received by CE1 through the eBGP session between CE1 and TOR1.

```
RP/0/0/CPU0:router(config)# interface Loopback1
RP/0/0/CPU0:router(config-if)# ipv4 address 209.165.200.225 255.255.255.255
!
RP/0/0/CPU0:router(config)# router bgp 500
RP/0/0/CPU0:router(config-bgp)# bgp router-id 209.165.200.225
RP/0/0/CPU0:router(config-bgp)# address-family ipv4 unicast
RP/0/0/CPU0:router(config-bgp-af)# redistribute connected
!
RP/0/0/CPU0:router(config-bgp)# neighbor 10.20.0.2 ← TOR1
RP/0/0/CPU0:router(config-bgp-nbr)# remote-as 100
RP/0/0/CPU0:router(config-bgp-nbr)# ebgp-multihop 4
RP/0/0/CPU0:router(config-bgp-nbr)# update-source Loopback1
RP/0/0/CPU0:router(config-bgp-nbr)# address-family ipv4 unicast
RP/0/0/CPU0:router(config-bgp-nbr-af)# route-policy pass in
RP/0/0/CPU0:router(config-bgp-nbr-af)# route-policy pass out
RP/0/0/CPU0:router(config-bgp-nbr-af)# commit
```

Running Configuration

This section shows the CE1 running configuration.

```
interface Loopback1
ipv4 address 209.165.200.225 255.255.255.255
!
router bgp 500
bgp router-id 209.165.200.225
address-family ipv4 unicast
  redistribute connected
!
!
neighbor 10.20.0.2 ← TOR1
  remote-as 100
  ebgp-multihop 4
  update-source Loopback1
  address-family ipv4 unicast
```

```
   route-policy pass in
   route-policy pass out
  !
!
!
```

Propagate CE1 Loopback Address Reachability to DCI Default VRF

This section explains the configuration required to propagate the reachability of CE1 loopback address to DCI default VRF.

CE1 Configuration

Redistribute the CE1 loopback address 209.165.200.225 into BGP. Advertise the CE1 loopback address 209.165.200.225 to TOR1 VRF-A through the eBGP session between CE1 and TOR1.

TOR1 Configuration

Advertise the CE1 loopback address 209.165.200.225 in VRF-A towards DCI.

DCI Configuration

```
RP/0/0/CPU0:router(config)# vrf VRF-A
RP/0/0/CPU0:router(config-vrf)# address-family ipv4 unicast
RP/0/0/CPU0:router(config-vrf-af)# import from default-vrf route-policy
vrf-a-default-vrf-import-policy advertise-as-vpn
RP/0/0/CPU0:router(config-vrf-af)# import route-target
RP/0/0/CPU0:router(config-vrf-import-rt)# 1:1
RP/0/0/CPU0:router(config-vrf-import-rt)# 100:1 stitching
!
RP/0/0/CPU0:router(config-vrf-import)# export to default-vrf route-policy
vrf-a-default-vrf-export-policy allow-imported-vpn
RP/0/0/CPU0:router(config-vrf-af)# export route-target
RP/0/0/CPU0:router(config-vrf-export-rt)# 1:1
RP/0/0/CPU0:router(config-vrf-export-rt)# 100:1 stitching
!
RP/0/0/CPU0:router(config-vrf-export-rt)# route-policy vrf-a-default-vrf-export-policy
RP/0/0/CPU0:router(config-rpl)# if destination in (209.165.200.225/32) then  ← CE1 Loopback1
 is leaked from VRF-A to Default VRF
RP/0/0/CPU0:router(config-rpl-if)  pass
RP/0/0/CPU0:router(config-rpl-if)# endif
RP/0/0/CPU0:router(config-rpl)# end-policy
```

Note Advertise only the CE1 subnet address 209.165.200.224/27 to the Internet. Do not advertise the CE1 Loopback1 address 209.165.200.225/32 to the Internet.

Running Configuration

This section shows the DCI running configuration.

```
vrf VRF-A
address-family ipv4 unicast
  import from default-vrf route-policy vrf-a-default-vrf-import-policy advertise-as-vpn
  import route-target
   1:1
   100:1 stitching
```

```
  !
  export to default-vrf route-policy vrf-a-default-vrf-export-policy allow-imported-vpn
  export route-target
   1:1
   100:1 stitching
  !
!

route-policy vrf-a-default-vrf-export-policy
  if destination in (209.165.200.225/32) then  ← CE1 Loopback1 is leaked from VRF-A to
Default VRF
    pass
  endif
end-policy
```

Configure eBGP Session between DCI Default VRF and CE

This configuration brings up eBGP IPv4 unicast session between DCI default VRF loopback address and CE1 loopback address.

DCI Configuration

Configure the eBGP neighbor configuration with CE1 loopback address 209.165.200.225.

```
RP/0/0/CPU0:router(config)# router bgp 300
RP/0/0/CPU0:router(config-bgp)# bgp router-id 172.16.0.1
!
RP/0/0/CPU0:router(config-bgp)# address-family ipv4 unicast
!
RP/0/0/CPU0:router(config-bgp)# neighbor 209.165.200.225 ←CE1
RP/0/0/CPU0:router(config-bgp-nbr)# remote-as 500
RP/0/0/CPU0:router(config-bgp-nbr)# ebgp-multihop 4
RP/0/0/CPU0:router(config-bgp-nbr)# update-source Loopback1
RP/0/0/CPU0:router(config-bgp-nbr)# address-family ipv4 unicast
RP/0/0/CPU0:router(config-bgp-nbr-af)# route-policy pass in
RP/0/0/CPU0:router(config-bgp-nbr-af)# route-policy pass out
RP/0/0/CPU0:router(config-bgp-nbr-af)# commit
```

Running Configuration

This section shows the DCI running configuration.

```
router bgp 300
bgp router-id 172.16.0.1
!
address-family ipv4 unicast
!
neighbor 209.165.200.225  ←CE1
  remote-as 500
  ebgp-multihop 4
  update-source Loopback1
  address-family ipv4 unicast
   route-policy pass in
   route-policy pass out
  !
!
!
```

CE1 Configuration

Configure the eBGP neighbor configuration with DCI default VRF loopback address 10.50.0.1.

```
RP/0/0/CPU0:router(config)# router bgp 500
RP/0/0/CPU0:router(config-bgp)#bgp router-id 209.165.200.225
RP/0/0/CPU0:router(config-bgp)# address-family ipv4 unicast
RP/0/0/CPU0:router(config-bgp-af)# redistribute connected
RP/0/0/CPU0:router(config-bgp-af)# neighbor 10.50.0.1 ←DCI
RP/0/0/CPU0:router(config-bgp-nbr)#remote-as 300
RP/0/0/CPU0:router(config-bgp-nbr)# ebgp-multihop 4
RP/0/0/CPU0:router(config-bgp-nbr)# update-source Loopback1
RP/0/0/CPU0:router(config-bgp-nbr)# address-family ipv4 unicast
RP/0/0/CPU0:router(config-bgp-nbr-af)# route-policy pass in
RP/0/0/CPU0:router(config-bgp-nbr-af)# route-policy pass out
RP/0/0/CPU0:router(config-bgp-nbr-af)!!
```

Running Configuration

This section shows the CE1 running configuration.

```
router bgp 500
bgp router-id 209.165.200.225
address-family ipv4 unicast
  redistribute connected
!
!
neighbor 10.50.0.1 ←DCI
  remote-as 300
  ebgp-multihop 4
  update-source Loopback1
  address-family ipv4 unicast
   route-policy pass in
   route-policy pass out
  !
!
!
```

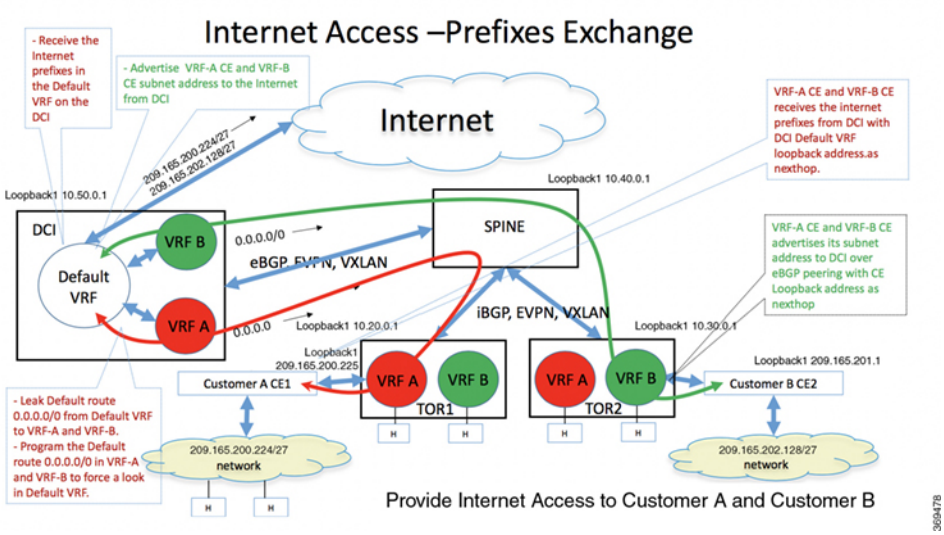
Exchange Prefixes between DCI Default VRF and CE1

You must establish an eBGP session between DCI default VRF loopback address and CE1 loopback address to enable the exchange of IPv4 unicast routes between DCI default VRF and CE1 to advertise:

- Internet prefixes on DCI default VRF to CE1 with DCI default VRF loopback address 10.50.0.1 as the next hop.
- CE1 prefix 209.165.200.224/27 to DCI default VRF with CE1 loopback address 209.165.200.225 as the next hop.

Note Advertise the same prefix 209.165.200.224/27 from CE1 into the data center VRF on TOR to enable forwarding of Internet traffic destined to CE in the data center fabric.

Figure 29: Internet Access - Prefixes Exchange



The following sections explain how to exchange prefixes between DCI default VRF and CE1:

Advertise Internet Prefixes from DCI to CE1

This section explains how to advertise Internet prefixes to CE1.

DCI

1. DCI receives the Internet prefixes in the default VRF.
2. Advertise these prefixes to CE1 with DCI default VRF loopback address 1050.0.1 as the next hop.
3. Configure the default route 0.0.0.0/0 in VRF-A to force a second look up in default VRF.
4. Advertise the default route (or configure default-originate) in VRF-A towards TOR1.

The default route in VRF-A is required to forward traffic destined to the Internet from CE1 network hosts through the data center fabric.

TOR1

TOR1 VRF-A has default route 0.0.0.0/0 with DCI default VRF loopback address 10.50.0.1 as the next hop.

Note Configure an outbound route-policy session with CE1 to block the advertisement of default route 0.0.0.0/0 to CE1.

CE1

CE1 receives Internet prefixes with DCI default VRF loopback address 10.50.0.1 as the next hop.

Note Do not advertise the Internet prefixes received by CE1 from DCI eBGP session to TOR1 eBGP session. Similarly, do not advertise the VRF-A routes received by CE1 from TOR1 eBGP session to DCI eBGP session.

Advertise CE1 Prefixes to DCI Default VRF

This section explains how to advertise CE1 subnet prefixes to the Internet.

CE1

From CE1, advertise CE1 subnet address 209.165.200.224/27 to DCI over eBGP peering with CE1 loopback address 209.165.200.225 as the next hop. Also, advertise the same CE1 subnet address 209.165.200.224/27 to VRF-A TOR1 over eBGP peering between CE1 and TOR1. This is required for forwarding traffic coming from the Internet to CE network hosts through the data center fabric.

TOR1

From TOR1, advertise the CE1 subnet address 209.165.200.224/27 to DCI VRF-A with TOR1 loopback address as the next hop.

DCI

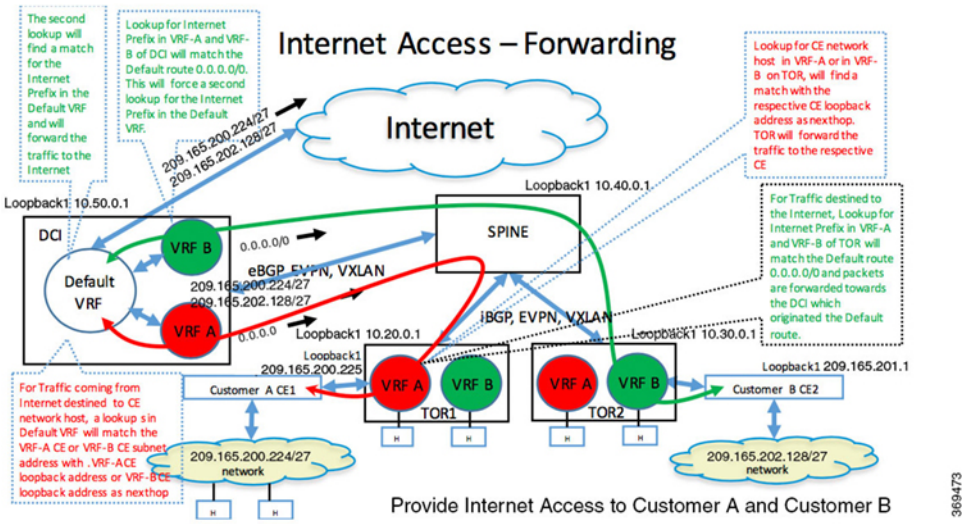
From DCI, advertise the CE1 subnet address 209.165.200.224/27 to the Internet with itself (DCI) as the next hop.

Note Do not advertise the /32 prefixes (that includes the CE loopback address 192.168.0.1/32 and DCI loopback address 10.50.0.1/32) in the default VRF to the Internet. Also, do not advertise these prefixes to CE1 through the eBGP session to CE1.

Forwarding Traffic to and from Internet in the Data Center Fabric

This section explains how forwarding works when CE1 network hosts access the Internet.

Figure 30: Forwarding Traffic to and from Internet in the Data Center Fabric



Forwarding CE1 traffic destined to Internet

This section explains how forwarding works for traffic destined to Internet from CE network hosts.

CE1

For traffic destined to Internet, a lookup for Internet prefix on CE1 finds a match with default VRF loopback address 10.50.0.1 as next hop. The next hop 10.50.0.1 is reachable through TOR1 VRF-A loopback address 10.20.0.2 forwards the traffic to TOR1 VRF-A.

TOR1

For traffic destined to the Internet from CE1, a lookup for Internet prefix in VRF-A on TOR1 matches the default route 0.0.0.0/0 with next hop pointing to the DCI, and forwards the traffic towards the DCI in VRF-A.

DCI

Lookup for Internet prefix in VRF-A matches the default route 0.0.0.0/0. The default route 0.0.0.0/0 is programmed to force a second look up in the default VRF. A second lookup in default VRF for Internet Prefix finds a match, and forwards the traffic to the Internet.

Forwarding Internet traffic destined to CE1

This section explains how forwarding works for traffic coming from Internet towards CE network hosts.

DCI

For traffic coming from Internet and destined to a CE1 network host, a lookup for CE1 network host in default VRF matches the CE1's subnet address with the CE1's loopback address as the next hop. The traffic is forwarded towards TOR1 in VRF-A.

TOR1

Lookup for CE1 network host in VRF-A matches the CE1 subnet address with CE1 loopback address as the next hop. The traffic is forwarded to CE1.

CE1

Lookup for CE1 network host on CE1 finds a match for the host route.

Inter-VRF Routing

The Inter-VRF Routing service provides connectivity between hosts in multiple data center VRFs.

You must import the routes of each VRF into the other VRF to enable Inter-VRF routing between the two VRFs. Configure the export RT in each VRF with the import RT of both VRFs or configure the import RT in each VRF with the export RTs of both VRFs. This configuration is required on all routers that have these VRFs configured. By default, it imports all the VRFs routes to the other VRF.

Alternatively, leak the routes of each VRF into the other VRF and advertise the leaked routes with the new VRFs context, such as RD and export RT to establish Inter-VRF routing between the two VRFs. This minimizes the configuration required for Inter-VRF routing to only one central router. In the data center, the DCI provides a gateway functionality through a central router. You can configure Inter-VRF routing on this central router using route leaking.

The following section explains how to enable Inter-VRF routing between the two VRFs:

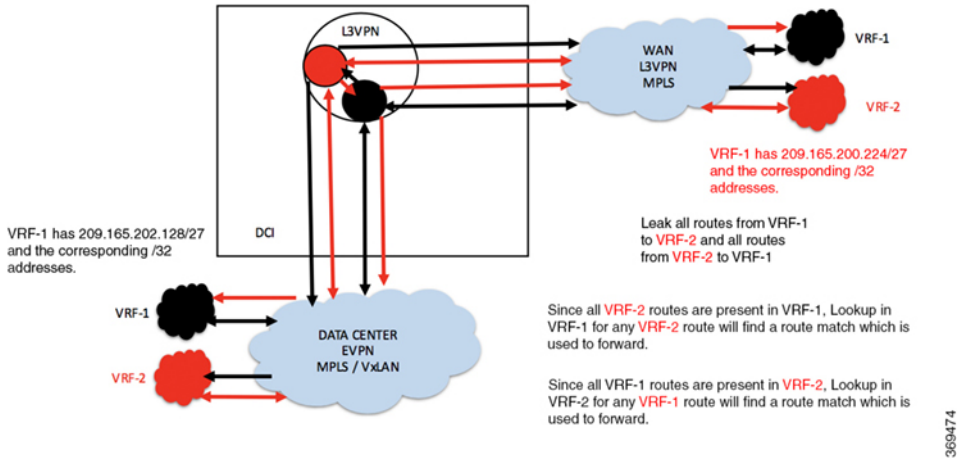
Inter-VRF Routing Between Small Number of VRFs

Leak All Routes

In this example, VRF-1 leaks all routes to VRF-2. VRF-2 leaks all routes to VRF-1. Both VRFs advertise the leaked routes in both VRFs with the context such as, RD and export RT of the leaked VRF.

Both VRFs contain the routes of both of them, so there is a reachability between hosts in both the VRFs.

Figure 31: Inter-VRF Routing - Leak All Routes



Configuration Example

```
RP/0/0/CPU0:router(config)# route-policy vrf-leak-from-vrf-1-to-vrf-2
RP/0/0/CPU0:router(config-rpl)# set extcommunity rt (400:1) additive
RP/0/0/CPU0:router(config-rpl)# pass
RP/0/0/CPU0:router(config-rpl)# end-policy
!
RP/0/0/CPU0:router(config)# route-policy vrf-leak-from-vrf-2-to-vrf-1
RP/0/0/CPU0:router(config-rpl)# set extcommunity rt (300:1) additive
RP/0/0/CPU0:router(config-rpl)# pass
RP/0/0/CPU0:router(config-rpl)# end-policy
!
RP/0/0/CPU0:router(config)# vrf VRF-1
RP/0/0/CPU0:router(config-vrf)# address-family ipv4 unicast
RP/0/0/CPU0:router(config-vrf-af)# import from vrf advertise-as-vpn
RP/0/0/CPU0:router(config-vrf-af)# import route-target
RP/0/0/CPU0:router(config-vrf-import-rt)# 1:1
RP/0/0/CPU0:router(config-vrf-import-rt)# 300:1
RP/0/0/CPU0:router(config-vrf-import-rt)# 100:1 stitching
RP/0/0/CPU0:router(config-vrf-import-rt)# 300:1 stitching
!
RP/0/0/CPU0:router(config-vrf-af)# export route-policy vrf-leak-from-vrf-1-to-vrf-2
RP/0/0/CPU0:router(config-vrf-af)# export to vrf allow-imported-vpn
RP/0/0/CPU0:router(config-vrf-af)# export route-target
RP/0/0/CPU0:router(config-vrf-export-rt)# 1:1
RP/0/0/CPU0:router(config-vrf-export-rt)# 100:1 stitching
!
```

Note: The import route target 400:1 is configured only on the DCI for leaking. Do not configure this route target 400:1 on any other router in the network.
We recommend that you do not use the same import route target, in this example, 400:1 to prevent unintended import in the network.

```
RP/0/0/CPU0:router(config)# vrf VRF-2
RP/0/0/CPU0:router(config-vrf)# address-family ipv4 unicast
RP/0/0/CPU0:router(config-vrf-af)#  import from vrf advertise-as-vpn
RP/0/0/CPU0:router(config-vrf-af)# import route-target
RP/0/0/CPU0:router(config-vrf-import-rt)# 2:1
RP/0/0/CPU0:router(config-vrf-import-rt)# 400:1
RP/0/0/CPU0:router(config-vrf-import-rt)# 200:1 stitching
```

```
RP/0/0/CPU0:router(config-vrf-import-rt)# 400:1 stitching
!
RP/0/0/CPU0:router(config-vrf-af)# export route-policy vrf-leak-from-vrf-2-to-vrf-1
RP/0/0/CPU0:router(config-vrf-af)# export to vrf allow-imported-vpn
RP/0/0/CPU0:router(config-vrf-af)# export route-target
RP/0/0/CPU0:router(config-vrf-export-rt)# 2:1
RP/0/0/CPU0:router(config-vrf-export-rt)# 200:1 stitching
!

Note: The import route target 300:1 is configured only on the DCI for leaking. Do not
configure this route target 300:1 on any other router in the network.
We recommend that you do not use the same import route target, in this example, 300:1 to
prevent unintended import in the network.
```

Running Configuration

This section shows running configuration.

```
route-policy vrf-leak-from-vrf-1-to-vrf-2
  set extcommunity rt (400:1) additive
  pass
end-policy
!

route-policy vrf-leak-from-vrf-2-to-vrf-1
  set extcommunity rt (300:1) additive
  pass
end-policy
!

vrf VRF-1
address-family ipv4 unicast
  import from vrf advertise-as-vpn
  import route-target
   1:1
   300:1
   100:1 stitching
   300:1 stitching
  !
  export route-policy vrf-leak-from-vrf-1-to-vrf-2
  export to vrf allow-imported-vpn
  export route-target
   1:1
   100:1 stitching
  !
!

vrf VRF-2
address-family ipv4 unicast
  import from vrf advertise-as-vpn
  import route-target
   2:1
   400:1
   200:1 stitching
   400:1 stitching
  !
  export route-policy vrf-leak-from-vrf-2-to-vrf-1
  export to vrf allow-imported-vpn
  export route-target
   2:1
   200:1 stitching
  !
!
```

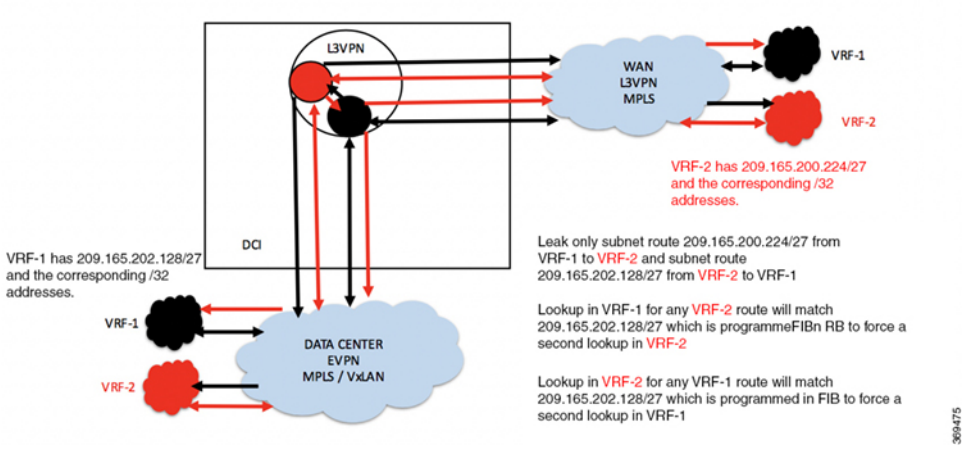
Leak Only Subnet Routes

Alternatively, in the following example, VRFs leak only /24 routes between them and they do not leak /32 route.

For forwarding to work, the traffic has to match /32 route in the original VRF.Configure the *Lookup in Source VRF* feature for the leaked /24 prefixes in destination VRF to accomplish the above-mentioned forwarding.

Lookup in VRF-1 for any VRF-2 route matches VRF-2 subnet route, which is programmed to force a second lookup in VRF-2. The second lookup finds an exact match. Similarly, lookup in VRF-2 for any VRF-1 route matches VRF-1 subnet route, which is programmed to force a second lookup in VRF-1. The second lookup finds an exact match.

Figure 32: Inter-VRF Routing - Only Subnet Routes



Configuration Example

```
RP/0/0/CPU0:router(config)# route-policy vrf-leak-from-vrf-1-to-vrf-2
RP/0/0/CPU0:router(config-rpl)# if destination in (209.165.200.224/27) then
RP/0/0/CPU0:router(config-rpl-if)# set extcommunity rt (400:1) additive
RP/0/0/CPU0:router(config-rpl-if)# pass
RP/0/0/CPU0:router(config-rpl-if)# endif
RP/0/0/CPU0:router(config-rpl)# end-policy
!
RP/0/0/CPU0:router(config)# route-policy vrf-1-import-policy
RP/0/0/CPU0:router(config-rpl)# if destination in (209.165.201.0/27) then
RP/0/0/CPU0:router(config-rpl-if)# set fallback-vrf-lookup
RP/0/0/CPU0:router(config-rpl-if)# pass
RP/0/0/CPU0:router(config-rpl-if)# endif
RP/0/0/CPU0:router(config-rpl)# end-policy
!
RP/0/0/CPU0:router(config)# route-policy vrf-leak-from-vrf-2-to-vrf-1
RP/0/0/CPU0:router(config-rpl)# if destination in (209.165.201.0/27) then
RP/0/0/CPU0:router(config-rpl-if)# set extcommunity rt (300:1) additive
RP/0/0/CPU0:router(config-rpl-if)# pass
RP/0/0/CPU0:router(config-rpl-if)# endif
RP/0/0/CPU0:router(config-rpl)# end-policy
!
RP/0/0/CPU0:router(config)# route-policy vrf-2-import-policy
RP/0/0/CPU0:router(config-rpl)# if destination in (209.165.200.224/27) then
RP/0/0/CPU0:router(config-rpl-if)# set fallback-vrf-lookup
RP/0/0/CPU0:router(config-rpl-if)# pass
```

```
RP/0/0/CPU0:router(config-rpl-if)# endif
RP/0/0/CPU0:router(config-rpl)# end-policy
!
RP/0/0/CPU0:router(config)# vrf VRF-1
RP/0/0/CPU0:router(config-vrf)# address-family ipv4 unicast
RP/0/0/CPU0:router(config-vrf-af)# import route-policy vrf-1-import-policy
RP/0/0/CPU0:router(config-vrf-af)# import from vrf advertise-as-vpn
RP/0/0/CPU0:router(config-vrf-af)# import route-target
RP/0/0/CPU0:router(config-vrf-import-rt)# 1:1
RP/0/0/CPU0:router(config-vrf-import-rt)# 300:1
RP/0/0/CPU0:router(config-vrf-import-rt)# 100:1 stitching
RP/0/0/CPU0:router(config-vrf-import-rt)# 300:1 stitching
!
RP/0/0/CPU0:router(config-vrf-af)# export route-policy vrf-leak-from-vrf-1-to-vrf-2
RP/0/0/CPU0:router(config-vrf-af)# export to vrf allow-imported-vpn
RP/0/0/CPU0:router(config-vrf-af)# export route-target
RP/0/0/CPU0:router(config-vrf-export-rt)# 1:1
RP/0/0/CPU0:router(config-vrf-export-rt)# 100:1 stitching
!

Note: The import route target 300:1 is configured only on the DCI for leaking. Do not
configure this route target 300:1 on any other router in the network.
We recommend that you do not use the same import route target, in this example, 300:1 to
prevent unintended import in the network.

RP/0/0/CPU0:router(config)# vrf VRF-2
RP/0/0/CPU0:router(config-vrf)# address-family ipv4 unicast
RP/0/0/CPU0:router(config-vrf-af)# import route-policy vrf-2-import-policy
RP/0/0/CPU0:router(config-vrf-af)# import from vrf advertise-as-vpn
RP/0/0/CPU0:router(config-vrf-af)# import route-target
RP/0/0/CPU0:router(config-vrf-import-rt)# 2:1
RP/0/0/CPU0:router(config-vrf-import-rt)# 400:1
RP/0/0/CPU0:router(config-vrf-import-rt)# 200:1 stitching
RP/0/0/CPU0:router(config-vrf-import-rt)# 400:1 stitching
!
RP/0/0/CPU0:router(config-vrf-af)# export route-policy vrf-leak-from-vrf-2-to-vrf-1
RP/0/0/CPU0:router(config-vrf-af)# export to vrf allow-imported-vpn
RP/0/0/CPU0:router(config-vrf-af)# export route-target
RP/0/0/CPU0:router(config-vrf-export-rt)# 2:1
RP/0/0/CPU0:router(config-vrf-export-rt)# 200:1 stitching
RP/0/0/CPU0:router(config-vrf-export-rt)# commit

Note: The import route target 400:1 is configured only on the DCI for leaking. Do not
configure this route target 400:1 on any other router in the network.
We recommend that you do not use the same import route target, in this example, 400:1 to
prevent unintended import in the network.
```

Running Configuration

This section shows the running configuration.

```
route-policy vrf-leak-from-vrf-1-to-vrf-2
  if destination in (209.165.200.224/27) then
    set extcommunity rt (400:1) additive
    pass
  endif
end-policy
!

route-policy vrf-1-import-policy
  if destination in (209.165.201.0/27) then
    set fallback-vrf-lookup
```

```
    pass
  endif
end-policy
!

route-policy vrf-leak-from-vrf-2-to-vrf-1
  if destination in (209.165.201.0/27) then
    set extcommunity rt (300:1) additive
    pass
  endif
end-policy
!

route-policy vrf-2-import-policy
  if destination in (209.165.200.224/27) then
    set fallback-vrf-lookup
    pass
  endif
end-policy
!

vrf VRF-1
address-family ipv4 unicast
  import route-policy vrf-1-import-policy
  import from vrf advertise-as-vpn
  import route-target
   1:1
   300:1
   100:1 stitching
   300:1 stitching
  !
  export route-policy vrf-leak-from-vrf-1-to-vrf-2
  export to vrf allow-imported-vpn
  export route-target
   1:1
   100:1 stitching
  !
!

vrf VRF-2
address-family ipv4 unicast
  import route-policy vrf-2-import-policy
  import from vrf advertise-as-vpn
  import route-target
   2:1
   400:1
   200:1 stitching
   400:1 stitching
  !
  export route-policy vrf-leak-from-vrf-2-to-vrf-1
  export to vrf allow-imported-vpn
  export route-target
   2:1
   200:1 stitching
  !
!
```

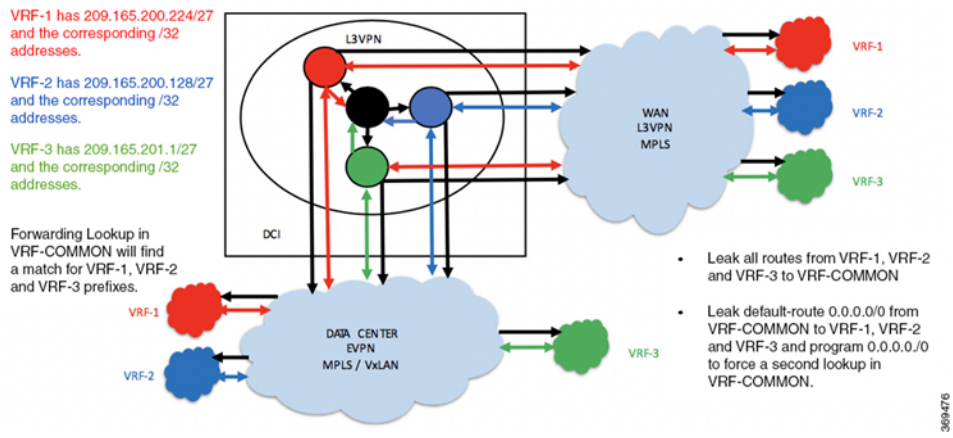
Inter-VRF Routing Between Large Number of VRFs

Leak All Routes

In this example, VRF-1, VRF-2, and VRF-3 leaks all routes to VRF-COMMON. A Static default-route 0.0.0.0/0 in VRF-COMMON is redistributed to BGP and leaked to VRF-1, VRF-2, and VRF-3. The Default-route 0.0.0.0/0 is programmed in VRF-1, VRF-2, and VRF-3 to force a fallback lookup in VRF-COMMON. VRF-1, VRF-2, and VRF-3 advertise the leaked default-route 0.0.0.0/0 with the context such as, RD and export RT of the leaked VRF, but with the local label (local vTEP in the case of VxLAN) of VRF-COMMON.

When hosts in VRF-1, VRF-2, or VRF-3 need to communicate to hosts not in its own VRF, the hosts forward the traffic to DCI. The DCI forces a lookup in VRF-COMMON, where the route finds a match and forwards the traffic to the destination host.

Figure 33: Inter-VRF Routing - Leak All Routes



Note You can enable lookup in source VRF in forwarding for a prefix leaked using **set fallback-vrf-lookup** command in the destination VRF. By default, the source VRF label for MPLS encapsulation or source VRF VNI for VxLAN encapsulation is advertised instead of destination VRF label or VNI.

For VxLAN encapsulation, if the source VRF is not configured with the VNI, then the prefix is not advertised from the destination VRF until the source VRF VNI is available. If the source VRF does not require a VNI, this default behavior can be overwritten by using the **export to vrf allow-imported-vpn disable-adv-source-vrf-vni** command in the source VRF. In this case VRF-COMMON under address family. This causes the destination VRF to send its own VNI rather than the source VRF VNI.

Configuration Example

```
RP/0/0/CPU0:router(config)# route-policy vrf-leak-to-vrf-common
RP/0/0/CPU0:router(config-rpl)# set extcommunity rt (600:1) additive
RP/0/0/CPU0:router(config-rpl)# pass
RP/0/0/CPU0:router(config-rpl)# end-policy
!
RP/0/0/CPU0:router(config)# route-policy vrf-leak-from-vrf-common-to-vrf
RP/0/0/CPU0:ROUTER(config-rpl)# if destination in (0.0.0.0/0) then
RP/0/0/CPU0:router(config-rpl-if)# set extcommunity rt (500:1) additive
```

```
RP/0/0/CPU0:router(config-rpl-if)# pass
RP/0/0/CPU0:ROUTER(config-rpl-if)# endif
RP/0/0/CPU0:router(config-rpl)# end-policy
!
RP/0/0/CPU0:ROUTER(config)# route-policy vrf-import-policy
RP/0/0/CPU0:ROUTER(config-rpl)# if destination in (0.0.0.0/0 eq 32) then
RP/0/0/CPU0:ROUTER(config-rpl-if)# set fallback-vrf-lookup
RP/0/0/CPU0:router(config-rpl-if)# pass
RP/0/0/CPU0:ROUTER(config-rpl-if)# endif
RP/0/0/CPU0:router(config-rpl)# end-policy
!
RP/0/0/CPU0:router(config)# vrf VRF-COMMON
RP/0/0/CPU0:router(config-vrf)# address-family ipv4 unicast
RP/0/0/CPU0:router(config-vrf-af)# import route-target
RP/0/0/CPU0:router(config-vrf-import-rt)# 400:1
RP/0/0/CPU0:router(config-vrf-import-rt)# 600:1
RP/0/0/CPU0:router(config-vrf-import-rt)# 400:1 stitching
RP/0/0/CPU0:router(config-vrf-import-rt)# 600:1 stitching
!
RP/0/0/CPU0:router(config-vrf)# export route-policy vrf-leak-from-vrf-common-to-vrf
RP/0/0/CPU0:router(config-vrf-af)# export route-target
RP/0/0/CPU0:router(config-vrf-export-rt)# 4:1
RP/0/0/CPU0:router(config-vrf-export-rt)# 400:1 stitching
!

Note: The import route target 600:1 is configured only on the DCI for leaking. Do not
configure this route target 600:1 on any other router in the network.
We recommend that you do not use the same import route target, in this example, 600:1 to
prevent unintended import in the network.

RP/0/0/CPU0:router(config)# vrf VRF-1
RP/0/0/CPU0:router(config-vrf)# address-family ipv4 unicast
RP/0/0/CPU0:ROUTER(config-vrf-af)# import route-policy vrf-import-policy
RP/0/0/CPU0:ROUTER(config-vrf-af)#  import from vrf advertise-as-vpn
RP/0/0/CPU0:router(config-vrf-af)# import route-target
RP/0/0/CPU0:router(config-vrf-import-rt)# 1:1
RP/0/0/CPU0:router(config-vrf-import-rt)# 500:1
RP/0/0/CPU0:router(config-vrf-import-rt)# 100:1 stitching
RP/0/0/CPU0:router(config-vrf-import-rt)# 500:1 stitching
!
RP/0/0/CPU0:router(config-vrf-af)# export route-policy vrf-leak-to-vrf-common
RP/0/0/CPU0:router(config-vrf-af)# export to vrf allow-imported-vpn
RP/0/0/CPU0:router(config-vrf-af)# export route-target
RP/0/0/CPU0:router(config-vrf-export-rt)# 1:1
RP/0/0/CPU0:router(config-vrf-export-rt)# 100:1 stitching
!
RP/0/0/CPU0:router(config)# vrf VRF-2
RP/0/0/CPU0:router(config-vrf)# address-family ipv4 unicast
RP/0/0/CPU0:ROUTER(config-vrf-af)# import route-policy vrf-import-policy
RP/0/0/CPU0:ROUTER(config-vrf-af)#  import from vrf advertise-as-vpn
RP/0/0/CPU0:router(config-vrf-af)# import route-target
RP/0/0/CPU0:router(config-vrf-import-rt)# 2:1
RP/0/0/CPU0:router(config-vrf-import-rt)# 500:1
RP/0/0/CPU0:router(config-vrf-import-rt)# 200:1 stitching
RP/0/0/CPU0:router(config-vrf-import-rt)# 500:1 stitching
!
RP/0/0/CPU0:router(config-vrf-af)# export route-policy vrf-leak-to-vrf-common
RP/0/0/CPU0:router(config-vrf-af)# export to vrf allow-imported-vpn
RP/0/0/CPU0:router(config-vrf-af)# export route-target
RP/0/0/CPU0:router(config-vrf-export-rt)# 2:1
RP/0/0/CPU0:router(config-vrf-export-rt)# 200:1 stitching
!
RP/0/0/CPU0:router(config)# vrf VRF-3
```

```
RP/0/0/CPU0:router(config-vrf)# address-family ipv4 unicast
RP/0/0/CPU0:ROUTER(config-vrf-af)# import route-policy vrf-import-policy
RP/0/0/CPU0:ROUTER(config-vrf-af)#  import from vrf advertise-as-vpn
RP/0/0/CPU0:router(config-vrf-af)# import route-target
RP/0/0/CPU0:router(config-vrf-import-rt)# 3:1
RP/0/0/CPU0:router(config-vrf-import-rt)# 500:1
RP/0/0/CPU0:router(config-vrf-import-rt)# 300:1 stitching
RP/0/0/CPU0:router(config-vrf-import-rt)# 500:1 stitching
!
RP/0/0/CPU0:router(config-vrf-af)# export route-policy vrf-leak-to-vrf-common
RP/0/0/CPU0:router(config-vrf-af)# export to vrf allow-imported-vpn
RP/0/0/CPU0:router(config-vrf-af)# export route-target
RP/0/0/CPU0:router(config-vrf-export-rt)# 3:1
RP/0/0/CPU0:router(config-vrf-export-rt)# 300:1 stitching
```

Note: The import route target 500:1 is configured only on the DCI for leaking. Do not configure this route target 500:1 on any other router in the network.
We recommend that you do not use the same import route target, in this example, 500:1 to prevent unintended import in the network.

```
/* Configuration to disable sending source VRF VNI when advertising leaked
“fallback-vrf-lookup” prefix
from destination VRF */

RP/0/0/CPU0:router(config)# vrf VRF-COMMON
RP/0/0/CPU0:router(config-vrf)#address-family ipv4 unicast
RP/0/0/CPU0:router(config-vrf-af)#export to vrf allow-imported-vpn disable-adv-source-vrf-vni
RP/0/0/CPU0:router(config-vrf)#address-family ipv6 unicast
RP/0/0/CPU0:router(config-vrf-af)#export to vrf allow-imported-vpn disable-adv-source-vrf-vni
```

Running Configuration

This section shows the running configuration.

```
route-policy vrf-leak-to-vrf-common
  set extcommunity rt (600:1) additive
  pass
end-policy
!
route-policy vrf-leak-from-vrf-common-to-vrf
  if destination in (0.0.0.0/0) then
    set extcommunity rt (500:1) additive
    pass
  endif
end-policy
!
route-policy vrf-import-policy
  if destination in (0.0.0.0/0 eq 32) then
    set fallback-vrf-lookup
    pass
  endif
end-policy
!

vrf VRF-COMMON
address-family ipv4 unicast
  import route-target
   400:1
   600:1
   400:1 stitching
   600:1 stitching
  !
```

```
  export route-policy vrf-leak-from-vrf-common-to-vrf
  export route-target
   4:1
   400:1 stitching
  !
!

vrf VRF-1
address-family ipv4 unicast
  import route-policy vrf-import-policy
  import from vrf advertise-as-vpn
  import route-target
   1:1
   500:1
   100:1 stitching
   500:1 stitching
  !
  export route-policy vrf-leak-to-vrf-common
  export to vrf allow-imported-vpn
  export route-target
  1:1
   100:1 stitching
  !
!

vrf VRF-2
address-family ipv4 unicast
  import route-policy vrf-import-policy
  import from vrf advertise-as-vpn
  import route-target
   2:1
   500:1
   200:1 stitching
   500:1 stitching
  !
  export route-policy vrf-leak-to-vrf-common
  export to vrf allow-imported-vpn
  export route-target
   2:1
   200:1 stitching
  !
!
vrf VRF-3
address-family ipv4 unicast
  import route-policy vrf-import-policy
  import from vrf advertise-as-vpn
  import route-target
   3:1
   500:1
   300:1 stitching
   500:1 stitching
  !
  export route-policy vrf-leak-to-vrf-common
  export to vrf allow-imported-vpn
  export route-target
   3:1
   300:1 stitching
  !
!
```

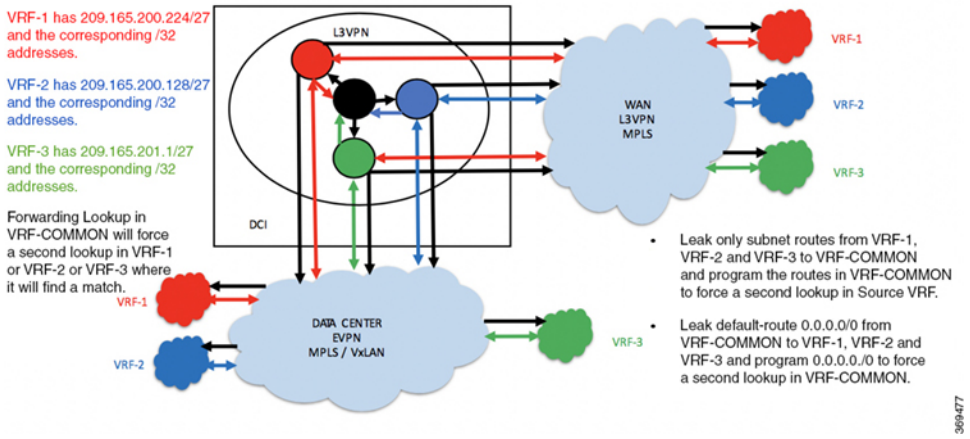
Leak Only Subnet Routes

Alternatively, in this example, VRF-1, VRF-2 and VRF-3 leaks only /24 routes to VRF-COMMON. They do not leak /32 routes. For forwarding to work, the traffic has to match /32 route in the original VRF. Configure the “Lookup in Source VRF” feature for the leaked /24 prefixes in VRF-COMMON.

In this example, VRF-1, VRF-2, and VRF-3 leaks only /24 routes to VRF-COMMON. A static default-route 0.0.0.0/0 in VRF-COMMON is redistributed to BGP and leaked to VRF-1, VRF-2, and VRF-3. The Default-route 0.0.0.0/0 is programmed in VRF-1, VRF-2, and VRF-3 to force a fallback lookup in VRF-COMMON.VRF-1, VRF-2, and VRF-3 advertise the leaked default-route 0.0.0.0/0 with the context such as, RD and export RT of the leaked VRF, but with the local label (local vTEP in the case of VxLAN) of VRF-COMMON.

When hosts in VRF-1, VRF-2, or VRF-3 need to communicate to hosts not in its own VRF, the hosts forward the traffic to DCI. The DCI forces a lookup in VRF-COMMON, where the route finds a match against /24 prefix, forcing a second look up in the original VRF, where it matches against /32 entry and forwards the traffic to the destination host.

Figure 34: Inter-VRF Routing - Leak Only Subnet Routes



Note You can enable lookup in source VRF in forwarding for a prefix leaked using **set fallback-vrf-lookup** command in the destination VRF. By default, the source VRF label for MPLS encapsulation or source VRF VNI for VxLAN encapsulation is advertised instead of destination VRF label or VNI.

For VxLAN encapsulation, if the source VRF is not configured with the VNI, then the prefix is not advertised from the destination VRF until the source VRF VNI is available. If the source VRF does not require a VNI, this default behavior can be overwritten by using the **export to vrf allow-imported-vpn disable-adv-source-vrf-vni** command in the source VRF. In this case VRF-COMMON under address family. This causes the destination VRF to send its own VNI rather than the source VRF VNI.

Configuration Example

```
RP/0/0/CPU0:router(config)# route-policy vrf-leak-from-vrf-1-to-vrf-common
RP/0/0/CPU0:router(config-rpl)# if destination in (209.165.200.224/27) then
RP/0/0/CPU0:router(config-rpl-if)# set extcommunity rt (600:1) additive
RP/0/0/CPU0:router(config-rpl-if)# pass
RP/0/0/CPU0:router(config-rpl-if)# endif
```

```
RP/0/0/CPU0:router(config-rpl)# end-policy
!
RP/0/0/CPU0:router(config)# route-policy vrf-leak-from-vrf-2-to-vrf-common
RP/0/0/CPU0:router(config-rpl)# if destination in (209.165.201.0/27) then
RP/0/0/CPU0:router(config-rpl-if)# set extcommunity rt (600:1) additive
RP/0/0/CPU0:router(config-rpl-if)# pass
RP/0/0/CPU0:router(config-rpl-if)# endif
RP/0/0/CPU0:router(config-rpl)# end-policy
!
RP/0/0/CPU0:router(config)# route-policy vrf-leak-from-vrf-3-to-vrf-common
RP/0/0/CPU0:router(config-rpl)# if destination in (209.165.202.128/27) then
RP/0/0/CPU0:router(config-rpl-if)# set extcommunity rt (600:1) additive
RP/0/0/CPU0:router(config-rpl-if)# pass
RP/0/0/CPU0:router(config-rpl-if)# endif
RP/0/0/CPU0:router(config-rpl)# end-policy
!
RP/0/0/CPU0:router(config)# route-policy vrf-leak-from-vrf-common-to-vrf
RP/0/0/CPU0:router(config-rpl)# if destination in (0.0.0.0/0) then
RP/0/0/CPU0:router(config-rpl-if)# set extcommunity rt (500:1) additive
RP/0/0/CPU0:router(config-rpl-if)# pass
RP/0/0/CPU0:router(config-rpl-if)# endif
RP/0/0/CPU0:router(config-rpl)# end-policy
!
RP/0/0/CPU0:router(config)# route-policy vrf-common-import-policy
RP/0/0/CPU0:ROUTER(config-rpl)# if destination in (209.165.200.224/27, 209.165.201.0/27,
209.165.202.128/27) then
RP/0/0/CPU0:ROUTER(config-rpl-if)# set fallback-vrf-lookup
RP/0/0/CPU0:router(config-rpl-if)# pass
RP/0/0/CPU0:router(config-rpl-if)# endif
RP/0/0/CPU0:router(config-rpl)# end-policy
!
RP/0/0/CPU0:ROUTER(config)# route-policy vrf-import-policy
RP/0/0/CPU0:ROUTER(config-rpl)# if destination in (0.0.0.0/0 eq 32) then
RP/0/0/CPU0:ROUTER(config-rpl-if)# set fallback-vrf-lookup
RP/0/0/CPU0:router(config-rpl-if)# pass
RP/0/0/CPU0:router(config-rpl-if)# endif
RP/0/0/CPU0:router(config-rpl)# end-policy
!
RP/0/0/CPU0:ROUTER(config)# vrf VRF-COMMON
RP/0/0/CPU0:router(config-vrf)# address-family ipv4 unicast
RP/0/0/CPU0:router(config-vrf-af)# import route-policy vrf-common-import-policy
RP/0/0/CPU0:router(config-vrf-af)# import route-target
RP/0/0/CPU0:router(config-vrf-import-rt)# 400:1
RP/0/0/CPU0:router(config-vrf-import-rt)# 600:1
RP/0/0/CPU0:router(config-vrf-import-rt)# 400:1 stitching
RP/0/0/CPU0:router(config-vrf-import-rt)# 600:1 stitching
!
RP/0/0/CPU0:router(config-vrf-af)# export route-policy vrf-leak-from-vrf-common-to-vrf
RP/0/0/CPU0:router(config-vrf-af)# export route-target
RP/0/0/CPU0:router(config-vrf-export-rt)# 4:1
RP/0/0/CPU0:router(config-vrf-export-rt)# 400:1 stitching

Note: The import route target 600:1 is configured only on the DCI for leaking. Do not
configure this route target 600:1 on any other router in the network.
We recommend that you do not use the same import route target, in this example, 600:1 to
prevent unintended import in the network.

RP/0/0/CPU0:ROUTER(config)# vrf VRF-1
RP/0/0/CPU0:router(config-vrf)# address-family ipv4 unicast
RP/0/0/CPU0:router(config-vrf-af)# import route-policy vrf-import-policy
RP/0/0/CPU0:router(config-vrf-af)# import from vrf advertise-as-vpn
RP/0/0/CPU0:router(config-vrf-af)# import route-target
RP/0/0/CPU0:router(config-vrf-import-rt)# 1:1
```

```
RP/0/0/CPU0:router(config-vrf-import-rt)# 500:1
RP/0/0/CPU0:router(config-vrf-import-rt)# 100:1 stitching
RP/0/0/CPU0:router(config-vrf-import-rt)# 500:1 stitching
!
RP/0/0/CPU0:router(config-vrf-af)# export route-policy vrf-leak-from-vrf-1-to-vrf-common
RP/0/0/CPU0:router(config-vrf-af)# export to vrf allow-imported-vpn
RP/0/0/CPU0:router(config-vrf-af)# export route-target
RP/0/0/CPU0:router(config-vrf-export-rt)# 1:1
RP/0/0/CPU0:router(config-vrf-export-rt)# 100:1 stitching
!
RP/0/0/CPU0:ROUTER(config)# vrf VRF-2
RP/0/0/CPU0:router(config-vrf)# address-family ipv4 unicast
RP/0/0/CPU0:router(config-vrf-af)# import route-policy vrf-import-policy
RP/0/0/CPU0:router(config-vrf-af)# import from vrf advertise-as-vpn
RP/0/0/CPU0:router(config-vrf-af)# import route-target
RP/0/0/CPU0:router(config-vrf-import-rt)# 2:1
RP/0/0/CPU0:router(config-vrf-import-rt)# 500:1
RP/0/0/CPU0:router(config-vrf-import-rt)# 200:1 stitching
RP/0/0/CPU0:router(config-vrf-import-rt)# 500:1 stitching
!
RP/0/0/CPU0:router(config-vrf-af)# export route-policy vrf-leak-from-vrf-2-to-vrf-common
RP/0/0/CPU0:router(config-vrf-af)# export to vrf allow-imported-vpn
RP/0/0/CPU0:router(config-vrf-af)# export route-target
RP/0/0/CPU0:router(config-vrf-export-rt)# 2:1
RP/0/0/CPU0:router(config-vrf-export-rt)# 200:1 stitching
!
RP/0/0/CPU0:ROUTER(config)# vrf VRF-3
RP/0/0/CPU0:router(config-vrf)# address-family ipv4 unicast
RP/0/0/CPU0:router(config-vrf-af)# import route-policy vrf-import-policy
RP/0/0/CPU0:router(config-vrf-af)# import from vrf advertise-as-vpn
RP/0/0/CPU0:router(config-vrf-af)# import route-target
RP/0/0/CPU0:router(config-vrf-import-rt)# 3:1
RP/0/0/CPU0:router(config-vrf-import-rt)# 500:1
RP/0/0/CPU0:router(config-vrf-import-rt)# 300:1 stitching
RP/0/0/CPU0:router(config-vrf-import-rt)# 500:1 stitching
!
RP/0/0/CPU0:router(config-vrf-af)# export route-policy vrf-leak-from-vrf-3-to-vrf-common
RP/0/0/CPU0:router(config-vrf-af)# export to vrf allow-imported-vpn
RP/0/0/CPU0:router(config-vrf-af)# export route-target
RP/0/0/CPU0:router(config-vrf-export-rt)# 3:1
RP/0/0/CPU0:router(config-vrf-export-rt)# 300:1 stitching

Note: The import route target 500:1 is configured only on the DCI for leaking. Do not
configure this route target 500:1 on any other router in the network.
We recommend that you do not use the same import route target, in this example, 500:1 to
prevent unintended import in the network.

/* Configuration to disable sending source VRF VNI when advertising leaked
“fallback-vrf-lookup” prefix
from destination VRF */

RP/0/0/CPU0:router(config)# vrf VRF-COMMON
RP/0/0/CPU0:router(config-vrf)#address-family ipv4 unicast
RP/0/0/CPU0:router(config-vrf-af)#export to vrf allow-imported-vpn disable-adv-source-vrf-vni
RP/0/0/CPU0:router(config-vrf)#address-family ipv6 unicast
RP/0/0/CPU0:router(config-vrf-af)#export to vrf allow-imported-vpn disable-adv-source-vrf-vni
```

Running Configuration

This section shows the running configuration.

```
route-policy vrf-leak-from-vrf-1-to-vrf-common
  if destination in (209.165.200.224/27) then
```

```
    set extcommunity rt (600:1) additive
    pass
  endif
end-policy
!
route-policy vrf-leak-from-vrf-2-to-vrf-common
  if destination in (209.165.201.0/27) then
    set extcommunity rt (600:1) additive
    pass
  endif
end-policy
!
route-policy vrf-leak-from-vrf-3-to-vrf-common
  if destination in (209.165.202.128/27) then
    set extcommunity rt (600:1) additive
    pass
  endif
end-policy
!
route-policy vrf-leak-from-vrf-common-to-vrf
  if destination in (0.0.0.0/0) then
    set extcommunity rt (500:1) additive
    pass
  endif
end-policy
!
route-policy vrf-common-import-policy
  if destination in (209.165.200.224/27, 209.165.201.0/27, 209.165.202.128/27) then
    set fallback-vrf-lookup
    pass
  endif
end-policy
!

route-policy vrf-import-policy
  if destination in (0.0.0.0/0 eq 32) then
    set fallback-vrf-lookup
    pass
  endif
end-policy
!

vrf VRF-COMMON
address-family ipv4 unicast
  import route-policy vrf-common-import-policy
  import route-target
   400:1
   600:1
   400:1 stitching
   600:1 stitching
  !
  export route-policy vrf-leak-from-vrf-common-to-vrf
  export route-target
   4:1
   400:1 stitching
  !
!
vrf VRF-1
address-family ipv4 unicast
  import route-policy vrf-import-policy
  import from vrf advertise-as-vpn
  import route-target
   1:1
   500:1
```

```
   100:1 stitching
   500:1 stitching
  !
  export route-policy vrf-leak-from-vrf-1-to-vrf-common
  export to vrf allow-imported-vpn
  export route-target
   1:1
   100:1 stitching
  !
!

vrf VRF-2
address-family ipv4 unicast
  import route-policy vrf-import-policy
  import from vrf advertise-as-vpn
  import route-target
   2:1
   500:1
   200:1 stitching
   500:1 stitching
  !
  export route-policy vrf-leak-from-vrf-2-to-vrf-common
  export to vrf allow-imported-vpn
  export route-target
   2:1
   200:1 stitching
  !
!
vrf VRF-3
address-family ipv4 unicast
  import route-policy vrf-import-policy
  import from vrf advertise-as-vpn
  import route-target
   3:1
   500:1
   300:1 stitching
   500:1 stitching
  !
  export route-policy vrf-leak-from-vrf-3-to-vrf-common
  export to vrf allow-imported-vpn
  export route-target
   3:1
   300:1 stitching
  !
!
```

OpFlex

Table 5: Feature History Table

Feature Name	Release Information	Feature Description
OpFlex Interop with ACI	Release 7.4.1	The OpFlex session between the OpFlex client running on ASR9k and the OpFlex server uses Transport Layer Security (TLS). With this release, this feature supports TLSv1.1 and TLSv1.2 to securely establish the session.

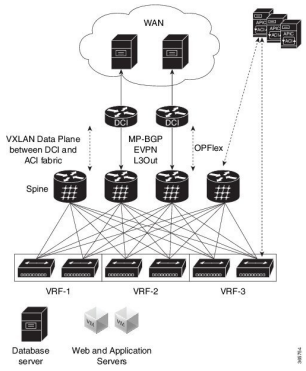
OpFlex is an open and extensible policy protocol used for transferring the policy information between a network policy controller such as the Cisco Application Policy Infrastructure Controller (APIC) and network

elements such as routers that are configured as Data Center Interconnect (DCI) gateway. The policies are distributed using the Cisco® Application Centric Infrastructure (ACI) infrastructure within the fabric to the spine nodes. The spine nodes send policies to the DCI gateway through the OpFlex framework. An OpFlex framework resides between the spines and the DCIs. It enables the distribution of the DCI policy model from the fabric to the DCI gateways. DCI gateway acts as an OpFlex agent and the spine acts a policy repository. Fabric tenant interconnect (FTI) is the OpFlex agent application that runs on the DCI to generate and apply the tenant device configuration on the DCI. Policies configure the DCI service for a given tenant on the DCI gateway.

OpFlex Topology

Consider the topology where OpFlex framework is used between the DCI gateway and the Cisco ACI spine switches to automate fabric-facing tenant provisioning on the DCI gateway. When you configure a new external Layer 3 outside (L3Out) policy for a tenant on the Cisco Application Policy Infrastructure Controller (APIC), the controller programs all related information associated with that tenant, such as VRF instance name and BGP extended community route-target attributes for the Cisco ACI spine switches. The OpFlex framework running on the spine switches reads the L3Out managed object and converts it to the OpFlex model. This information is then pushed to the DCI gateway, which acts as a policy element for the OpFlex framework. On the DCI, the fabric facing configuration for the tenant VFR is auto-generated.

Figure 35: OpFlex Topology



Restrictions

The OpFlex feature is supported with the following restrictions:

- OpFlex feature is not supported on ASR 9000 series router with power PC based route-processor.
- FTI cannot generate configuration for multiple RTs of one address family in a tenant VRF provisioned in one fabric.
- The FTI configuration gets deleted during the following scenarios:
  - If the complete DCI configuration gets removed.
  - If the fabric gets removed or the corresponding parts of each fabric get removed.
  - If the last OpFlex peer gets removed.
- On exhaustion of FTI configuration pools, the OpFlex notifications to add tenants are ignored. If existing tenants are deleted, the new tenants must be added again to enable OpFlex notifications to be re-sent to the DCI.

- FTI supports only Type 0 RT format: 2 byte ASN + 4 byte value. Type 1 and Type 2 RT formats are not supported.
- XML configuration and oper schema are not supported for FTI configuration and show commands.
- While removing the OpFlex peers:
  - When OpFlex peer is removed from an OpFlex session, the session is shut down and the corresponding tenant configuration is marked as stale. The stale entries are deleted after the sweep timer expires. The OpFlex session is updated with the remaining peers.
  - When the last OpFlex peer is removed from an OpFlex session, the session is shut down and the corresponding tenant configuration is deleted. The OpFlex session is not updated.

Configure OpFlex

Perform the following tasks to configure the OpFlex session to automate fabric-facing tenant provisioning on the DCI gateway. This includes the one-time configuration that must be done on the DCI to enable DCI hand-off from an ACI fabric.

Configure BGP

Perform this task to enable address-family under BGP routing process for fabric and WAN peering.

```
Router# configure
Router(config)# router bgp 1234
Router(config-bgp)# bgp router-id 198.51.100.1
Router(config-bgp)# address-family vpnv4 unicast
Router(config-bgp-af)# commit
```

Configure BGP Session on the Fabric Side

Perform this task to configure BGP session on the fabric side.

```
Router# configure
Router(config)# router bgp 200
Router(config-bgp)# neighbor 209.165.201.1
Router(config-bgp-nbr)# remote-as 100
Router(config-bgp-nbr)# update-source loopback2
Router(config-bgp-nbr)# address-fmaily l2vpn evpn
Router(config-bgp-nbr-af)# import stitching-rt reoriginate
Router(config-bgp-nbr-af)# advertise vpnv4 unicast re-originated
Router(config-bgp-nbr-af)# commit
```

Configure BGP Session on the WAN Side

Perform this task to configure BGP session on the WAN side.

```
Router# configure
Router(config)# router bgp 200
Router(config-bgp)# neighbor 209.165.200.226
Router(config-bgp-nbr)# remote-as 100
Router(config-bgp-nbr)# update-source loopback2
Router(config-bgp-nbr)# address-family vpnv4 unicast
Router(config-bgp-nbr-af)# import re-originate stitching-rt
Router(config-bgp-nbr-af)# advertise vpnv4 unicast re-originated
Router(config-bgp-nbr-af)# commit
```

Configure DCI Underlay for Fabric and WAN Interfaces

Perform this task to configure DCI underlay for fabric facing interface and WAN facing interface. Perform this task on both the interfaces.

```
Router# configure
Router(config)# interface GigabitEthernet 0/0/0/0
Router(config-if)# ipv4 address 209.165.200.226 255.255.255.224
Router(config-if)# commit
```

Configure IGP for ACI and WAN Reachability

Perform this task to configure IGP for ACI and WAN reachability.

```
Router# configure
Router(config)# router ospf 100
Router(config-ospf)# area 0
Router(config-ospf-ar)# interface GigabitEthernet 0/0/0/1
Router(config-ospf-ar-if)# exit
Router(config-ospf-ar)# exit
Router(config-ospf)# area 100
Router(config-ospf-ar)# nssa
Router(config-ospf-ar)# interface loopback0
Router(config-ospf-ar-if)# exit
Router(config-ospf-ar)# interface GigabitEthernet 0/0/0/0
Router(config-ospf-ar)# commit
```

Configure MPLS towards WAN

Perform this task to configure MPLS on the DCI.

```
Router# configure
Router# mpls ldp
Router(config-ldp)# interface GigabitEthernet 0/0/0/1
Router(config-ldp-if)# exit
Router(config-ldp)# exit
Router(config)# interface Loopback0
Router(config-if)# ipv4 address 209.165.200.227 255.255.255.224
Router(config-if)# exit
Router(config)# interface nve 1
Router(config-if)# source-interface loopback 0
Router(config-if)# commit
```

Configure FTI Auto-Configuration Parameters

Perform this task to configure FTI auto-configuration parameters.

```
Router# configure
Router(config)# dci-fabric-interconnect
Router(config-fti)# auto-configuration-pool
Router(config-fti-acp)# bgp-as 1234
Router(config-fti-acp)# bridge group bg1
Router(config-fti-acp)# vrf vrf1 ipv4-address 198.51.100.1
Router(config-fti-acp)# bd-pool 1 1000
Router(config-fti-acp)# vni-pool 1 1000
Router(config-fti-acp)# local-vtep nve 1
Router(config-fti-acp)# commit
```

Configure OpFlex Session

This task enables the fabric tenant interconnect to setup an OpFlex session with the spine.

```
Router# configure
Router(config)# dci-fabric-interconnect
Router(config-fti)# fabric 1001
Router(config-fti-fabric)# opflex-peer 192.0.2.1
Router(config-fti-fabric)# exit
Router(config-fti)# identity 203.0.113.1
Router(config-fti)# commit
```

OpFlex using Loopback Interface

The OpFlex using Loopback Interface feature prevents flapping of OpFlex session when one of the physical connections from the Data Center Interconnect (DCI) to the spine goes down. The loopback IP address which serves as the identity of the OpFlex session is used to establish the connection to the spine.

When an OpFlex session is established between the DCI and the spine where the OpFlex server is running, the session uses the physical IP address of the interface connection between the DCI and the spine to send information. When the physical connection goes down, the OpFlex session is brought down. If there is another physical connection between the spine and the DCI, the session gets re-established with this new physical IP address. However, this causes a flap. This feature enables you to have the OpFlex session up and running even when one of the physical connections to the spine goes down.

To enable this feature, use **identity loopback intf-name** command.

If you use the **identity loopback ip-address** command when the loopback IP address is unreachable, the OpFlex session falls back to using the physical IP address and continues to use even after the loopback becomes reachable. When you use the **identity loopback intf-name** command, the OpFlex session is up only as long as the loopback interface is reachable. However, only one of these two commands can be configured at a given time.

Configure OpFlex using Loopback Interface

Perfrom these tasks to configure OpFlex using loopback interface.

Configuration Example

```
RP/0/RSP0/CPU0:router# configure
RP/0/RSP0/CPU0:router(config)# dci-fabric-interconnect
RP/0/RSP0/CPU0:router(config-fti)# auto-configuration-pool
RP/0/RSP0/CPU0:router(config-fti-acp)# bgp-as 100
RP/0/RSP0/CPU0:router(config-fti-acp)# bridge group bg 1001
RP/0/RSP0/CPU0:router(config-fti-acp)# bd-pool 1001 4000
RP/0/RSP0/CPU0:router(config-fti-acp)# bvi-pool 1001 4000
RP/0/RSP0/CPU0:router(config-fti-acp)# vni-pool 1001 4000
RP/0/RSP0/CPU0:router(config-fti-acp)# local-vtep nve 1001
RP/0/RSP0/CPU0:router(config-fti-acp)# exit
RP/0/RSP0/CPU0:router(config-fti)# fabric 1001
RP/0/RSP0/CPU0:router(config-fti-fabric)# opflex-peer 192.0.2.1
RP/0/RSP0/CPU0:router(config-fti-fabric)# opflex-peer 192.0.2.2
RP/0/RSP0/CPU0:router(config-fti-fabric)# exit
RP/0/RSP0/CPU0:router(config-fti)# fabric 1002
RP/0/RSP0/CPU0:router(config-fti-fabric)# opflex-peer 192.0.2.3
RP/0/RSP0/CPU0:router(config-fti-fabric)# exit
RP/0/RSP0/CPU0:router(config-fti)# fabric 1003
RP/0/RSP0/CPU0:router(config-fti-fabric)# opflex-peer 192.0.2.4
RP/0/RSP0/CPU0:router(config-fti-fabric)# exit
RP/0/RSP0/CPU0:router(config-fti)# identity Loopback0
RP/0/RSP0/CPU0:router(config-fti)# commit
```

Running Configuration

This section shows OpFlex using loopback interface running configuration.

```
dci-fabric-interconnect
 auto-configuration-pool
  bgp-as 100
  bridge-group bg1001
  bd-pool 1001 4000
  bvi-pool 1001 4000
  vni-pool 1001 4000
  local-vtep nve 1001
 !
 fabric 1001
  opflex-peer 192.0.2.1
  opflex-peer 192.0.2.2
 !
 fabric 1002
  opflex-peer 192.0.2.3
 !
 fabric 1003
  opflex-peer 192.0.2.4
 !
 identity Loopback0
!
```

Related Topics

- OpFlex using Loopback Interface, on page 251

Associated Commands

show dci-fabric-interconnect