JUNIPER NETWORKS

Junos® OS

Layer 3 VPNs Configuration Guide

Release
12.1

Published: 2012-03-13

Juniper Networks, Inc.
1194 North Mathilda Avenue
Sunnyvale, California 94089
USA
408-745-2000
www.juniper.net

Junos® OS Layer 3 VPNs Configuration Guide
12.1

The information in this document is current as of the date on the title page.

YEAR 2000 NOTICE

Juniper Networks hardware and software products are Year 2000 compliant. Junos OS has no known time-related limitations through the year 2038. However, the NTP application is known to have some difficulty in the year 2036.

END USER LICENSE AGREEMENT

The Juniper Networks product that is the subject of this technical documentation consists of (or is intended for use with) Juniper Networks software. Use of such software is subject to the terms and conditions of the End User License Agreement ("EULA") posted at http://www.juniper.net/support/eula.html. By downloading, installing or using such software, you agree to the terms and conditions of that EULA.

Table of Contents

Part 2 Configuration

List of Figures

List of Tables

About the Documentation

Documentation and Release Notes

To obtain the most current version of all Juniper Networks® technical documentation, see the product documentation page on the Juniper Networks website at http://www.juniper.net/techpubs/.

If the information in the latest release notes differs from the information in the documentation, follow the product Release Notes.

Juniper Networks Books publishes books by Juniper Networks engineers and subject matter experts. These books go beyond the technical documentation to explore the nuances of network architecture, deployment, and administration. The current list can be viewed at http://www.juniper.net/books.

Using the Examples in This Manual

If you want to use the examples in this manual, you can use the **load merge** or the **load merge relative** command. These commands cause the software to merge the incoming configuration into the current candidate configuration. The example does not become active until you commit the candidate configuration.

If the example configuration contains the top level of the hierarchy (or multiple hierarchies), the example is a *full example*. In this case, use the **load merge** command.

If the example configuration does not start at the top level of the hierarchy, the example is a *snippet*. In this case, use the **load merge relative** command. These procedures are described in the following sections.

Merging a Full Example

To merge a full example, follow these steps:

1. From the HTML or PDF version of the manual, copy a configuration example into a text file, save the file with a name, and copy the file to a directory on your routing platform.

   For example, copy the following configuration to a file and name the file **ex-script.conf**. Copy the **ex-script.conf** file to the **/var/tmp** directory on your routing platform.

   ```
   system {
     scripts {
       commit {
         file ex-script.xsl;
       }
     }
   }
   interfaces {
     fxp0 {
       disable;
       unit 0 {
         family inet {
           address 10.0.0.1/24;
         }
       }
     }
   }
   ```

2. Merge the contents of the file into your routing platform configuration by issuing the **load merge** configuration mode command:

   ```
   [edit]
   user@host# load merge /var/tmp/ex-script.conf
   load complete
   ```

Merging a Snippet

To merge a snippet, follow these steps:

1. From the HTML or PDF version of the manual, copy a configuration snippet into a text file, save the file with a name, and copy the file to a directory on your routing platform.

   For example, copy the following snippet to a file and name the file **ex-script-snippet.conf**. Copy the **ex-script-snippet.conf** file to the **/var/tmp** directory on your routing platform.

   ```
   commit {
     file ex-script-snippet.xsl; }
   ```

2. Move to the hierarchy level that is relevant for this snippet by issuing the following configuration mode command:

   ```
   [edit]
   user@host# edit system scripts
   [edit system scripts]
   ```

3. Merge the contents of the file into your routing platform configuration by issuing the **load merge relative** configuration mode command:

```
[edit system scripts]
user@host# load merge relative /var/tmp/ex-script-snippet.conf
load complete
```

For more information about the **load** command, see the *Junos OS CLI User Guide*.

Documentation Conventions

Table 1 on page xvii defines notice icons used in this guide.

Table 1: Notice Icons

Icon	Meaning	Description
	Informational note	Indicates important features or instructions.
	Caution	Indicates a situation that might result in loss of data or hardware damage.
	Warning	Alerts you to the risk of personal injury or death.
	Laser warning	Alerts you to the risk of personal injury from a laser.

Table 2 on page xvii defines the text and syntax conventions used in this guide.

Table 2: Text and Syntax Conventions

Convention	Description	Examples
Bold text like this	Represents text that you type.	To enter configuration mode, type the **configure** command: user@host> **configure**
`Fixed-width text like this`	Represents output that appears on the terminal screen.	`user@host>` **show chassis alarms** `No alarms currently active`
Italic text like this	• Introduces important new terms. • Identifies book names. • Identifies RFC and Internet draft titles.	• A policy *term* is a named structure that defines match conditions and actions. • *Junos OS System Basics Configuration Guide* • RFC 1997, *BGP Communities Attribute*

Table 2: Text and Syntax Conventions *(continued)*

Convention	Description	Examples
Italic text like this	Represents variables (options for which you substitute a value) in commands or configuration statements.	Configure the machine's domain name: [edit] root@# **set system domain-name** *domain-name*
Text like this	Represents names of configuration statements, commands, files, and directories; interface names; configuration hierarchy levels; or labels on routing platform components.	• To configure a stub area, include the **stub** statement at the **[edit protocols ospf area area-id]** hierarchy level. • The console port is labeled **CONSOLE**.
< > (angle brackets)	Enclose optional keywords or variables.	**stub** <**default-metric** *metric*>**;**
| (pipe symbol)	Indicates a choice between the mutually exclusive keywords or variables on either side of the symbol. The set of choices is often enclosed in parentheses for clarity.	**broadcast | multicast** (***string1*** **|** ***string2*** **|** ***string3***)
# (pound sign)	Indicates a comment specified on the same line as the configuration statement to which it applies.	**rsvp { # Required for dynamic MPLS only**
[ ] (square brackets)	Enclose a variable for which you can substitute one or more values.	**community name members [** ***community-ids*** **]**

Indention and braces ({ })	Identify a level in the configuration hierarchy.	[edit]
		routing-options {
		  static {
		  route default {
		  nexthop *address*;
		  retain;
		  }
		  }
		}

; (semicolon)	Identifies a leaf statement at a configuration hierarchy level.	
J-Web GUI Conventions		
Bold text like this	Represents J-Web graphical user interface (GUI) items you click or select.	• In the Logical Interfaces box, select **All Interfaces**. • To cancel the configuration, click **Cancel**.
> (bold right angle bracket)	Separates levels in a hierarchy of J-Web selections.	In the configuration editor hierarchy, select **Protocols>Ospf**.

Documentation Feedback

We encourage you to provide feedback, comments, and suggestions so that we can improve the documentation. You can send your comments to **techpubs-comments@juniper.net**, or fill out the documentation feedback form at

https://www.juniper.net/cgi-bin/docbugreport/ . If you are using e-mail, be sure to include the following information with your comments:

- Document or topic name
- URL or page number
- Software release version (if applicable)

Requesting Technical Support

Technical product support is available through the Juniper Networks Technical Assistance Center (JTAC). If you are a customer with an active J-Care or JNASC support contract, or are covered under warranty, and need post-sales technical support, you can access our tools and resources online or open a case with JTAC.

- JTAC policies—For a complete understanding of our JTAC procedures and policies, review the *JTAC User Guide* located at http://www.juniper.net/us/en/local/pdf/resource-guides/7100059-en.pdf .
- Product warranties—For product warranty information, visit http://www.juniper.net/support/warranty/ .
- JTAC hours of operation—The JTAC centers have resources available 24 hours a day, 7 days a week, 365 days a year.

Self-Help Online Tools and Resources

For quick and easy problem resolution, Juniper Networks has designed an online self-service portal called the Customer Support Center (CSC) that provides you with the following features:

- Find CSC offerings: http://www.juniper.net/customers/support/
- Search for known bugs: http://www2.juniper.net/kb/
- Find product documentation: http://www.juniper.net/techpubs/
- Find solutions and answer questions using our Knowledge Base: http://kb.juniper.net/
- Download the latest versions of software and review release notes: http://www.juniper.net/customers/csc/software/
- Search technical bulletins for relevant hardware and software notifications: https://www.juniper.net/alerts/
- Join and participate in the Juniper Networks Community Forum: http://www.juniper.net/company/communities/
- Open a case online in the CSC Case Management tool: http://www.juniper.net/cm/

To verify service entitlement by product serial number, use our Serial Number Entitlement (SNE) Tool: https://tools.juniper.net/SerialNumberEntitlementSearch/

Opening a Case with JTAC

You can open a case with JTAC on the Web or by telephone.

- Use the Case Management tool in the CSC at http://www.juniper.net/cm/ .
- Call 1-888-314-JTAC (1-888-314-5822 toll-free in the USA, Canada, and Mexico).

For international or direct-dial options in countries without toll-free numbers, see http://www.juniper.net/support/requesting-support.html .

PART 1

Overview

CHAPTER 1

Introduction to Layer 3 VPNs

Layer 3 VPN Introduction

In Junos OS, Layer 3 VPNs are based on RFC 4364. RFC 4364 defines a mechanism by which service providers can use their IP backbones to provide VPN services to their customers. A Layer 3 VPN is a set of sites that share common routing information and whose connectivity is controlled by a collection of policies. The sites that make up a Layer 3 VPN are connected over a provider's existing public Internet backbone.

RFC 4364 VPNs are also known as BGP/MPLS VPNs because BGP is used to distribute VPN routing information across the provider's backbone, and MPLS is used to forward VPN traffic across the backbone to remote VPN sites.

Customer networks, because they are private, can use either public addresses or private addresses, as defined in RFC 1918, *Address Allocation for Private Internets*. When customer networks that use private addresses connect to the public Internet infrastructure, the private addresses might overlap with the same private addresses used by other network users. MPLS/BGP VPNs solve this problem by adding a VPN identifier prefix to each address from a particular VPN site, thereby creating an address that is unique both within the VPN and within the public Internet. In addition, each VPN has its own VPN-specific routing table that contains the routing information for that VPN only.

Layer 3 VPN Platform Support

Layer 3 VPNs are supported on most combinations of Juniper Networks routing platforms and PICs capable of running the JUNOS Software.

MX Series routers configured to be in Ethernet services mode can support some of the Junos OS Layer 3 VPN features. For Layer 3 VPNs, Ethernet services mode supports configuring a loopback interface for a VPN routing and forwarding (VRF) instance. You can configure up to two VRF instances in Ethernet services mode. Each VRF instance can handle up to 10,000 routes. The **ping mpls l3vpn** operational mode command is also supported.

Layer 3 VPN Attributes

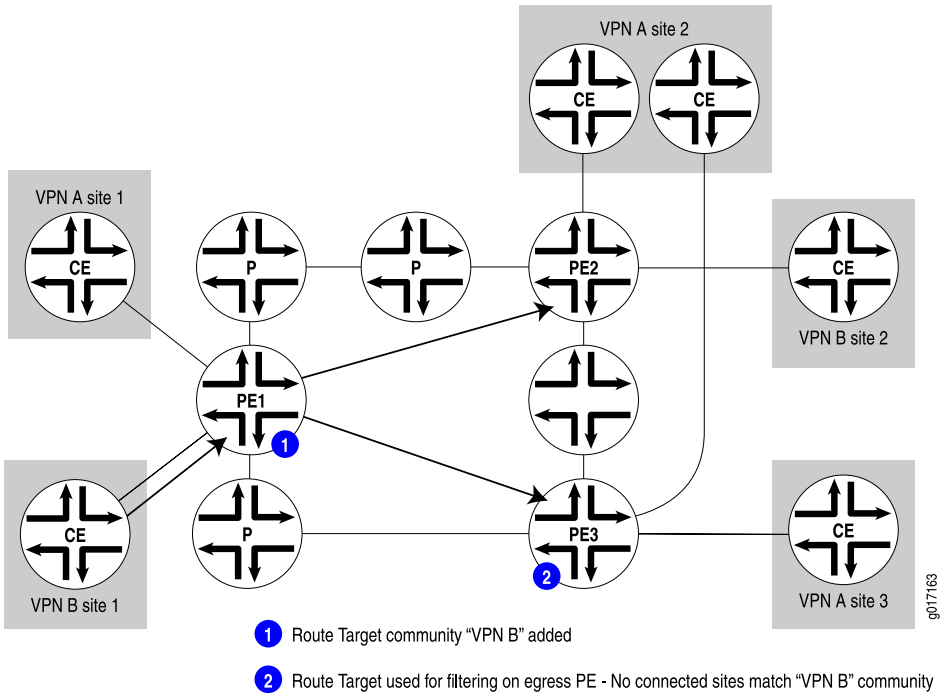
Route distribution within a VPN is controlled through BGP extended community attributes. RFC 4364 defines the following three attributes used by VPNs:

- Target VPN—Identifies a set of sites within a VPN to which a provider edge (PE) router distributes routes. This attribute is also called the *route target*. The route target is used by the egress PE router to determine whether a received route is destined for a VPN that the router services.

  Figure 1 on page 5 illustrates the function of the route target. PE Router PE1 adds the route target “VPN B” to routes received from the customer edge (CE) router at Site 1 in VPN B. When it receives the route, the egress router PE2 examines the route target, determines that the route is for a VPN that it services, and accepts the route. When the egress router PE3 receives the same route, it does not accept the route because it does not service any CE routers in VPN B.

- VPN of origin—Identifies a set of sites and the corresponding route as having come from one of the sites in that set.
- Site of origin—Uniquely identifies the set of routes that a PE router learned from a particular site. This attribute ensures that a route learned from a particular site through a particular PE-CE connection is not distributed back to the site through a different PE-CE connection. It is particularly useful if you are using BGP as the routing protocol between the PE and CE routers and if different sites in the VPN have been assigned the same autonomous system (AS) numbers.

Figure 1: VPN Attributes and Route Distribution

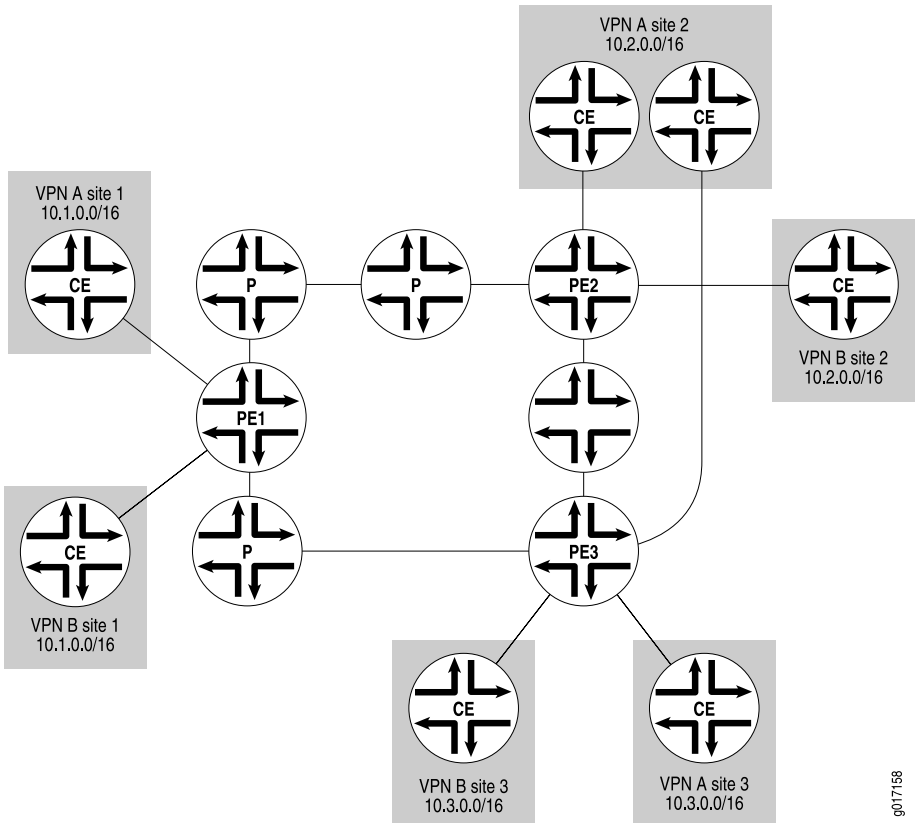


VPN-IPv4 Addresses and Route Distinguishers

Because Layer 3 VPNs connect private networks—which can use either public addresses or private addresses, as defined in RFC 1918 (*Address Allocation for Private Internets*)—over the public Internet infrastructure, when the private networks use private addresses, the addresses might overlap with the addresses of another private network.

Figure 2 on page 6 illustrates how private addresses of different private networks can overlap. Here, sites within VPN A and VPN B use the address spaces **10.1.0.0/16**, **10.2.0.0/16**, and **10.3.0.0/16** for their private networks.

Figure 2: Overlapping Addresses Among Different VPNs



To avoid overlapping private addresses, you can configure the network devices to use public addresses instead of private addresses. However, this is a large and complex undertaking. The solution provided in RFC 4364 uses the existing private network numbers to create a new address that is unambiguous. The new address is part of the VPN-IPv4 address family, which is a BGP address family added as an extension to the BGP protocol. In VPN-IPv4 addresses, a value that identifies the VPN, called a route distinguisher, is prefixed to the private IPv4 address, providing an address that uniquely identifies a private IPv4 address.

Only the PE routers need to support the VPN-IPv4 address extension to BGP. When an ingress PE router receives an IPv4 route from a device within a VPN, it converts it into a VPN-IPv4 route by adding the route distinguisher prefix to the route. The VPN-IPv4 addresses are used only for routes exchanged between PE routers. When an egress PE router receives a VPN-IPv4 route, it converts the VPN-IPv4 route back to an IPv4 route by removing the route distinguisher before announcing the route to its connected CE routers.

VPN-IPv4 addresses have the following format:

- Route distinguisher is a 6-byte value that you can specify in one of the following formats:
  - ***as-number*:*number***, where ***as-number*** is an AS number (a 2-byte value) and ***number*** is any 4-byte value. The AS number can be in the range 1 through 65,535. We recommend that you use an Internet Assigned Numbers Authority (IANA)-assigned,

nonprivate AS number, preferably the Internet service provider's (ISP's) own or the customer's own AS number.

- ***ip-address*:*number***, where ***ip-address*** is an IP address (a 4-byte value) and ***number*** is any 2-byte value. The IP address can be any globally unique unicast address. We recommend that you use the address that you configure in the **router-id** statement, which is a nonprivate address in your assigned prefix range.

- IPv4 address—4-byte address of a device within the VPN.

Figure 2 on page 6 illustrates how the AS number can be used in the route distinguisher. Suppose that VPN A is in AS **65535** and that VPN B is in AS **666** (both these AS numbers belong to the ISP), and suppose that the route distinguisher for Site 2 in VPN A is **65535:02** and that the route distinguisher for Site 2 in VPN B is **666:02**. When Router PE2 receives a route from the CE router in VPN A, it converts it from its IP address of **10.2.0.0** to a VPN-IPv4 address of **65535:02:10.2.0.0**. When the PE router receives a route from VPN B, which uses the same address space as VPN A, it converts it to a VPN-IPv4 address of **666:02:10.2.0.0**.

If the IP address is used in the route distinguisher, suppose Router PE2's IP address is **172.168.0.1**. When the PE router receives a route from VPN A, it converts it to a VPN-IPv4 address of **172.168.0.1:0:10.2.0.0/16**, and it converts a route from VPN B to **172.168.0.0:1:10.2.0.0/16**.

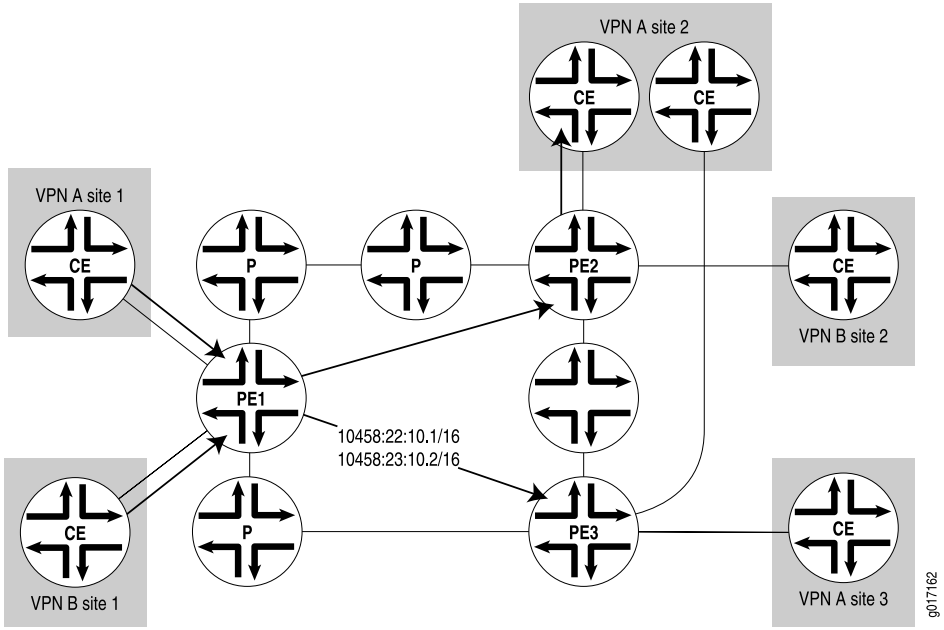
Route distinguishers are used only among PE routers to IPv4 addresses from different VPNs. The ingress PE router creates a route distinguisher and converts IPv4 routes received from CE routers into VPN-IPv4 addresses. The egress PE routers convert VPN-IPv4 routes into IPv4 routes before announcing them to the CE router.

Because VPN-IPv4 addresses are a type of BGP address, you must configure IBGP sessions between pairs of PE routers so that the PE routers can distribute VPN-IPv4 routes within the provider's core network. (All PE routers are assumed to be within the same AS.)

You define BGP communities to constrain the distribution of routes among the PE routers. Defining BGP communities does not, by itself, distinguish IPv4 addresses.

Figure 3 on page 8 illustrates how Router PE1 adds the route distinguisher **10458:22:10.1/16** to routes received from the CE router at Site 1 in VPN A and forwards these routes to the other two PE routers. Similarly, Router PE1 adds the route distinguisher **10458:23:10.2/16** to routes received by the CE router at Site 1 in VPN B and forwards these routes to the other PE routers.

Figure 3: Route Distinguishers



IPv6 Layer 3 VPNs

The interfaces between the PE and CE routers of a Layer 3 VPN can be configured to carry IP version 6 (IPv6) traffic. IP allows numerous nodes on different networks to interoperate seamlessly. IPv4 is currently used in intranets and private networks, as well as the Internet. IPv6 is the successor to IPv4, and is based for the most part on IPv4.

In the Juniper Networks implementation of IPv6, the service provider implements an MPLS-enabled IPv4 backbone to provide VPN service for IPv6 customers. The PE routers have both IPv4 and IPv6 capabilities. They maintain IPv6 VPN routing and forwarding (VRF) tables for their IPv6 sites and encapsulate IPv6 traffic in MPLS frames that are then sent into the MPLS core network.

IPv6 for Layer 3 VPNs is supported for BGP and for static routes.

IPv6 over Layer 3 VPNs is described in RFC 4659, *BGP-MPLS IP Virtual Private Network (VPN) Extension for IPv6 VPN*.

For more information about IPv6, see the *Junos OS Routing Protocols Configuration Guide*.

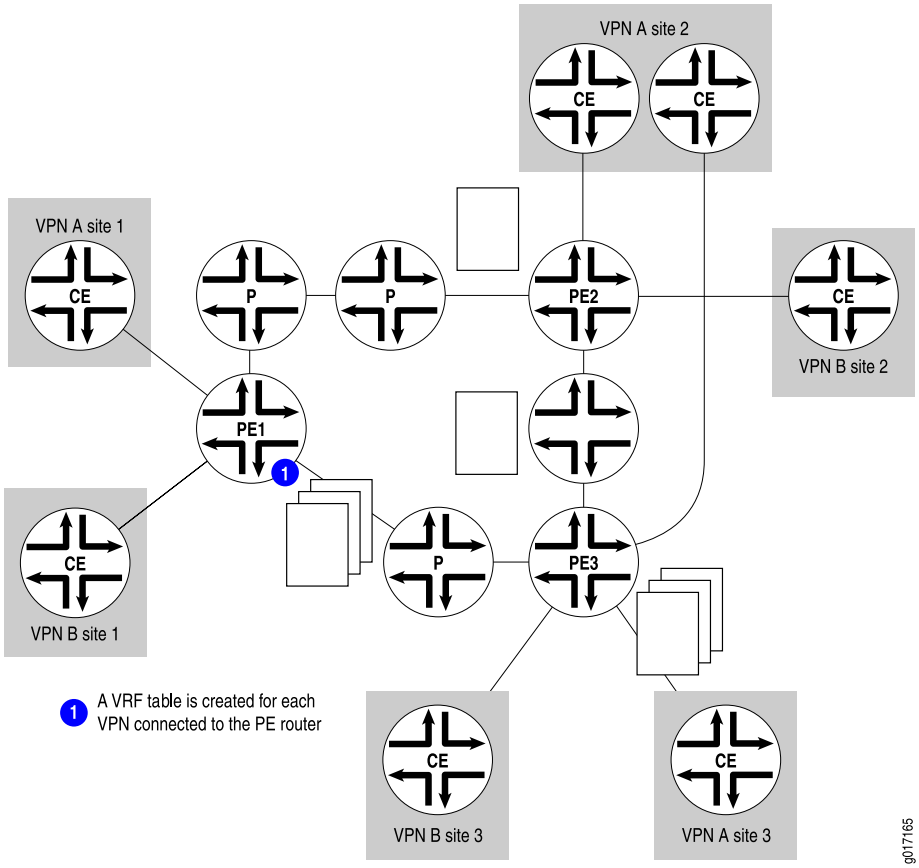
VPN Routing and Forwarding Tables

To separate a VPN's routes from routes in the public Internet or those in other VPNs, the PE router creates a separate routing table for each VPN, called a VPN routing and forwarding (VRF) table. The PE router creates one VRF table for each VPN that has a

connection to a CE router. Any customer or site that belongs to the VPN can access only the routes in the VRF tables for that VPN.

Figure 4 on page 9 illustrates the VRF tables that are created on the PE routers. The three PE routers have connections to CE routers that are in two different VPNs, so each PE router creates two VRF tables, one for each VPN.

Figure 4: VRF Tables



Each VRF table is populated from routes received from directly connected CE sites associated with that VRF routing instance and from routes received from other PE routers that passed BGP community filtering and are in the same VPN.

Each PE router also maintains one global routing table (**inet.0**) to reach other routers in and outside the provider's core network.

Each customer connection (that is, each logical interface) is associated with one VRF table. Only the VRF table associated with a customer site is consulted for packets from that site.

You can configure the router so that if a next hop to a destination is not found in the VRF table, the router performs a lookup in the global routing table, which is used for Internet access.

The Junos OS uses the following routing tables for VPNs:

- **bgp.l3vpn.0**—Stores all VPN-IPv4 unicast routes received from other PE routers. (This table does not store routes received from directly connected CE routers.) This table is present only on PE routers.

  When a PE router receives a route from another PE router, it places the route into its **bgp.l3vpn.0** routing table. The route is resolved using the information in the **inet.3** routing table. The resultant route is converted into IPv4 format and redistributed to all ***routing-instance-name*.inet.0** routing tables on the PE router if it matches the VRF import policy.

  The **bgp.l3vpn.0** table is also used to resolve routes over the MPLS tunnels that connect the PE routers. These routes are stored in the **inet.3** routing table. PE-to-PE router connectivity must exist in **inet.3** (not just in **inet.0**) for VPN routes to be resolved properly.

  When a router is advertising non-local VPN-IPv4 unicast routes and the router is a route reflector or is performing external peering, the VPN-IPv4 unicast routes are automatically exported into the VPN routing table (**bgp.l3vpn.0**). This enables the router to perform path selection and advertise from the **bgp.l3vpn.0** routing table.

  To determine whether to add a route to the **bgp.l3vpn.0** routing table, the Junos OS checks it against the VRF instance import policies for all the VPNs configured on the PE router. If the VPN-IPv4 route matches one of the policies, it is added to the **bgp.l3vpn.0** routing table. To display the routes in the **bgp.l3vpn.0** routing table, use the **show route table bgp.l3vpn.0** command.

- ***routing-instance-name*.inet.0**—Stores all unicast IPv4 routes received from directly connected CE routers in a routing instance (that is, in a single VPN) and all explicitly configured static routes in the routing instance. This is the VRF table and is present only on PE routers. For example, for a routing instance named **VPN-A**, the routing table for that instance is named **VPN-A.inet.0**.

  When a CE router advertises to a PE router, the PE router places the route into the corresponding ***routing-instance-name*.inet.0** routing table and advertises the route to other PE routers if it passes a VRF export policy. Among other things, this policy tags the route with the route distinguisher (route target) that corresponds to the VPN site to which the CE belongs. A label is also allocated and distributed with the route. The **bgp.l3vpn.0** routing table is not involved in this process.

  The ***routing-instance-name*.inet.0** table also stores routes announced by a remote PE router that match the VRF import policy for that VPN. The remote PE router redistributed these routes from its **bgp.l3vpn.0** table.

  Routes are not redistributed from the ***routing-instance-name*.inet.0** table to the **bgp.l3vpn.0** table; they are directly advertised to other PE routers.

  For each ***routing-instance-name*.inet.0** routing table, one forwarding table is maintained in the router's Packet Forwarding Engine. This table is maintained in addition to the forwarding tables that correspond to the router's **inet.0** and **mpls.0** routing tables. As with the **inet.0** and **mpls.0** routing tables, the best routes from the ***routing-instance-name*.inet.0** routing table are placed into the forwarding table.

To display the routes in the ***routing-instance-name*.inet.0** table, use the **show route table *routing-instance-name*.inet.0** command.

- **inet.3**—Stores all MPLS routes learned from LDP and RSVP signaling done for VPN traffic. The routing table stores the MPLS routes only if the **traffic-engineering bgp-igp** option is not enabled.

  For VPN routes to be resolved properly, the **inet.3** table must contain routes to all the PE routers in the VPN.

  To display the routes in the **inet.3** table, use the **show route table inet.3** command.

- **inet.0**—Stores routes learned by the IBGP sessions between the PE routers. To provide Internet access to the VPN sites, configure the ***routing-instance-name*.inet.0** routing table to contain a default route to the **inet.0** routing table.

  To display the routes in the **inet.0** table, use the **show route table inet.0** command.

The following routing policies, which are defined in VRF import and export statements, are specific to VRF tables.

- Import policy—Applied to VPN-IPv4 routes learned from another PE router to determine whether the route should be added to the PE router's **bgp.l3vpn.0** routing table. Each routing instance on a PE router has a VRF import policy.
- Export policy—Applied to VPN-IPv4 routes that are announced to other PE routers. The VPN-IPv4 routes are IPv4 routes that have been announced by locally connected CE routers.

VPN route processing differs from normal BGP route processing in one way. In BGP, routes are accepted if they are not explicitly rejected by import policy. However, because many more VPN routes are expected, the Junos OS does not accept (and hence store) VPN routes unless the route matches at least one VRF import policy. If no VRF import policy explicitly accepts the route, it is discarded and not even stored in the **bgp.l3vpn.0** table. As a result, if a VPN change occurs on a PE router—such as adding a new VRF table or changing a VRF import policy—the PE router sends a BGP route refresh message to the other PE routers (or to the route reflector if this is part of the VPN topology) to retrieve all VPN routes so they can be reevaluated to determine whether they should be kept or discarded.

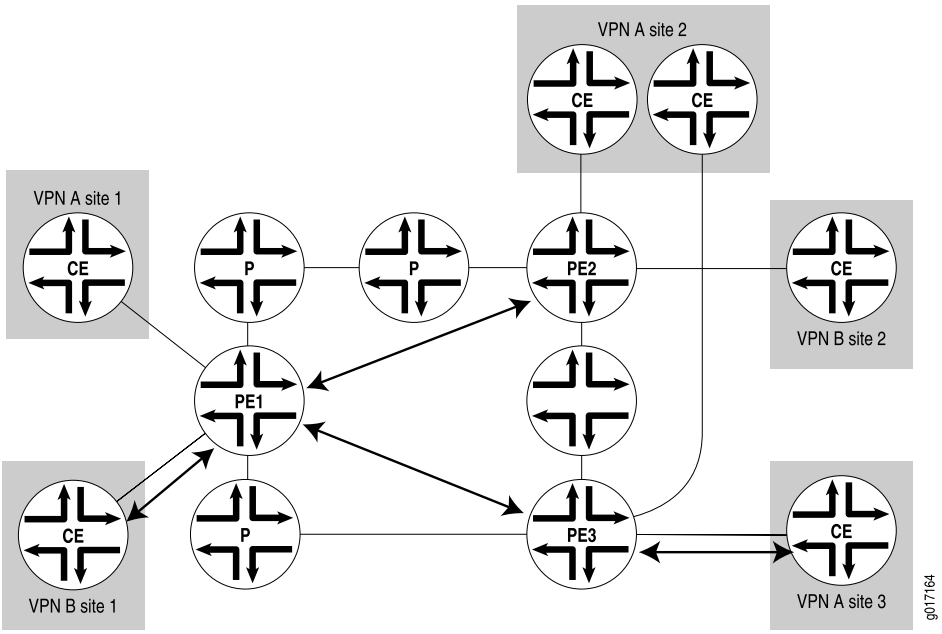
Related Documentation

- IGP Shortcuts and VPNs

Route Distribution Within a Layer 3 VPN

Within a VPN, the distribution of VPN-IPv4 routes occurs between the PE and CE routers and between the PE routers (see Figure 5 on page 12).

Figure 5: Route Distribution Within a VPN



This section discusses the following topics:

- Distribution of Routes from CE to PE Routers on page 12
- Distribution of Routes Between PE Routers on page 13
- Distribution of Routes from PE to CE Routers on page 14

Distribution of Routes from CE to PE Routers

A CE router announces its routes to the directly connected PE router. The announced routes are in IPv4 format. The PE router places the routes into the VRF table for the VPN. In the Junos OS, this is the ***routing-instance-name*.inet.0** routing table, where ***routing-instance-name*** is the configured name of the VPN.

The connection between the CE and PE routers can be a remote connection (a WAN connection) or a direct connection (such as a Frame Relay or Ethernet connection).

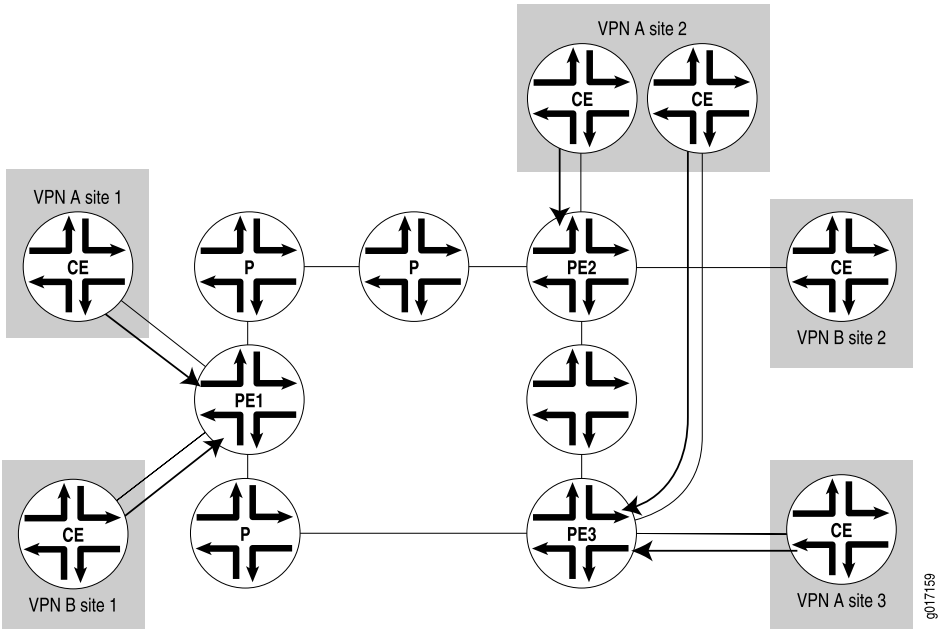
CE routers can communicate with PE routers using one of the following:

- OSPF
- RIP

- BGP
- Static route

Figure 6 on page 13 illustrates how routes are distributed from CE routers to PE routers. Router PE1 is connected to two CE routers that are in different VPNs. Therefore, it creates two VRF tables, one for each VPN. The CE routers announce IPv4 routes. The PE router installs these routes into two different VRF tables, one for each VPN. Similarly, Router PE2 creates two VRF tables into which routes are installed from the two directly connected CE routers. Router PE3 creates one VRF table because it is directly connected to only one VPN.

Figure 6: Distribution of Routes from CE Routers to PE Routers



Distribution of Routes Between PE Routers

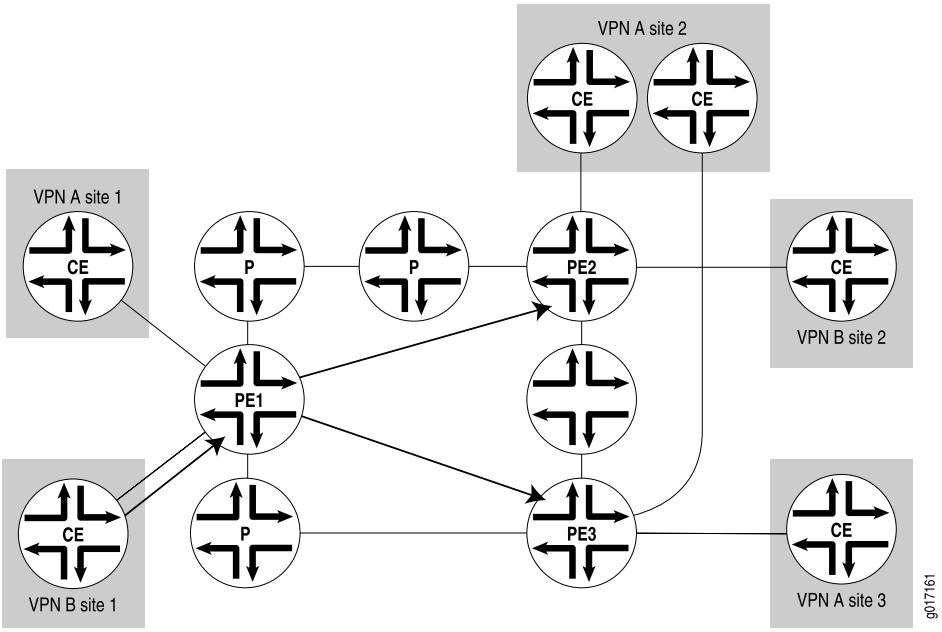
When one PE router receives routes advertised from a directly connected CE router, it checks the received route against the VRF export policy for that VPN. If it matches, the route is converted to VPN-IPv4 format—that is, the route distinguisher (route target) is added to the route. The PE router then announces the route in VPN-IPv4 format to the remote PE routers. The routes are distributed using IBGP sessions, which are configured in the provider's core network. If the route does not match, it is not exported to other PE routers, but can still be used locally for routing, for example, if two CE routers in the same VPN are directly connected to the same PE router.

The remote PE router places the route into its **bgp.l3vpn.0** table if the route passes the import policy on the IBGP session between the PE routers. At the same time, it checks the route against the VRF import policy for the VPN. If it matches, the route distinguisher

is removed from the route and it is placed into the VRF table (the ***routing-instance-name*.inet.0** table) in IPv4 format.

Figure 7 on page 14 illustrates how Router PE1 distributes routes to the other PE routers in the provider's core network. Router PE2 and Router PE3 each have VRF import policies that they use to determine whether to accept routes received over the IBGP sessions and install them in their VRF tables.

Figure 7: Distribution of Routes Between PE Routers



Distribution of Routes from PE to CE Routers

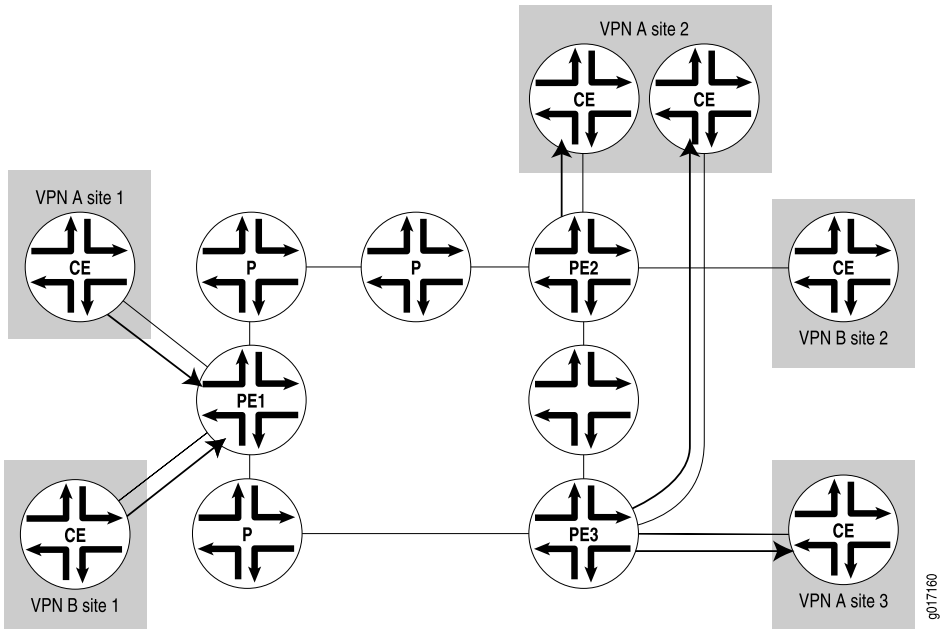
The remote PE router announces the routes in its VRF tables, which are in IPv4 format, to its directly connected CE routers.

PE routers can communicate with CE routers using one of the following routing protocols:

- OSPF
- RIP
- BGP
- Static route

Figure 8 on page 15 illustrates how the three PE routers announce their routes to their connected CE routers.

Figure 8: Distribution of Routes from PE Routers to CE Routers



Forwarding Across the Provider's Core Network

The PE routers in the provider's core network are the only routers that are configured to support VPNs and hence are the only routers to have information about the VPNs. From the point of view of VPN functionality, the provider (P) routers in the core—those P routers that are not directly connected to CE routers—are merely routers along the tunnel between the ingress and egress PE routers.

The tunnels can be either LDP or MPLS. Any P routers along the tunnel must support the protocol used for the tunnel, either LDP or MPLS.

When PE-router-to-PE router forwarding is tunneled over MPLS label-switched paths (LSPs), the MPLS packets have a two-level label stack (see Figure 9 on page 16):

- Outer label—Label assigned to the address of the BGP next hop by the IGP next hop
- Inner label—Label that the BGP next hop assigned for the packet's destination address

Figure 9: Using MPLS LSPs to Tunnel Between PE Routers

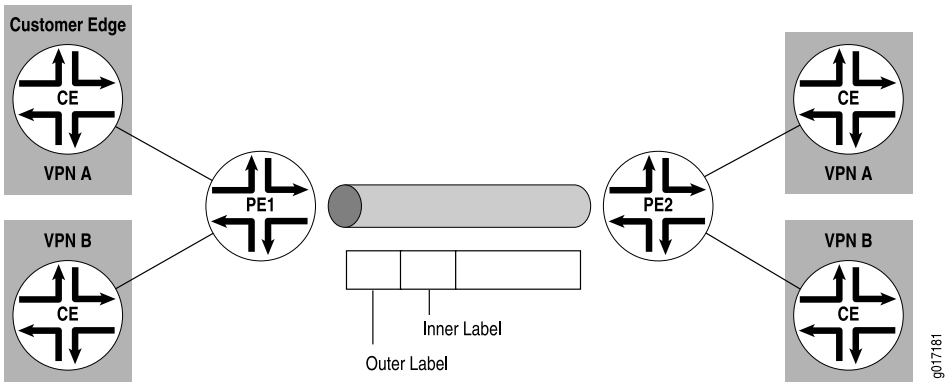
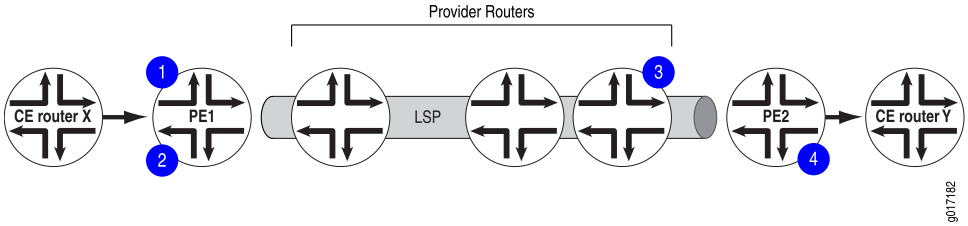


Figure 10 on page 16 illustrates how the labels are assigned and removed:

1. When CE Router X forwards a packet to Router PE1 with a destination of CE Router Y, the PE route identifies the BGP next hop to Router Y and assigns a label that corresponds to the BGP next hop and identifies the destination CE router. This label is the inner label.
2. Router PE1 then identifies the IGP route to the BGP next hop and assigns a second label that corresponds to the LSP of the BGP next hop. This label is the outer label.
3. The inner label remains the same as the packet traverses the LSP tunnel. The outer label is swapped at each hop along the LSP and is then popped by the penultimate hop router (the third P router).
4. Router PE2 pops the inner label from the route and forwards the packet to Router Y.

Figure 10: Label Stack



Routing Instances for VPNs

To implement Layer 3 VPNs in the JUNOS Software, you configure one routing instance for each VPN. You configure the routing instances on PE routers only. Each VPN routing instance consists of the following components:

- VRF table—On each PE router, you configure one VRF table for each VPN.
- Set of interfaces that use the VRF table—The logical interface to each directly connected CE router must be associated with a VRF table. You can associate more than one interface with the same VRF table if more than one CE router in a VPN is directly connected to the PE router.

- Policy rules—These control the import of routes into and the export of routes from the VRF table.
- One or more routing protocols that install routes from CE routers into the VRF table—You can use the BGP, OSPF, and RIP routing protocols, and you can use static routes.

Multicast over Layer 3 VPNs

You can configure multicast routing over a network running a Layer 3 VPN that complies with RFC 4364. This section describes this type of network application and includes these topics:

Multicast over Layer 3 VPNs Overview

In the unicast environment for Layer 3 VPNs, all VPN state information is contained within the PE routers. However, with multicast for Layer 3 VPNs, Protocol Independent Multicast (PIM) adjacencies are established in one of the following ways:

- You can set PIM adjacencies between the CE router and the PE router through a VRF instance at the **[edit routing-instances *instance-name* protocols pim]** hierarchy level. You must include the **group-address** statement for the provider tunnel, specifying a multicast group. The rendezvous point (RP) listed within the VRF-instance is the VPN customer RP (C-RP).
- You can also set the master PIM instance and the PE's IGP neighbors by configuring statements at the [edit protocols pim] hierarchy level. You must add the multicast group specified in the VRF instance to the master PIM instance. The set of master PIM adjacencies throughout the service provider network makes up the forwarding path that becomes an RP tree rooted at the service provider RP (SP-RP). Therefore, P routers within the provider core must maintain multicast state information for the VPNs.

For this to work properly, you need two types of RP routers for each VPN:

- A C-RP—An RP router located somewhere within the VPN (can be either a service provider router or a customer router).
- An SP-RP—An RP router located within the service provider network.

NOTE: A PE router can act as the SP-RP and the C-RP. Moving these multicast configuration tasks to service provider routers helps to simplify the multicast Layer 3 VPN configuration process for customers. However, configuration of both SP-RP and VPN C-RP on the same PE router is not supported.

To configure multicast over a Layer 3 VPN, you must install a Tunnel Services Physical Interface Card (PIC) on the following devices:

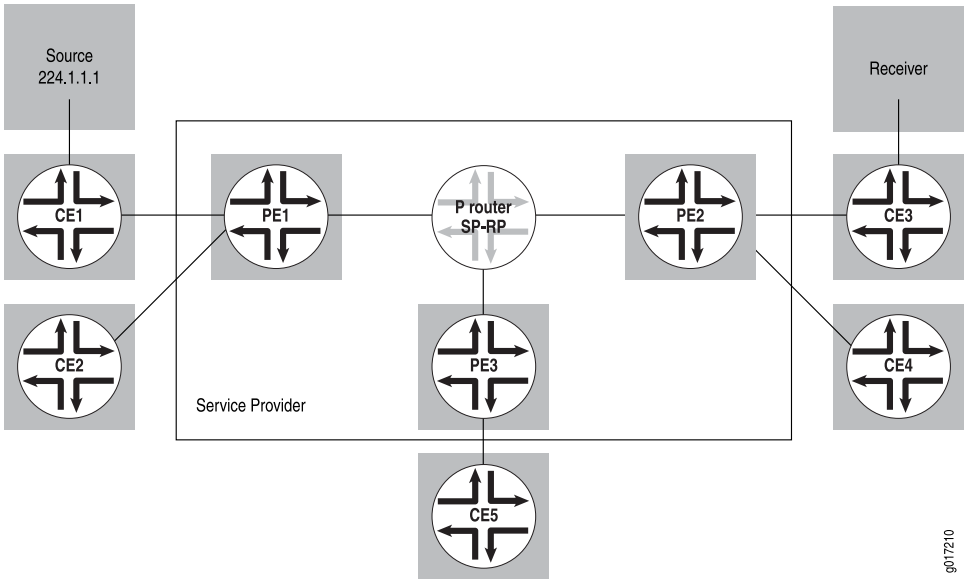
- P routers acting as RPs
- PE routers configured to run multicast routing
- CE routers acting as designated routers or as VPN-RPs

For more information about running multicast over Layer 3 VPNs, see the following documents:

- Internet draft draft-rosen-vpn-mcast-02.txt, *Multicast in MPLS/BGP VPNs*
- *Junos OS Multicast Protocols Configuration Guide*

The sections that follow describe the operation of a multicast VPN. Figure 11 on page 18 illustrates the network topology used.

Figure 11: Multicast Topology Overview



Sending PIM Hello Messages to the PE Routers

The first step in initializing multicast over a Layer 3 VPN is the distribution of a PIM Hello message from a PE router (called PE3 in this section) to all the other PE routers on which PIM is configured.

You configure PIM on the Layer 3 VPN routing instance on the PE3 router. If a Tunnel Services PIC is installed in the routing platform, a multicast interface is created. This interface is used to communicate between the PIM instance within the VRF routing instance and the master PIM instance.

The following occurs when a PIM Hello message is sent to the PE routers:

1. A PIM Hello message is sent from the VRF routing instance over the multicast interface. A generic routing encapsulation (GRE) header is prepended to the PIM Hello message. The header message includes the VPN group address and the loopback address of the PE3 router.
2. A PIM register header is prepended to the Hello message as the packet is looped through the PIM encapsulation interface. This header contains the destination address of the SP-RP and the loopback address of the PE3 router.
3. The packet is sent to the SP-RP.
4. The SP-RP removes the top header from the packet and sends the remaining GRE-encapsulated Hello message to all the PE routers.
5. The master PIM instance on each PE router handles the GRE encapsulated packet. Because the VPN group address is contained in the packet, the master instance removes the GRE header from the packet and sends the Hello message, which contains the proper VPN group address within the VRF routing instance, over the multicast interface.

Sending PIM Join Messages to the PE Routers

To receive a multicast broadcast from a multicast network, a CE router must send a PIM Join message to the C-RP. The process described in this section refers to Figure 11 on page 18.

The CE5 router needs to receive a multicast broadcast from multicast source **224.1.1.1**. To receive the broadcast, it sends a PIM Join message to the C-RP (the PE3 router):

1. The PIM Join message is sent through the multicast interface, and a GRE header is prepended to the message. The GRE header contains the VPN group ID and the loopback address of the PE3 router.
2. The PIM Join message is then sent through the PIM encapsulation interface and a register header is prepended to the packet. The register header contains the IP address of the SP-RP and the loopback address of the PE3 router.
3. The PIM Join message is sent to the SP-RP by means of unicast routing.
4. On the SP-RP, the register header is stripped off (the GRE header remains) and the packet is sent to all the PE routers.
5. The PE2 router receives the packet, and because the link to the C-RP is through the PE2 router, it sends the packet through the multicast interface to remove the GRE header.
6. Finally, the PIM Join message is sent to the C-RP.

Receiving the Multicast Transmission

The steps that follow outline how a multicast transmission is propagated across the network:

1. The multicast source connected to the CE1 router sends the packet to group **224.1.1.1** (the VPN group address). The packet is encapsulated into a PIM register.
2. Because this packet already includes the PIM header, it is forwarded by means of unicast routing to the C-RP over the Layer 3 VPN.
3. The C-RP removes the packet and sends it out the downstream interfaces (which include the interface back to the CE3 router). The CE3 router also forwards this to the PE3 router.
4. The packet is sent through the multicast interface on the PE2 router; in the process, the GRE header is prepended to the packet.
5. Next, the packet is sent through the PIM encapsulation interface, where the register header is prepended to the data packet.
6. The packet is then forwarded to the SP-RP, which removes the register header, leaves the GRE header intact, and sends the packet to the PE routers.
7. PE routers remove the GRE header and forward the packet to the CE routers that requested the multicast broadcast by sending the PIM Join message.

NOTE: PE routers that have not received requests for multicast broadcasts from their connected CE routers still receive packets for the broadcast. These PE routers drop the packets as they are received.

CHAPTER 2

Introduction to Configuring Layer 3 VPNs

Configuring a VPN Tunnel for VRF Table Lookup

You can configure a VPN tunnel to facilitate VRF table lookup based on MPLS labels. You might want to enable this functionality to forward traffic on a PE-router-to-CE-device interface in a shared medium, where the CE device is a Layer 2 switch without IP capabilities (for example, a metro Ethernet switch), or to perform egress filtering at the egress PE router.

For more information about VPN tunnels and VT interfaces, see the *Junos OS Services Interfaces Configuration Guide*.

PART 2

Configuration

CHAPTER 3

Configuring Layer 3 VPNs

Introduction to Configuring Layer 3 VPNs

To configure Layer 3 virtual private network (VPN) functionality, you must enable VPN support on the provider edge (PE) router. You must also configure any provider (P) routers that service the VPN, and you must configure the customer edge (CE) routers so that their routes are distributed into the VPN.

To configure Layer 3 VPNs, you include the following statements:

```
description text;
instance-type vrf;
interface interface-name;
protocols {
    bgp {
        group group-name {
            peer-as as-number;
            neighbor ip-address;
        }
        multihop ttl-value;
    }
    (ospf | ospf3) {
        area area {
            interface interface-name;
        }
        domain-id domain-id;
        domain-vpn-tag number;
        sham-link  {
            local address;
        }
        sham-link-remote address <metric number>;
    }
    rip {
        rip-configuration;
    }
}
route-distinguisher (as-number:id | ip-address:id);
router-id address;
routing-options {
    autonomous-system autonomous-system {
        independent-domain;
        loops number;
    }
    forwarding-table {
        export [ policy-names ];
    }
    interface-routes {
        rib-group group-name;
    }
    martians {
        destination-prefix match-type <allow>;
    }
    maximum-paths {
        path-limit;
        log-interval interval;
```

```
        log-only;
        threshold percentage;
      }
      maximum-prefixes {
        prefix-limit;
        log-interval interval;
        log-only;
        threshold percentage;
      }
      multipath {
        vpn-unequal-cost;
      }
      options {
        syslog (level level | upto level);
      }
      rib routing-table-name {
        martians {
          destination-prefix match-type <allow>;
        }
        multipath {
          vpn-unequal-cost;
        }
        static {
          defaults {
            static-options;
          }
          route destination-prefix {
            next-hop [next-hops];
            static-options;
          }
        }
      }
    }
    static {
      defaults {
        static-options;
      }
      route destination-prefix {
        policy [ policy-names ];
        static-options;
      }
    }
    vrf-advertise-selective {
      family {
        inet-mvpn;
        inet6-mvpn;
      }
    }
    vrf-export [ policy-names ];
    vrf-import [ policy-names ];
    vrf-target (community | export community-name | import community-name);
    vrf-table-label;
```

You can include these statements at the following hierarchy levels:

- **[edit routing-instances *routing-instance-name*]**

- **[edit logical-systems *logical-system-name* routing-instances *routing-instance-name*]**

For Layer 3 VPNs, only some of the statements in the **[edit routing-instances]** hierarchy are valid. For the full hierarchy, see the *Junos OS Routing Protocols Configuration Guide*.

In addition to these statements, you must enable a signaling protocol, IBGP sessions between the PE routers, and an interior gateway protocol (IGP) on the PE and P routers.

By default, Layer 3 VPNs are disabled.

Many of the configuration procedures for Layer 3 VPNs are common to all types of VPNs.

Related Documentation

- Centralized Internet Access on page 221
- Configuring Hub-and-Spoke VPN Topologies: One Interface on page 104
- Configuring Hub-and-Spoke VPN Topologies: Two Interfaces on page 116
- Configuring Overlapping VPNs Using Automatic Route Export on page 166
- Configuring Overlapping VPNs Using Routing Table Groups on page 155
- Configuring a Full-Mesh VPN Topology with Route Reflectors on page 103
- Configuring a GRE Tunnel Interface Between PE Routers on page 170
- Configuring a GRE Tunnel Interface Between a PE and CE Router on page 176
- Configuring a Simple Full-Mesh VPN Topology on page 89
- Configuring an Application-Based Layer 3 VPN Topology on page 145
- Configuring an ES Tunnel Interface Between a PE and CE Router on page 179
- Configuring an LDP-over-RSVP VPN Topology on page 131
- Configuring an OSPF Domain ID for a Layer 3 VPN on page 150
- Distributed Internet Access on page 198
- Routing Internet Traffic Through a Separate NAT Device on page 214
- Routing VPN and Internet Traffic Through Different Interfaces on page 199
- Routing VPN and Internet Traffic Through the Same Interface Bidirectionally (VPN Has Private Addresses) on page 210
- Routing VPN and Internet Traffic Through the Same Interface Bidirectionally (VPN Has Public Addresses) on page 206
- Routing VPN and Outgoing Internet Traffic Through the Same Interface and Routing Return Internet Traffic Through a Different Interface on page 205
- Setting the Forwarding Class of the Ping Packets

Configuring Routing Between PE and CE Routers in Layer 3 VPNs

For the PE router to distribute VPN-related routes to and from connected CE routers, you must configure routing within the VPN routing instance. You can configure a routing

protocol—BGP, OSPF, or RIP—or you can configure static routing. For the connection to each CE router, you can configure only one type of routing.

The following sections explain how to configure VPN routing between the PE and CE routers:

Configuring BGP Between the PE and CE Routers

To configure BGP as the routing protocol between the PE and the CE routers, include the **bgp** statement:

```
bgp {
  group group-name {
    peer-as as-number;
    neighbor ip-address;
  }
}
```

You can include this statement at the following hierarchy levels:

- **[edit routing-instances *routing-instance-name* protocols]**
- **[edit logical-systems *logical-system-name* routing-instances *routing-instance-name* protocols]**

  Please be aware of the following limitations regarding configuring BGP for routing instances:

  - In a VRF routing instance, do not configure the local autonomous system (AS) number using an AS number that is already in use by a remote BGP peer in a separate VRF routing instance. Doing so creates an autonomous system loop where all the routes received from this remote BGP peer are hidden.

    You configure the local AS number using either the **autonomous-system** statement at the **[edit routing-instances *routing-instance-name* routing-options]** hierarchy level or the **local-as** statement at any of the following hierarchy levels:

    - **[edit routing-instances *routing-instance-name* protocols bgp]**
    - **[edit routing-instances *routing-instance-name* protocols bgp group *group-name*]**
    - **[edit routing-instances *routing-instance-name* protocols bgp group *group-name* neighbor *address*]**

You configure the AS number for a BGP peer using the **peer-as** statement at the **[edit routing-instances *routing-instance-name* protocols bgp group *group-name*]** hierarchy level.

Configuring OSPF Between the PE and CE Routers

You can configure OSPF (version 2 or version 3) to distribute VPN-related routes between PE and CE routers.

The following sections describe how to configure OSPF as a routing protocol between the PE and the CE routers:

Configuring OSPF Version 2 Between the PE and CE Routers

To configure OSPF version 2 as the routing protocol between a PE and CE router, include the **ospf** statement:

```
ospf {
  area area {
    interface interface-name;
  }
}
```

You can include this statement at the following hierarchy levels:

- **[edit routing-instances *routing-instance-name* protocols]**
- **[edit logical-systems *logical-system-name* routing-instances *routing-instance-name* protocols]**

Configuring OSPF Version 3 Between the PE and CE Routers

To configure OSPF version 3 as the routing protocol between a PE and CE router, include the **ospf3** statement:

```
ospf3 {
  area area {
    interface interface-name;
  }
}
```

You can include this statement at the following hierarchy levels:

- **[edit routing-instances *routing-instance-name* protocols]**
- **[edit logical-systems *logical-system-name* routing-instances *routing-instance-name* protocols]**

Configuring OSPF Sham Links for Layer 3 VPNs

When you configure OSPF between the PE and CE routers of a Layer 3 VPN, you can also configure OSPF sham links to compensate for issues related to OSPF intra-area links.

The following sections describe OSPF sham links and how to configure them:

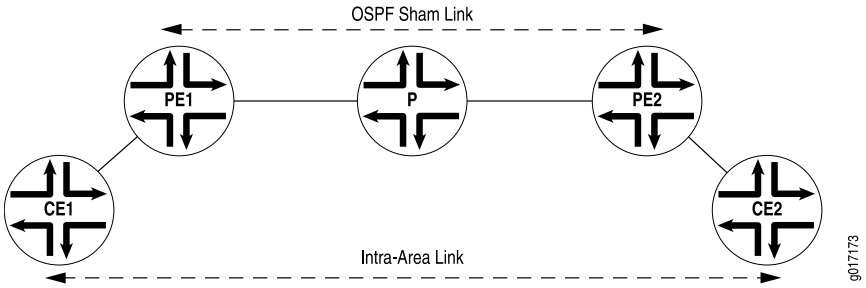
OSPF Sham Links Overview

Figure 12 on page 31 provides an illustration of when you might configure an OSPF sham link. Router CE1 and Router CE2 are located in the same OSPF area. These CE routers are linked together by a Layer 3 VPN over Router PE1 and Router PE2. In addition, Router CE1 and Router CE2 are connected by an intra-area link used as a backup.

OSPF treats the link through the Layer 3 VPN as an interarea link. By default, OSPF prefers intra-area links to interarea links, so OSPF selects the backup intra-area link as the active path. This is not acceptable in configurations where the intra-area link is not the expected primary path for traffic between the CE routers.

An OSPF sham link is also an intra-area link, except that it is configured between the PE routers as shown in Figure 12 on page 31. You can configure the metric for the sham link to ensure that the path over the Layer 3 VPN is preferred to a backup path over an intra-area link connecting the CE routers.

Figure 12: OSPF Sham Link



You should configure an OSPF sham link under the following circumstances:

- Two CE routers are linked together by a Layer 3 VPN.
- These CE routers are in the same OSPF area.
- An intra-area link is configured between the two CE routers.

If there is no intra-area link between the CE routers, you do not need to configure an OSPF sham link.

For more information about OSPF sham links, see the Internet draft draft-ietf-l3vpn-ospf-2547-01.txt, *OSPF as the PE/CE Protocol in BGP/MPLS VPNs*.

Configuring OSPF Sham Links

The sham link is an unnumbered point-to-point intra-area link and is advertised by means of a type 1 link-state advertisement (LSA). Sham links are valid only for routing instances and OSPF version 2.

Each sham link is identified by a combination of the local and remote sham link end-point address and the OSPF area to which it belongs. Sham links must be configured manually. You configure the sham link between two PE routers, both of which are within the same VRF routing instance.

You need to specify the address for the local end point of the sham link. This address is used as the source for the sham link packets and is also used by the remote PE router as the sham link remote end-point.

The OSPF sham link's local address must be specified with a loopback address for the local VPN. The route to this address must be propagated by BGP. Specify the address for the local end point using the **local** option of the **sham-link** statement:

```
sham-link {
  local address;
}
```

You can include this statement at the following hierarchy levels:

- **[edit routing-instances *routing-instance-name* protocols ospf]**
- **[edit logical-systems *logical-system-name* routing-instances *routing-instance-name* protocols ospf]**

The OSPF sham link's remote address must be specified with a loopback address for the remote VPN. The route to this address must be propagated by BGP. To specify the address for the remote end point, include the **sham-link-remote** statement:

```
sham-link-remote address <metric number>;
```

You can include this statement at the following hierarchy levels:

- **[edit routing-instances *routing-instance-name* protocols ospf area *area-id*]**
- **[edit logical-systems *logical-system-name* routing-instances *routing-instance-name* protocols ospf area *area-id*]**

Optionally, you can include the **metric** option to set a metric value for the remote end point. The metric value specifies the cost of using the link. Routes with lower total path metrics are preferred over those with higher path metrics.

You can configure a value from 1 through 65,535. The default value is 1.

OSPF Sham Links Example

This example shows how to enable OSPF sham links on a PE router.

The following is the loopback interface configuration on the PE router. The address configured is for the local end point of the OSPF sham link:

```
[edit]
interfaces {
  lo0 {
    unit 1 {
      family inet {
        address 10.1.1.1/32;
      }
    }
  }
}
```

The following is the routing instance configuration on the PE router, including the configuration for the OSPF sham link. The **sham-link local** statement is configured with the address for the local loopback interface:

```
[edit]
routing-instances {
  example-sham-links {
    instance-type vrf;
    interface e1-1/0/2.0;
    interface lo0.1;
    route-distinguisher 3:4;
    vrf-import vpn-red-import;
    vrf-export vpn-red-export;
    protocols {
      ospf {
        sham-link local 10.1.1.1;
        area 0.0.0.0 {
          sham-link-remote 10.2.2.2 metric 1;
          interface e1-1/0/2.0 metric 1;
        }
      }
    }
  }
}
```

Configuring an OSPF Domain ID

For most OSPF configurations involving Layer 3 VPNs, you do not need to configure an OSPF domain ID. However, for a Layer 3 VPN connecting multiple OSPF domains, configuring OSPF domain IDs can help you control LSA translation (for Type 3 and Type 5 LSAs) between the OSPF domains and back-door paths. Each VPN routing and forwarding (VRF) table in a PE router associated with an OSPF instance is configured with the same OSPF domain ID. The default OSPF domain ID is the null value 0.0.0.0. As shown in Table 3 on page 33, a route with a null domain ID is handled differently from a route without any domain ID at all.

Table 3: How a PE Router Redistributes and Advertises Routes

Route Received	Domain ID of the Route Received	Domain ID on the Receiving Router	Route Redistributed and Advertised As
Type 3 route	A.B.C.D	A.B.C.D	Type 3 LSA
Type 3 route	A.B.C.D	E.F.G.H	Type 5 LSA

Table 3: How a PE Router Redistributes and Advertises Routes *(continued)*

Route Received	Domain ID of the Route Received	Domain ID on the Receiving Router	Route Redistributed and Advertised As
Type 3 route	0.0.0.0	0.0.0.0	Type 3 LSA
Type 3 route	Null	0.0.0.0	Type 3 LSA
Type 3 route	Null	Null	Type 3 LSA
Type 3 route	0.0.0.0	Null	Type 3 LSA
Type 3 route	A.B.C.D	Null	Type 5 LSA
Type 3 route	Null	A.B.C.D	Type 5 LSA
Type 5 route	Not applicable	Not applicable	Type 5 LSA

You can configure an OSPF domain ID for both version 2 and version 3 of OSPF. The only difference in the configuration is that you include statements at the **[edit routing-instances *routing-instance-name* protocols ospf]** hierarchy level for OSPF version 2 and at the **[edit routing-instances *routing-instance-name* protocols ospf3]** hierarchy level for OSPF version 3. The configuration descriptions that follow present the OSPF version 2 statement only. However, the substatements are also valid for OSPF version 3.

To configure an OSPF domain ID, include the **domain-id** statement:

```
domain-id domain-Id;
```

You can include this statement at the following hierarchy levels:

- **[edit routing-instances *routing-instance-name* protocols ospf]**
- **[edit logical-systems *logical-system-name* routing-instances *routing-instance-name* protocols ospf]**

You can set a VPN tag for the OSPF external routes generated by the PE router to prevent looping. By default, this tag is automatically calculated and needs no configuration. However, you can configure the domain VPN tag for Type 5 LSAs explicitly by including the **domain-vpn-tag** statement:

```
domain-vpn-tag number;
```

You can include this statement at the following hierarchy levels:

- **[edit routing-instances *routing-instance-name* protocols ospf]**
- **[edit logical-systems *logical-system-name* routing-instances *routing-instance-name* protocols ospf]**

The range is 1 through 4,294,967,295 ($2^{32} - 1$). If you set VPN tags manually, you must set the same value for all PE routers in the VPN.

For an example of this type of configuration, see "Configuring an OSPF Domain ID for a Layer 3 VPN" on page 150.

Hub-and-Spoke Layer 3 VPNs and OSPF Domain IDs

The default behavior of an OSPF domain ID causes some problems for hub-and-spoke Layer 3 VPNs configured with OSPF between the hub PE router and the hub CE router when the routes are not aggregated. A hub-and-spoke configuration has a hub PE router with direct links to a hub CE router. The hub PE router receives Layer 3 BGP updates from the other remote spoke PE routers, and these are imported into the spoke routing instance. From the spoke routing instance, the OSPF LSAs are originated and sent to the hub CE router.

The hub CE router typically aggregates these routes, and then sends these newly originated LSAs back to the hub PE router. The hub PE router exports the BGP updates to the remote spoke PE routers containing the aggregated prefixes. However, if there are nonaggregated Type 3 summary LSAs or external LSAs, two issues arise with regard to how the hub PE router originates and sends LSAs to the hub CE router, and how the hub PE router processes LSAs received from the hub CE router:

- By default, all LSAs originated by the hub PE router in the spoke routing instance have the DN bit set. Also, all externally originated LSAs have the VPN route tag set. These settings help prevent routing loops. For Type 3 summary LSAs, routing loops are not a concern because the hub CE router, as an area border router (ABR), reoriginates the LSAs with the DN bit clear and sends them back to the hub PE router. However, the hub CE router does not reoriginate external LSAs, because they have an AS flooding scope.

  You can originate the external LSAs (before sending them to the hub CE router) with the DN bit clear and the VPN route tag set to 0 by altering the hub PE router's routing instance configuration. To clear the DN bit and set the VPN route tag to zero on external LSAs originated by a PE router, configure 0 for the **domain-vpn-tag** statement at the **[edit routing-instances *routing-instance-name* protocols ospf]** hierarchy level. You should include this configuration in the routing instance on the hub PE router facing the hub CE router where the LSAs are sent. When the hub CE router receives external LSAs from the hub PE router and then forwards them back to the hub PE router, the hub PE router can use the LSAs in its OSPF route calculation.

- When LSAs flooded by the hub CE router arrive at the hub PE router's routing instance, the hub PE router, acting as an ABR, does not consider these LSAs in its OSPF route calculations, even though the LSAs do not have the DN bits set and the external LSAs do not have a VPN route tag set. The LSAs are assumed to be from a disjoint backbone area.

  You can change the configuration of the PE router's routing instance to cause the PE router to act as a non-ABR by including the **disable** statement at the **[edit routing-instances *routing-instance-name* protocols ospf domain-id]** hierarchy level. You make this configuration change to the hub PE router that receives the LSAs from the hub CE router.

By making this configuration change, the PE router's routing instance acts as a non-ABR. The PE router then considers the LSAs arriving from the hub CE router as if they were coming from a contiguous nonbackbone area.

Configuring RIP Between the PE and CE Routers

For a Layer 3 VPN, you can configure RIP on the PE router to learn the routes of the CE router or to propagate the routes of the PE router to the CE router. RIP routes learned from neighbors configured at any **[edit routing-instances]** hierarchy level are added to the routing instance's **inet** table (***instance_name*.inet.0**).

To configure RIP as the routing protocol between the PE and the CE router, include the **rip** statement:

```
rip {
  group group-name {
    export policy-names;
    neighbor interface-name;
  }
}
```

You can include this statement at the following hierarchy levels:

- **[edit routing-instances *routing-instance-name* protocols]**
- **[edit logical-systems *logical-system-name* routing-instances *routing-instance-name* protocols]**

By default, RIP does not advertise the routes it receives. To advertise routes from a PE router to a CE router, you need to configure an export policy on the PE router for RIP. For information about how to define an export policy, see the *Junos OS Policy Framework Configuration Guide*.

To specify an export policy for RIP, include the **export** statement:

```
export [ policy-names ];
```

You can include this statement for RIP at the following hierarchy levels:

- **[edit routing-instances *routing-instance-name* protocols rip group *group-name*]**
- **[edit logical-systems *logical-system-name* routing-instances *routing-instance-name* protocols rip group *group-name*]**

To install routes learned from a RIP routing instance into multiple routing tables, include the **rib-group** and **group** statements:

```
rib-group inet group-name;
group group-name {
  neighbor interface-name;
}
```

You can include these statements at the following hierarchy levels:

- **[edit protocols]**
- **[edit routing-instances** *routing-instance-name* **protocols]**
- **[edit logical-systems** *logical-system-name* **protocols]**
- **[edit logical-systems** *logical-system-name* **routing-instances** *routing-instance-name* **protocols]**

To configure a routing table group, include the **rib-groups** statement:

```
rib-groups group-name;
```

You can include this statement at the following hierarchy levels:

- **[edit routing-options]**
- **[edit logical-systems** *logical-system-name* **routing-options]**

To add a routing table to a routing table group, include the **import-rib** statement. The first routing table name specified under the **import-rib** statement must be the name of the routing table you are configuring. For more information about how to configure routing tables and routing table groups, see the *Junos OS Routing Protocols Configuration Guide*.

```
import-rib [ group-names ];
```

You can include this statement at the following hierarchy levels:

- **[edit routing-options rib-groups** *group-name***]**
- **[edit logical-systems** *logical-system-name* **routing-options rib-groups** *group-name***]**

Configuring Static Routes Between the PE and CE Routers

You can configure static (nonchanging) routes between the PE and CE routers of a VPN routing instance. To configure a static route for a VPN, you need to configure it within the VPN routing instance configuration at the **[edit routing-instances** *routing-instance-name* **routing-options]** hierarchy level.

To configure a static route between the PE and the CE routers, include the **static** statement:

```
static {
  route destination-prefix {
    next-hop [ next-hops ];
    static-options;
  }
}
```

You can include this statement at the following hierarchy levels:

- **[edit routing-instances** *routing-instance-name* **routing-options]**
- **[edit logical-systems** *logical-system-name* **routing-instances** *routing-instance-name* **routing-options]**

For more information about configuring routing protocols and static routes, see the *Junos OS Routing Protocols Configuration Guide*.

Limiting the Number of Paths and Prefixes Accepted from CE Routers in Layer 3 VPNs

You can configure a maximum limit on the number of prefixes and paths that can be installed into the routing tables. Using prefix and path limits, you can curtail the number of prefixes and paths received from a CE router in a VPN. Prefix and path limits apply only to dynamic routing protocols, and are not applicable to static or interface routes.

To limit the number of paths accepted by a PE router from a CE router, include the **maximum-paths** statement:

maximum-paths *path-limit* <log-interval *interval* | log-only | threshold *percentage*>;

For a list of hierarchy levels at which you can configure this statement, see the statement summary section for this statement.

To limit the number of prefixes accepted by a PE router from a CE router, include the **maximum-prefixes** statement:

maximum-prefixes *prefix-limit* <log-interval *interval* | log-only | threshold *percentage*>;

For a list of hierarchy levels at which you can configure this statement, see the statement summary section for this statement.

A mandatory path or prefix limit, in addition to triggering a warning message, rejects any additional paths or prefixes once the limit is reached.

NOTE: **Setting a path or prefix limit might result in unpredictable dynamic routing protocol behavior.**

You can also configure the following options for both the **maximum-paths** and **maximum-prefixes** statements:

- **log-interval**—Specify the interval at which log messages are sent.
- **log-only**—Generate warning messages only. No limit is placed on the number of paths or prefixes stored in the routing tables.
- **threshold**—Generate warning messages after the specified percentage of the maximum paths or prefixes has been reached.

Configuring Layer 3 VPNs to Carry IPv6 Traffic

You can configure IP version 6 (IPv6) between the PE and CE routers of a Layer 3 VPN. The PE router must have the PE router to PE router BGP session configured with the **family inet6-vpn** statement. The CE router must be capable of receiving IPv6 traffic. You can configure BGP or static routes between the PE and CE routers.

The following sections explains how to configure IPv6 VPNs between the PE routers:

Configuring IPv6 on the PE Router

To configure IPv6 between the PE and CE routers, include the **family inet6-vpn** statement in the configuration on the PE router:

```
family inet6-vpn {
  (any | multicast | unicast) {
    aggregate-label community community-name;
    prefix-limit maximum prefix-limit;
    rib-group rib-group-name;
  }
}
```

For a list of hierarchy levels at which you can configure this statement, see the statement summary section for this statement.

You also must include the **ipv6-tunneling** statement:

```
ipv6-tunneling;
```

You can include this statement at the following hierarchy levels:

- **[edit protocols mpls]**
- **[edit logical-systems *logical-system-name* protocols mpls]**

Configuring the Connection Between the PE and CE Routers

To support IPv6 routes, you must configure BGP, OSPF version 3, or static routes for the connection between the PE and CE routers in the Layer 3 VPN. You can configure BGP to handle just IPv6 routes or both IP version 4 (IPv4) and IPv6 routes.

For more information about IPv6, see the *Junos OS Routing Protocols Configuration Guide*.

The following sections explain how to configure BGP and static routes:

Configuring BGP on the PE Router to Handle IPv6 Routes

To configure BGP in the Layer 3 VPN routing instance to handle IPv6 routes, include the **bgp** statement:

```
bgp {
  group group-name {
```

```
        local-address IPv6-address;
        family inet6 {
          unicast;
        }
        peer-as as-number;
        neighbor IPv6-address;
      }
    }
```

You can include this statement at the following hierarchy levels:

- **[edit routing-instances *routing-instance-name* protocols]**
- **[edit logical-systems *logical-system-name* routing-instances *routing-instance-name* protocols]**

Configuring BGP on the PE Router for IPv4 and IPv6 Routes

To configure BGP in the Layer 3 VPN routing instance to handle both IPv4 and IPv6 routes, include the **bgp** statement:

```
bgp {
  group group-name {
    local-address IPv4-address;
    family inet {
      unicast;
    }
    family inet6 {
      unicast;
    }
    peer-as as-number;
    neighbor address;
  }
}
```

You can include this statement at the following hierarchy levels:

- **[edit routing-instances *routing-instance-name* protocols]**
- **[edit logical-systems *logical-system-name* routing-instances *routing-instance-name* protocols]**

Configuring OSPF Version 3 on the PE Router

To configure OSPF version 3 in the Layer 3 VPN routing instance to handle IPv6 routes, include the **ospf3** statement:

```
ospf3 {
  area area-id {
    interface interface-name;
  }
}
```

You can include this statement at the following hierarchy levels:

- **[edit routing-instances *routing-instance-name* protocols]**

- **[edit logical-systems *logical-system-name* routing-instances *routing-instance-name* protocols]**

For complete configuration guidelines for this statement, see the *Junos OS Routing Protocols Configuration Guide*.

Configuring Static Routes on the PE Router

To configure a static route to the CE router in the Layer 3 VPN routing instance, include the **routing-options** statement:

```
routing-options {
  rib routing-table.inet6.0 {
    static {
      defaults {
        static-options;
      }
    }
  }
}
```

You can include this statement at the following hierarchy levels:

- **[edit routing-instances *routing-instance-name*]**
- **[edit logical-systems *logical-system-name* routing-instances *routing-instance-name*]**

Configuring IPv6 on the Interfaces

You need to configure IPv6 on the PE router interfaces to the CE routers and on the CE router interfaces to the PE routers.

To configure the interface to handle IPv6 routes, include the **family inet6** statement:

```
family inet6 {
  address ipv6-address;
}
```

You can include this statement at the following hierarchy levels:

- **[edit interfaces *interface-name* unit *unit-number*]**
- **[edit logical-systems *logical-system-name* interfaces *interface-name* unit *unit-number*]**

If you have configured the Layer 3 VPN to handle both IPv4 and IPv6 routes, configure the interface to handle both IPv4 and IPv6 routes by including the **unit** statement:

```
unit unit-number {
  family inet {
    address ipv4-address;
  }
  family inet6 {
    address ipv6-address;
  }
}
```

You can include this statement at the following hierarchy levels:

- **[edit interfaces** *interface-name***]**
- **[edit logical-systems** *logical-system-name* **interfaces** *interface-name***]**

Configuring EBGP Multihop Sessions Between PE and CE Routers in Layer 3 VPNs

You can configure an EBGP or IBGP multihop session between the PE and CE routers of a Layer 3 VPN. This allows you to have one or more routers between the PE and CE routers. Using IBGP between PE and CE routers does not require the configuration of any additional statements. However, using EBGP between the PE and CE routers requires the configuration of the **multihop** statement.

To configure an external BGP multihop session for the connection between the PE and CE routers, include the **multihop** statement on the PE router. To help prevent routing loops, you have to configure a time-to-live (TTL) value for the multihop session:

multihop *ttl-value*;

For the list of hierarchy levels at which you can configure this statement, see the summary section for this statement.

Configuring Layer 3 VPNs to Carry IBGP Traffic

When you configure BGP as the routing protocol between a PE router and a CE router in a Layer 3 VPN, you typically configure external peering sessions between the Layer 3 VPN service provider and the customer network ASs.

If the customer network has several sites advertising routes through an external BGP session to the service provider network and if the same AS is used by all the customer sites, the CE routers reject routes from the other CE routers. They detect a loop in the BGP AS path attribute.

To prevent the CE routers from rejecting each other's routes, you could configure the following:

- PE routers advertising routes received from remote PE routers can remap the customer network AS number to its own AS number.
- AS path loops can be configured.
- The customer network can be configured with different AS numbers at each site.

These types of configurations can work when there are no BGP routing exchanges between the customer network and other networks. However, they do have limitations for customer networks that use BGP internally for purposes other than carrying traffic between the CE routers and the PE routers. When those routes are advertised outside the customer network, the service provider ASs are present in the AS path.

To improve the transparency of Layer 3 VPN services for customer networks, you can configure the routing instance for the Layer 3 VPN to isolate the customer's network attributes from the service provider's network attributes.

When you include the **independent-domain** statement in the Layer 3 VPN routing instance configuration, BGP attributes received from the customer network (from the CE router) are stored in a BGP attribute (ATTRSET) that functions like a stack. When that route is advertised from the remote PE router to the remote CE router, the original BGP attributes are restored. This is the default behavior for BGP routes that are advertised to Layer 3 VPNs located in different domains.

This functionality is described in the Internet draft draft-marques-ppvpn-ibgp-version.txt, *RFC 2547bis Networks Using Internal BGP as PE-CE Protocol*.

To allow a Layer 3 VPN to transport IBGP traffic, include the **independent-domain** statement:

```
independent-domain;
```

You can include this statement at the following hierarchy levels:

- **[edit routing-instances *routing-instance-name* routing-options autonomous-system *number*]**
- **[edit logical-systems *logical-system-name* routing-instances *routing-instance-name* routing-options autonomous-system *number*]**

NOTE: All PE routers participating in a Layer 3 VPN with the **independent-domain** statement in its configuration must be running Junos OS Release 6.3 or later.

Filtering Packets in Layer 3 VPNs Based on IP Headers

Including the **vrf-table-label** statement in the configuration for a routing instance makes it possible to map the inner label to a specific VRF routing table; such mapping allows the examination of the encapsulated IP header at an egress VPN router. You might want to enable this functionality so that you can do either of the following:

- Forward traffic on a PE-router-to-CE-device interface, in a shared medium, where the CE device is a Layer 2 switch without IP capabilities (for example, a metro Ethernet switch).

  The first lookup is done on the VPN label to determine which VRF table to refer to, and the second lookup is done on the IP header to determine how to forward packets to the correct end hosts on the shared medium.

- Perform egress filtering at the egress PE router.

  The first lookup on the VPN label is done to determine which VRF routing table to refer to, and the second lookup is done on the IP header to determine how to filter and forward packets. You can enable this functionality by configuring output filters on the VRF interfaces.

  When you include the **vrf-table-label** statement in the configuration of a VRF routing table, a label-switched interface (LSI) logical interface label is created and mapped

to the VRF routing table. Any routes in such a VRF routing table are advertised with the LSI logical interface label allocated for the VRF routing table. When packets for this VPN arrive on a core-facing interface, they are treated as if the enclosed IP packet arrived on the LSI interface and are then forwarded and filtered based on the correct table.

To filter traffic based on the IP header, include the **vrf-table-label** statement:

vrf-table-label;

You can include the statement at the following hierarchy levels:

- **[edit routing-instances *routing-instance-name*]**
- **[edit logical-systems *logical-system-name* routing-instances *routing-instance-name*]**

You can include the **vrf-table-label** statement for both IPv4 and IPv6 Layer 3 VPNs. If you include the statement for a dual-stack VRF routing table (where both IPv4 and IPv6 routes are supported), the statement applies to both the IPv4 and IPv6 routes and the same label is advertised for both sets of routes.

The following sections provide more information about traffic filtering based on the IP header:

- Egress Filtering Options on page 44
- Support on Aggregated and VLAN Interfaces for IP-Based Filtering on page 45
- Support on ATM and Frame Relay Interfaces for IP-Based Filtering on page 45
- Support on Ethernet, SONET/SDH, and T1/T3/E3 Interfaces for IP-Based Filtering on page 46
- Support on SONET/SDH and DS3/E3 Channelized Enhanced Intelligent Queuing Interfaces for IP-Based Filtering on page 46
- Support on Multilink PPP and Multilink Frame Relay Interfaces for IP-Based Filtering on page 48
- Support for IP-Based Filtering of Packets with Null Top Labels on page 48
- General Limitations on IP-Based Filtering on page 49

Egress Filtering Options

You can enable egress filtering (which allows egress Layer 3 VPN PE routers to perform lookups on the VPN label and IP header at the same time) by including the **vrf-table-label** statement at the **[edit routing-instances *instance-name*]** hierarchy level. There is no restriction on including this statement for CE-router-to-PE-router interfaces, but there are several limitations on other interface types, as described in subsequent sections in this topic.

You can also enable egress filtering by configuring a VPN tunnel (VT) interface on routing platforms equipped with a Tunnel Services Physical Interface Card (PIC). When you enable egress filtering this way, there is no restriction on the type of core-facing interface used. There is also no restriction on the type of CE-router-to-PE-router interface used.

Support on Aggregated and VLAN Interfaces for IP-Based Filtering

Support for the **vrf-table-label** statement over aggregated and VLAN interfaces is available on the routers summarized in Table 4 on page 45.

Table 4: Support for Aggregated and VLAN Interfaces

Interfaces	J Series Router in Switching Mode	M Series Router Without an Enhanced FPC	M Series Router with an Enhanced FPC	M320 Router	T Series Router
Aggregated	N/A	No	Yes	Yes	Yes
VLAN	Yes	No	Yes	Yes	Yes

NOTE: The **vrf-table-label** statement is not supported for Aggregated Gigabit Ethernet, 10-Gigabit Ethernet, and VLAN physical interfaces on M120 routers.

Support on ATM and Frame Relay Interfaces for IP-Based Filtering

Support for the **vrf-table-label** statement over Asynchronous Transfer Mode (ATM) and Frame Relay interfaces is available on the routers summarized in Table 5 on page 45.

Table 5: Support for ATM and Frame Relay Interfaces

Interfaces	J Series Router	M Series Router Without an Enhanced FPC	M Series Router with an Enhanced FPC	M320 Router	T Series Router
ATM1	N/A	No	No	No	No
ATM2 intelligent queuing (IQ)	N/A	No	Yes	Yes	Yes
Frame Relay	Yes	No	Yes	Yes	Yes
Channelized	N/A	No	No	No	No

When you include the **vrf-table-label** statement, be aware of the following limitations with ATM or Frame Relay interfaces:

- The **vrf-table-label** statement is supported on ATM interfaces, but with the following limitations:
  - ATM interfaces can be configured on the M320 router and the T Series routers, and on M Series routers with an enhanced FPC.
  - The interface can only be a PE router interface receiving traffic from a P router.

- The router must have an ATM2 IQ PIC.

- The **vrf-table-label** statement is also supported on Frame Relay encapsulated interfaces, but with the following limitations:

  - Frame Relay interfaces can be configured on the M320 router and the T Series routers, and on M Series routers with an enhanced FPC.
  - The interface can only be a PE router interface receiving traffic from a P router.

Support on Ethernet, SONET/SDH, and T1/T3/E3 Interfaces for IP-Based Filtering

Support for the **vrf-table-label** statement over Ethernet, SONET/SDH, and T1/T3/E3 interfaces is available on the routers summarized in Table 6 on page 46.

Table 6: Support for Ethernet, SONET/SDH, and T1/T3/E3 Interfaces

Interfaces	J Series Router	M Series Router Without an Enhanced FPC	M Series Router with an Enhanced FPC	M320 Router	T Series Router
Ethernet	Yes	Yes	Yes	Yes	Yes
SONET/SDH	N/A	Yes	Yes	Yes	Yes
T1/T3/E3	Yes	Yes	Yes	Yes	Yes

Only the following Ethernet PICs support the **vrf-table-label** statement on M Series routers without an Enhanced FPC:

- 1-port Gigabit Ethernet
- 2-port Gigabit Ethernet
- 4-port Fast Ethernet

Support on SONET/SDH and DS3/E3 Channelized Enhanced Intelligent Queuing Interfaces for IP-Based Filtering

Support for the **vrf-table-label** statement for the specified channelized IQE interfaces is only available on M120 and M320 routers with Enhanced III FPCs as summarized in Table 7 on page 46.

Table 7: Support for Channelized IQE Interfaces on M320 Routers with Enhanced III FPCs

Interfaces	M120 Routers with Enhanced III FPCs	M320 Routers with Enhanced III FPCs
OC12	Yes	Yes
STM4	Yes	Yes
OC3	Yes	Yes

Table 7: Support for Channelized IQE Interfaces on M320 Routers with Enhanced III FPCs *(continued)*

Interfaces	M120 Routers with Enhanced III FPCs	M320 Routers with Enhanced III FPCs
STM1	Yes	Yes
DS3	Yes	Yes
E3	Yes	Yes

The following IQE Type-1 PICs are supported:

- 1-port OC12/STM4 IQE with SFP
- 4-port OC3/STM1 IQE with SFP
- 4-port DS3/E3 IQE with BNC
- 2-port Channelized OC3/STM1 IQE with SFP, with no SONET partitions
- 1-port Channelized OC12/STM4 IQE with SFP, with no SONET partitions

The following constraints are applicable with respect to a router configuration utilizing logical systems:

- Multiport IQE PIC interfaces constraints—On multiport IQE PICs, such as the 2-port Channelized OC3/STM1 IQE with SFP, if the port 1 interface is configured as one logical system with its own routing-instance and the port 2 interface is configured as a different logical system with its own routing instances such that there are core-facing logical interfaces on both port 1 and port 2, then you cannot configure the **vrf-table-label** statement on routing-instance in both logical systems. Only one set of LSI labels are supported; the last routing instance with the **vrf-table-label** statement configured is committed.
- Frame Relay encapsulation and logical interfaces across logical systems constraints—Similar to the multiport PIC with logical systems, if you try to configure one logical interface of an IQE PIC with Frame Relay encapsulation in one logical system and configure another logical interface on the same IQE PIC in the second logical system, the configuration will not work for all the **vrf-table-label** statement configured instances. It will only work for the instances configured in one of the logical systems.

Both the above constraints occur because the router configuration maintains one LSI tree in the Packet Forwarding Engine per logical system, which is common across all streams. The stream channel table lookup is then adjusted to point to the LSI tree. In the case of multiport type-1 IQE PICs, all physical interfaces share the same stream. Therefore, the logical interfaces (multiport or not) obviously share the same stream. Consequently, the LSI binding is at the stream level. Hence, provisioning logical interfaces under the same stream provisioned to be core-facing and supporting a different set of routing instances with the **vrf-table-label** statement is not supported.

Support on Multilink PPP and Multilink Frame Relay Interfaces for IP-Based Filtering

Support for the **vrf-table-label** statement over Multilink Point-to-Point Protocol (MLPPP) and Multilink Frame Relay (MLFR) interfaces is available on the routers summarized in Table 8 on page 48.

Table 8: Support for Multilink PPP and Multilink Frame Relay Interfaces

Interfaces	J Series Router	M Series Router Without an Enhanced FPC	M Series Router with an Enhanced FPC	M320	T Series Router	MX Series Router
MLPPP	Yes	No	Yes	No	No	No
End-to-End MLFR (FRF.15)	Yes	No	Yes	No	No	No
UNI/NNI MLFR (FRF.16)	Yes	No	No	No	No	No

M Series routers must have an AS PIC to support the **vrf-table-label** statement over MLPPP and MLFR interfaces. The **vrf-table-label** statement over MLPPP interfaces is not supported on M120 routers.

Support for IP-Based Filtering of Packets with Null Top Labels

You can include the **vrf-table-label** statement in the configuration for core-facing interfaces receiving MPLS packets with a null top label, which might be transmitted by some vendors' equipment. These packets can be received only on the M320 router, the M10i router, and T Series Core routers using one of the following PICs:

- 1-port Gigabit Ethernet with SFP
- 2-port Gigabit Ethernet with SFP
- 4-port Gigabit Ethernet with SFP
- 10-port Gigabit Ethernet with SFP
- 1-port SONET STM4
- 4-port SONET STM4
- 1-port SONET STM16
- 1-port SONET STM16 (non-SFP)
- 4-port SONET STM16
- 1-port SONET STM64

The following PICs can receive packets with null top labels, but only when installed in an M120 router or an M320 router with an Enhanced III FPC:

- 1-port 10-Gigabit Ethernet

- 1-port 10-Gigabit Ethernet IQ2

General Limitations on IP-Based Filtering

The following limitations apply when you include the **vrf-table-label** statement:

- The time-to-live (TTL) value in the MPLS header is not copied back to the IP header of packets sent from the PE router to the CE router.
- You cannot include the statement in a routing instance configuration that also includes a virtual loopback tunnel interface; the commit operation fails in this case.
- You cannot include the statement in source class usage (SCU) or destination class usage (DCU) configurations. For information about SCU and DCU configuration, see the *Junos OS Network Interfaces Configuration Guide*.
- You can include the statement in the configuration for Multilink Frame Relay (MLFR FRF.16) encapsulated PE-router-to-P-router interfaces only on J Series routers.
- When you include the statement, MPLS packets with label-switched interface (LSI) labels that arrive on core-facing interfaces are not counted at the logical interface level if the core-facing interface is any of the following:
  - ATM
  - Frame Relay
  - Ethernet configured with VLANs
  - Aggregated Ethernet configured with VLANs
- You cannot include the statement in the configuration of a VRF routing instance if the PE-router-to-P-router interface is any of the following interfaces:
  - Aggregated SONET/SDH interface
  - Channelized interface
  - Tunnel interface (for example, generic routing encapsulation [GRE] or IP Security [IPsec])
  - Circuit cross-connect (CCC) or translational cross-connect (TCC) encapsulated interface
  - Logical tunnel interface
  - Virtual private LAN service (VPLS) encapsulated interface

  NOTE: All CE-router-to-PE-router and PE-router-to-CE-router interfaces are supported.

- You cannot include the **vrf-table-label** statement in the configuration of a VRF routing instance if the PE-router-to-P-router PIC is one of the following PICs:

- 10-port E1
- 8-port Fast Ethernet
- 12-port Fast Ethernet
- 48-port Fast Ethernet
- ATM PIC other than the ATM2 IQ

Applying Custom MPLS EXP Classifiers to Routing Instances in Layer 3 VPNs

When you include the **vrf-table-label** statement in the configuration for a routing instance (as described in "Filtering Packets in Layer 3 VPNs Based on IP Headers" on page 43) but do not explicitly apply a classifier to the routing instance, the default MPLS EXP classifier is applied.

For PICs that are installed on Enhanced FPCs, you can apply a custom classifier to override the default MPLS EXP classifier for the routing instance. For detailed instructions, see the *Junos OS Class of Service Configuration Guide*. The following instructions serve as a summary:

1. Filter traffic based on the IP header by including the **vrf-table-label** statement at the **[edit routing-instances *routing-instance-name*]** hierarchy level:

   ```
   [edit routing-instances routing-instance-name]
   vrf-table-label;
   ```

2. Configure a custom MPLS EXP classifier by including the appropriate statements at the **[edit class-of-service]** hierarchy level. For instructions, see the *Junos OS Class of Service Configuration Guide*.

3. Configure the routing instance for CoS by including the **routing-instances** statement at the **[edit class-of-service]** hierarchy level:

   ```
   [edit class-of-service]
   routing-instances routing-instance-name {
     classifiers {
       exp (classifier-name | default);
     }
   }
   ```

4. Configure the routing instance to use the custom MPLS EXP classifier by including the **classifiers** statement at the **[edit class-of-service routing-instances *routing-instance-name*]** hierarchy level:

   ```
   [edit class-of-service routing-instances routing-instance-name]
   classifiers {
     exp classifier-name;
   }
   ```

To display the MPLS EXP classifiers associated with all routing instances, issue the **show class-of-service routing-instances** command.

NOTE: The following caveats apply to custom MPLS EXP classifiers for routing instances:

- An Enhanced FPC is required.
- Logical systems are not supported.

Load Balancing and IP Header Filtering for Layer 3 VPNs

You can now simultaneously enable both load balancing of traffic across both internal and external BGP paths and filtering of traffic based on the IP header. This enables you to configure filters and policers at the egress PE router for traffic that is simultaneously being load-balanced across both internal and external BGP paths. This feature is available only on the M120 router, M320 router, MX Series routers, and T Series routers.

To enable these features on a Layer 3 VPN routing instance, include the **vpn-unequal-cost equal-external-internal** statement at the **[edit routing-instances *routing-instance-name* routing-options multipath]** hierarchy level and the **vrf-table-label** statement at the **[edit routing-instances *routing-instance-name*]** hierarchy level.

If you issue the **show route detail** command, you can discover whether or not a route is being load-balanced (equal-external-internal) and what its interface index is.

If you have also configured fast reroute, please be aware of the following behavior:

- If an IBGP path goes down, it could be replaced by either an active EBGP path or an active IBGP path.
- If an EBGP path goes down, it can only be replaced by another active EBGP path. This prevents the forwarding of core-facing interface traffic to an IBGP destination.

NOTE: You can include the **vpn-unequal-cost equal-external-internal** statement and the **l3vpn-composite-nexthop** statement simultaneously. However, if you do this, EBGP does not work. This means that when there are both paths with chained nexthops and paths with nonchained nexthops as candidates for EBGP equal-cost multipath (ECMP), the paths using chained nexthops are excluded. In a typical case, the excluded paths are the internal paths.

Example: Load Balancing Layer 3 VPN Traffic Using IP Header Filtering

This example shows how to configure load balancing in a Layer 3 VPN (with internal and external BGP paths) using IP header filtering.

Requirements

This example requires the following hardware and software components:

- M Series Multiservice Edge Routers (M120 and M320 only), MX Series 3D Universal Edge Routers, or T Series Core Routers
- Junos OS Release 12.1 or later

Overview

The following example shows how to configure load balancing with IP header filtering in a Layer 3 VPN.

The Junos OS BGP provides a multipath feature that allows load balancing between peers in the same or different autonomous systems (ASs). This example uses the **equal-external-internal** statement at the **[edit routing-instances instance-name routing-options multipath vpn-unequal-cost]** hierarchy level to perform load balancing. The **vrf-table-label** statement is configured at the **[edit routing-instances instance-name]** hierarchy level to enable IP header filtering.

```
[edit]
routing-instances {
   instance-name {
      vrf-table-label;
      routing-options {
         multipath {
            vpn-unequal-cost {
               equal-external-internal;
            }
         }
      }
   }
}
```

NOTE: These statements are available only in the context of a routing instance.

In this example, Device CE1 is in AS1 and connected to Device PE1. Devices PE1, PE2, PE3, and P are in AS2. Device CE2 is connected to Devices PE2 and PE3 and is in AS3. Device CE3 is connected to Device PE3 and is in AS4. BGP and MPLS are configured through the network. OSPF is the interior gateway protocol (IGP) that is used in this network.

The configuration for Devices PE1, PE2, and PE3 includes the **equal-external-internal** statement at the **[edit routing-instances instance-name routing-options multipath vpn-unequal-cost]** hierarchy level to enable load balancing in the network. IP header filtering is enabled when the **vrf-table-label** statement is configured at the **[edit routing-instances instance-name]** hierarchy level on the PE devices.

Figure 13 on page 53 shows the topology used in this example.

Figure 13: Layer 3 VPN Load Balancing Using IP Header Filtering

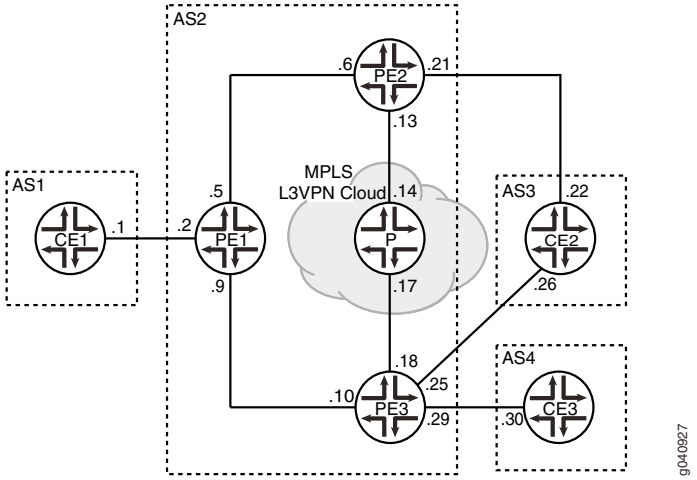


Table 9 on page 53 shows the list of IPs used in this example for quick reference.

Table 9: Device IP Address Quick Reference

Device	AS	Device ID	Device Interface Units	Device Interface Unit IPs
CE1	1	1.1.1.1/32	Unit 1	10.1.1.1/30
PE1	2	1.1.1.2/32	Unit 2	10.1.1.2/30
			Unit 5	10.1.2.5/30
			Unit 9	10.1.3.9/30
PE2	2	1.1.1.3/32	Unit 6	10.1.2.6/30
			Unit 13	10.1.4.13/30
			Unit 21	10.1.6.21/30

Table 9: Device IP Address Quick Reference *(continued)*

Device	AS	Device ID	Device Interface Units	Device Interface Unit IPs
PE3	2	1.1.1.4/32	Unit 10	10.1.3.10/30
			Unit 18	10.1.5.18/30
			Unit 25	10.1.7.25/30
			Unit 29	10.1.8.29/30
P	2	1.1.1.5/32	Unit 14	10.1.4.14/30
			Unit 17	10.1.5.17/30
CE2	3	1.1.1.6/32	Unit 22	10.1.6.22/30
			Unit 26	10.1.7.26/30
CE3	4	1.1.1.7/32	Unit 30	10.1.8.30/30

NOTE: This example was tested using logical systems (logical routers). Therefore all the physical interfaces in the example are the same and the configuration is done on separate logical interfaces. In an non-test network, you will use separate physical routers and separate physical interfaces for the connections to other devices.

Configuration

CLI Quick Configuration

To quickly configure this example, copy the following commands, paste them into a text file, remove any line breaks, change any details necessary to match your network configuration, and then copy and paste the commands into the CLI at the **[edit]** hierarchy level.

Device CE1

```
set interfaces ge-2/1/10 unit 1 family inet address 10.1.1.1/30
set interfaces ge-2/1/10 unit 1 family mpls
set interfaces ge-2/1/10 unit 1 description toPE1
set interfaces lo0 unit 4 family inet address 1.1.1.1/32
set routing-options router-id 1.1.1.1
set routing-options autonomous-system 1
set protocols bgp group toPE1 type external
set protocols bgp group toPE1 export send-direct
set protocols bgp group toPE1 peer-as 2
set protocols bgp group toPE1 neighbor 10.1.1.2
set policy-options policy-statement send-direct from protocol direct
set policy-options policy-statement send-direct then accept
```

Device PE1

```
set interfaces ge-2/1/10 unit 2 family inet address 10.1.1.2/30
set interfaces ge-2/1/10 unit 2 family mpls
set interfaces ge-2/1/10 unit 2 description toCE1
set interfaces ge-2/1/10 unit 5 family inet address 10.1.2.5/30
set interfaces ge-2/1/10 unit 5 family mpls
set interfaces ge-2/1/10 unit 5 description toPE2
set interfaces ge-2/1/10 unit 9 family inet address 10.1.3.9/30
set interfaces ge-2/1/10 unit 9 family mpls
set interfaces ge-2/1/10 unit 9 description toPE3
set interfaces lo0 unit 5 family inet address 1.1.1.2/32
set protocols mpls interface all
set protocols ldp interface all
set protocols ospf area 0.0.0.0 interface lo0.5 passive
set protocols ospf area 0.0.0.0 interface ge-2/1/10.5 metric 10
set protocols ospf area 0.0.0.0 interface ge-2/1/10.9 metric 10
set protocols bgp group toInternal type internal
set protocols bgp group toInternal family inet-vpn unicast
set protocols bgp group toInternal local-address 1.1.1.2
set protocols bgp group toInternal neighbor 1.1.1.3
set protocols bgp group toInternal neighbor 1.1.1.4
set routing-options router-id 1.1.1.2
set routing-options autonomous-system 2
set routing-options forwarding-table export lb
set routing-instances purple instance-type vrf
set routing-instances purple interface ge-2/1/10.2
set routing-instances purple route-distinguisher 2:1
set routing-instances purple vrf-target target:2:1
set routing-instances purple vrf-table-label
set routing-instances purple protocols bgp group toCE1 type external
set routing-instances purple protocols bgp group toCE1 peer-as 1
set routing-instances purple protocols bgp group toCE1 neighbor 10.1.1.1
set routing-instances purple routing-options multipath vpn-unequal-cost
    equal-external-internal
set policy-options policy-statement lb then load-balance per-packet
```

Device PE2

```
set interfaces ge-2/1/10 unit 6 family inet address 10.1.2.6/30
set interfaces ge-2/1/10 unit 6 family mpls
set interfaces ge-2/1/10 unit 6 description toPE1
set interfaces ge-2/1/10 unit 13 family inet address 10.1.4.13/30
set interfaces ge-2/1/10 unit 13 family mpls
set interfaces ge-2/1/10 unit 13 description toP
set interfaces ge-2/1/10 unit 21 family inet address 10.1.6.21/30
set interfaces ge-2/1/10 unit 21 family mpls
set interfaces ge-2/1/10 unit 21 description toCE2
set interfaces lo0 unit 6 family inet address 1.1.1.3/32
set protocols mpls interface all
set protocols ldp interface all
set protocols ospf area 0.0.0.0 interface lo0.6 passive
set protocols ospf area 0.0.0.0 interface ge-2/1/10.6 metric 10
set protocols ospf area 0.0.0.0 interface ge-2/1/10.13 metric 5
set protocols bgp group toInternal type internal
set protocols bgp group toInternal family inet-vpn unicast
```

```
set protocols bgp group toInternal local-address 1.1.1.3
set protocols bgp group toInternal neighbor 1.1.1.2
set protocols bgp group toInternal neighbor 1.1.1.4
set routing-options router-id 1.1.1.3
set routing-options autonomous-system 2
set routing-options forwarding-table export lb
set routing-instances purple instance-type vrf
set routing-instances purple interface ge-2/1/10.21
set routing-instances purple route-distinguisher 2:1
set routing-instances purple vrf-target target:2:1
set routing-instances purple vrf-table-label
set routing-instances purple protocols bgp group toCE2 type external
set routing-instances purple protocols bgp group toCE2 peer-as 3
set routing-instances purple protocols bgp group toCE2 neighbor 10.1.6.22
set routing-instances purple routing-options multipath vpn-unequal-cost
  equal-external-internal
set policy-options policy-statement lb then load-balance per-packet
```

Device PE3

```
set interfaces ge-2/1/10 unit 10 family inet address 10.1.3.10/30
set interfaces ge-2/1/10 unit 10 family mpls
set interfaces ge-2/1/10 unit 10 description toPE1
set interfaces ge-2/1/10 unit 18 family inet address 10.1.5.18/30
set interfaces ge-2/1/10 unit 18 family mpls
set interfaces ge-2/1/10 unit 18 description toP
set interfaces ge-2/1/10 unit 25 family inet address 10.1.7.25/30
set interfaces ge-2/1/10 unit 25 family mpls
set interfaces ge-2/1/10 unit 25 description toCE2
set interfaces ge-2/1/10 unit 29 family inet address 10.1.8.29/30
set interfaces ge-2/1/10 unit 29 family mpls
set interfaces ge-2/1/10 unit 29 description toCE3
set interfaces lo0 unit 7 family inet address 1.1.1.4/32
set protocols mpls interface all
set protocols ldp interface all
set protocols ospf area 0.0.0.0 interface lo0.7 passive
set protocols ospf area 0.0.0.0 interface ge-2/1/10.10 metric 10
set protocols ospf area 0.0.0.0 interface ge-2/1/10.18 metric 5
set protocols bgp group toInternal type internal
set protocols bgp group toInternal local-address 1.1.1.4
set protocols bgp group toInternal family inet unicast
set protocols bgp group toInternal family inet-vpn unicast
set protocols bgp group toInternal family route-target
set protocols bgp group toInternal export next-hop-self
set protocols bgp group toInternal neighbor 1.1.1.2
set protocols bgp group toInternal neighbor 1.1.1.3
set routing-options router-id 1.1.1.4
set routing-options autonomous-system 2
set routing-options forwarding-table export lb
set routing-instances purple instance-type vrf
set routing-instances purple interface ge-2/1/10.25
set routing-instances purple interface ge-2/1/10.29
set routing-instances purple route-distinguisher 2:1
set routing-instances purple vrf-target target:2:1
set routing-instances purple vrf-table-label
set routing-instances purple protocols bgp group toCE2 type external
```

```
set routing-instances purple protocols bgp group toCE2 peer-as 3
set routing-instances purple protocols bgp group toCE2 neighbor 10.1.7.26
set routing-instances purple protocols bgp group toCE3 type external
set routing-instances purple protocols bgp group toCE3 peer-as 4
set routing-instances purple protocols bgp group toCE3 neighbor 10.1.8.30
set routing-instances purple routing-options multipath vpn-unequal-cost
  equal-external-internal
set policy-options policy-statement lb then load-balance per-packet
set policy-options policy-statement next-hop-self then next-hop self
```

Device P

```
set interfaces ge-2/1/10 unit 14 family inet address 10.1.4.14/30
set interfaces ge-2/1/10 unit 14 family mpls
set interfaces ge-2/1/10 unit 14 description toPE2
set interfaces ge-2/1/10 unit 17 family inet address 10.1.5.17/30
set interfaces ge-2/1/10 unit 17 family mpls
set interfaces ge-2/1/10 unit 17 description toPE3
set interfaces lo0 unit 8 family inet address 1.1.1.5/32
set protocols mpls interface all
set protocols ldp interface all
set protocols ospf area 0.0.0.0 interface lo0.8 passive
set protocols ospf area 0.0.0.0 interface ge-2/1/10.14 metric 5
set protocols ospf area 0.0.0.0 interface ge-2/1/10.17 metric 5
set routing-options router-id 1.1.1.5
set routing-options autonomous-system 2
```

Device CE2

```
set interfaces ge-2/1/10 unit 22 family inet address 10.1.6.22/30
set interfaces ge-2/1/10 unit 22 family mpls
set interfaces ge-2/1/10 unit 22 description toPE2
set interfaces ge-2/1/10 unit 26 family inet address 10.1.7.26/30
set interfaces ge-2/1/10 unit 26 family mpls
set interfaces ge-2/1/10 unit 26 description toPE3
set interfaces lo0 unit 6 family inet address 1.1.1.6/32
set routing-options router-id 1.1.1.6
set routing-options autonomous-system 3
set protocols bgp group toAS2 type internal
set protocols bgp group toAS2 export send-direct
set protocols bgp group toAS2 peer-as 2
set protocols bgp group toAS2 neighbor 10.1.6.21
set protocols bgp group toAS2 neighbor 10.1.7.25
set policy-options policy-statement send-direct from protocol direct
set policy-options policy-statement send-direct then accept
set policy-options policy-statement lb then load-balance per-packet
```

Device CE3

```
set interfaces ge-2/1/10 unit 30 family inet address 10.1.8.30/30
set interfaces ge-2/1/10 unit 30 family mpls
set interfaces ge-2/1/10 unit 30 description toPE3
set interfaces lo0 unit 7 family inet address 1.1.1.7/32
set routing-options router-id 1.1.1.7
set routing-options autonomous-system 4
set protocols bgp group toPE3 type internal
set protocols bgp group toPE3 export send-direct
```

```
set protocols bgp group toPE3 peer-as 2
set protocols bgp group toPE3 neighbor 10.1.8.29
set policy-options policy-statement send-direct from protocol direct
set policy-options policy-statement send-direct then accept
```

Step-by-Step Procedure

The following example requires that you navigate various levels in the configuration hierarchy. For information about navigating the CLI, see Using the CLI Editor in Configuration Mode in the *Junos OS CLI User Guide*.

To configure unequal-cost load balancing across the VPN setup:

1. Configure the router ID on Device CE1, and assign the device to its autonomous system.

   ```
   [edit routing-options]
   user@CE1# set routing-options router-id 1.1.1.1
   user@CE1# set routing-options autonomous-system 1
   ```

   Similarly, configure all other devices.

2. Configure BGP groups for traffic through the entire network.

   a. Configure the BGP group for traffic to and from the MPLS network (CE devices).

      ```
      [edit protocols bgp group toPE1]
      user@CE1# set type external
      user@CE1# set peer-as 2
      user@CE1# set neighbor 10.1.1.2
      ```

   b. Configure a similar BGP group (**toPE3**) on Device CE3 by modifying the **peer-as** and **neighbor** statements accordingly.

   c. Configure the BGP group for traffic through the MPLS network (PE devices).

      ```
      [edit protocols bgp group toInternal]
      user@PE1# set type internal
      user@PE1# set family inet-vpn unicast
      user@PE1# set local-address 1.1.1.2
      user@PE1# set neighbor 1.1.1.3
      user@PE1# set neighbor 1.1.1.4
      ```

   d. Configure the same BGP group (**toInternal**) on Devices PE2 and PE3 by modifying the **local-address** and **neighbor** statements accordingly.

3. Configure routing policies for exporting routes to and from the MPLS network (**send-direct** policy and **next-hop-self** policy) and a policy for load balancing traffic network across the MPLS network (**lb** policy).

   a. Configure a policy (**send-direct**) for exporting routes from the routing table into BGP on Device CE1.

      ```
      [edit policy-options policy-statement send-direct]
      user@CE1# set from protocol direct
      user@CE1# set then accept

      [edit protocols bgp group toPE1]
      user@CE1# set export send-direct
      ```

      Similarly, configure the **send-direct** policy on Devices CE2 and CE3.

b. Configure a policy (**next-hop-self**) for creating a label-to-next-hop mapping in the MPLS forwarding table on Device PE3.

See BGP Next-Hop-Self Overview for more information.

```
[edit policy-options policy-statement next-hop-self]
user@PE3# set then next-hop self

[edit protocols bgp group toInternal ]
user@PE3# set export next-hop-self
```

c. Configure a policy (**lb**) for exporting routes from the routing table into the forwarding table on Device PE1.

The **lb** policy configures per-packet load balancing, which ensures that all next-hop addresses for a destination are installed in the forwarding table.

```
[edit policy-options policy-statement lb]
user@PE1# set then load-balance per-packet

[edit routing-options]
user@PE1# set forwarding-table export lb
```

Similarly, configure the **lb** policy on Devices CE2, PE2, and PE3.

4. Configure the following:

a. Configure the routing instance on the PE devices for exporting routes through the autonomous systems.

b. Include the **equal-external-internal** statement at the **[edit routing-instances instance-name routing-options multipath vpn-unequal-cost]** hierarchy level to enable load balancing in the network.

c. Include the **vrf-target-label** statement at the **[edit routing-instances instance-name]** hierarchy level for filtering traffic prior to exiting the egress device (Device CE3).

Device PE1

```
[edit routing-instances purple]
user@PE1# set instance-type vrf
user@PE1# set interface ge-2/1/10.2
user@PE1# set route-distinguisher 2:1
user@PE1# set vrf-target target:2:1
user@PE1# set vrf-table-label
user@PE1# set protocols bgp group toCE1 type external
user@PE1# set protocols bgp group toCE1 peer-as 1
user@PE1# set protocols bgp group toCE1 neighbor 10.1.1.1
user@PE1# set routing-options multipath vpn-unequal-cost equal-external-internal
```

Device PE2

```
[edit routing-instances purple]
user@PE2# set instance-type vrf
user@PE2# set interface ge-2/1/10.21
user@PE2# set route-distinguisher 2:1
user@PE2# set vrf-target target:2:1
user@PE2# set vrf-table-label
```

```
user@PE2# set protocols bgp group toCE2 type external
user@PE2# set protocols bgp group toCE2 peer-as 3
user@PE2# set protocols bgp group toCE2 neighbor 10.1.6.22
user@PE2# set routing-options multipath vpn-unequal-cost equal-external-internal
```

Device PE3

```
[edit routing-instances purple]
user@PE3# set instance-type vrf
user@PE3# set interface ge-2/1/10.25
user@PE3# set interface ge-2/1/10.29
user@PE3# set route-distinguisher 2:1
user@PE3# set vrf-target target:2:1
user@PE3# set vrf-table-label
user@PE3# set protocols bgp group toCE2 type external
user@PE3# set protocols bgp group toCE2 peer-as 3
user@PE3# set protocols bgp group toCE2 neighbor 10.1.7.26
user@PE3# set protocols bgp group toCE3 type external
user@PE3# set protocols bgp group toCE3 peer-as 4
user@PE3# set protocols bgp group toCE3 neighbor 10.1.8.30
user@PE3# set routing-options multipath vpn-unequal-cost equal-external-internal
```

Results From configuration mode, confirm your configuration by entering the **show interfaces**, **show protocols**, **show policy-options**, **show routing-options**, and **show routing-instances** commands. If the output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
user@PE3# show interfaces
ge-2/1/10 {
    unit 10 {
        description toPE1;
        family inet {
            address 10.1.3.10/30;
        }
        family mpls
    }
    unit 18 {
        description toP;
        family inet {
            address 10.1.5.18/30;
        }
        family mpls
    }
    unit 25 {
        description toCE2;
        family inet {
            address 10.1.7.25/30;
        }
        family mpls
    }
    unit 29 {
        description toCE3;
        family inet {
            address 10.1.8.29/30;
        }
        family mpls
```



```
    }
  }
  lo0 {
    unit 7 {
      family inet {
        address 1.1.1.4/32;
      }
    }
  }
```

```
user@PE3# show protocols
mpls {
    interface all;
}
bgp {
    group toInternal {
        type internal;
        local-address 1.1.1.4;
        family inet {
            unicast;
        }
        family inet-vpn {
            unicast;
        }
        family route-target;
        export next-hop-self;
        neighbor 1.1.1.2;
        neighbor 1.1.1.3;
    }
}
ospf {
    area 0.0.0.0 {
        interface lo0.7 {
            passive;
        }
        interface ge-2/1/10.10 {
            metric 10;
        }
        interface ge-2/1/10.18 {
            metric 5;
        }
    }
}
ldp {
    interface all;
}
```

```
user@PE3# show policy-options
policy-statement lb {
    then {
        load-balance per-packet;
    }
}
policy-statement next-hop-self {
    then {
        next-hop self;
    }
```

```
}
```

```
user@PE3# show routing-instances
purple {
    instance-type vrf;
    interface ge-2/1/10.25;
    interface ge-2/1/10.29;
    route-distinguisher 2:1;
    vrf-target target:2:1;
    vrf-table-label;
    routing-options {
        multipath {
            vpn-unequal-cost equal-external-internal;
        }
    }
    protocols {
        bgp {
            group toCE2 {
                type external;
                peer-as 3;
                neighbor 10.1.7.26;
            }
            group toCE3 {
                type external;
                peer-as 4;
                neighbor 10.1.8.30;
            }
        }
    }
}
```

```
user@PE3# show routing-options
router-id 1.1.1.4;
autonomous-system 2;
forwarding-table {
    export lb;
}
```

If you are done configuring the device, enter **commit** from configuration mode.

Verification

Confirm that the configuration is working properly.

- Verifying BGP on page 62
- Verifying Next Hops on page 64
- Verifying Load Balancing on page 65
- Verifying Load Balancing Using IP Header Filtering on page 68

Verifying BGP

Purpose Verify that BGP is working.

Action From operational mode, run the **show route protocol bgp** command.

```
user@PE3> show route protocol bgp

inet.0: 11 destinations, 11 routes (11 active, 0 holddown, 0 hidden)

inet.3: 3 destinations, 3 routes (3 active, 0 holddown, 0 hidden)

purple.inet.0: 9 destinations, 14 routes (9 active, 0 holddown, 0 hidden)
@ = Routing Use Only, # = Forwarding Use Only
+ = Active Route, - = Last Active, * = Both

1.1.1.1/32         *[BGP/170] 04:47:14, localpref 100, from 1.1.1.2
                      AS path: 1 I
                    > to 10.1.3.9 via ge-2/1/10.10, Push 16
1.1.1.6/32         @[BGP/170] 00:13:28, localpref 100
                      AS path: 3 I
                    > to 10.1.7.26 via ge-2/1/10.25
                     [BGP/170] 00:10:36, localpref 100, from 1.1.1.3
                      AS path: 3 I
                    > to 10.1.5.17 via ge-2/1/10.18, Push 16, Push 299776(top)
1.1.1.7/32         *[BGP/170] 00:10:56, localpref 100
                      AS path: 4 I
                    > to 10.1.8.30 via ge-2/1/10.29
10.1.1.0/30        *[BGP/170] 04:47:14, localpref 100, from 1.1.1.2
                      AS path: I
                    > to 10.1.3.9 via ge-2/1/10.10, Push 16
10.1.6.20/30       *[BGP/170] 04:47:03, localpref 100, from 1.1.1.3
                      AS path: I
                    > to 10.1.5.17 via ge-2/1/10.18, Push 16, Push 299776(top)
                     [BGP/170] 00:13:28, localpref 100
                      AS path: 3 I
                    > to 10.1.7.26 via ge-2/1/10.25
10.1.7.24/30        [BGP/170] 00:13:28, localpref 100
                      AS path: 3 I
                    > to 10.1.7.26 via ge-2/1/10.25
10.1.8.28/30        [BGP/170] 00:10:56, localpref 100
                      AS path: 4 I
                    > to 10.1.8.30 via ge-2/1/10.29

mpls.0: 9 destinations, 9 routes (9 active, 0 holddown, 0 hidden)

bgp.l3vpn.0: 4 destinations, 4 routes (4 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

2:1:1.1.1.1/32
                   *[BGP/170] 04:47:14, localpref 100, from 1.1.1.2
                      AS path: 1 I
                    > to 10.1.3.9 via ge-2/1/10.10, Push 16
2:1:1.1.1.6/32
                   *[BGP/170] 00:10:36, localpref 100, from 1.1.1.3
                      AS path: 3 I
                    > to 10.1.5.17 via ge-2/1/10.18, Push 16, Push 299776(top)
2:1:10.1.1.0/30
                   *[BGP/170] 04:47:14, localpref 100, from 1.1.1.2
                      AS path: I
                    > to 10.1.3.9 via ge-2/1/10.10, Push 16
2:1:10.1.6.20/30
                   *[BGP/170] 04:47:03, localpref 100, from 1.1.1.3
```

The output lists the BGP routes installed into the routing table. The lines of output that start with **1.1.1.1/32**, **10.1.1.0/30**, and **2:1:1.1.1.1/32** show the BGP routes to Device CE1,

which is in AS1. The lines of output that start with **1.1.1.6/32**, **2:1:1.1.1.6/32**, and **2:1:10.1.6.20/30** show the BGP routes to Device CE2, which is in AS3. The line of output that starts with **1.1.1.7/32** shows the BGP route to Device CE3, which is in AS4.

Meaning BGP is functional in the network.

Verifying Next Hops

Purpose Verify that a route has multiple next hops.

Action From operational mode, run the **show route** command.

```
user@CE2> show route 10.1.1.1

inet.0: 9 destinations, 14 routes (9 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

10.1.1.0/30        *[BGP/170] 00:15:52, localpref 100, from 10.1.7.25
                      AS path: 2 I
                    > to 10.1.6.21 via ge-2/1/10.22
                      to 10.1.7.25 via ge-2/1/10.26
                    [BGP/170] 00:13:00, localpref 100
                      AS path: 2 I
                    > to 10.1.6.21 via ge-2/1/10.22
```

The output shows that for a route from Device CE2 to Device CE1, or the **10.1.1.0** network, there are two next hops. One is **10.1.6.21** through the **ge-2/1/10.22** interface, and another is **10.1.7.25** through the **ge-2/1/10.26** interface.

```
user@CE2> show route 10.1.1.1 detail

inet.0: 9 destinations, 14 routes (9 active, 0 holddown, 0 hidden)
10.1.1.0/30 (2 entries, 1 announced)
        *BGP    Preference: 170/-101
                Next hop type: Router, Next hop index: 1048584
                Address: 0x92a0078
                Next-hop reference count: 6
                Source: 10.1.7.25
                Next hop: 10.1.6.21 via ge-2/1/10.22, selected
                Next hop: 10.1.7.25 via ge-2/1/10.26
                State: (Active Ext)
                Local AS:     3 Peer AS:     2
                Age: 17:02
                Task: BGP_2.10.1.7.25+59418
                Announcement bits (1): 0-KRT
                AS path: 2 I
                Communities: target:2:1
                Accepted Multipath
                Localpref: 100
                Router ID: 10.1.7.25
         BGP    Preference: 170/-101
                Next hop type: Router, Next hop index: 1310
                Address: 0x91f0208
                Next-hop reference count: 6
                Source: 10.1.6.21
                Next hop: 10.1.6.21 via ge-2/1/10.22, selected
                State: (NotBest Ext)
                Inactive reason: Not Best in its group - Active preferred
```

```
Local AS:     3 Peer AS:     2
Age: 14:10
Task: BGP_2.10.1.6.21+58156
AS path: 2 I
Communities: target:2:1
Accepted MultipathContrib
Localpref: 100
Router ID: 10.1.6.21
```

The detailed output listed here shows that out of the two available next hops for a route to the **10.1.1.0** network, **10.1.6.21** is the preferred next hop.

Meaning A route has multiple next hops through the network, which indicates that the network is configured correctly for load balancing to work.

Verifying Load Balancing

Purpose Verify that forwarding is taking place in both directions by checking:

- If both next hops are installed in the forwarding table for a route.
- If external BGP routes are installed in the forwarding table for a route.

Action From operational mode, run the **show route forwarding-table** and **show route forwarding-table destination <destination IP>** commands.

```
user@PE3> show route forwarding-table

Router: PE3
Routing table: default.inet
Internet:
Destination        Type RtRef Next hop           Type Index NhRef Netif
default            perm     0                    rjct   593     1
0.0.0.0/32         perm     0                    dscd   579     1
1.1.1.2/32         user     1 10.1.3.9           ucst   999     8 ge-2/1/10.10
1.1.1.3/32         user     1 10.1.5.17          ucst  1243    12 ge-2/1/10.18
1.1.1.4/32         intf     0 1.1.1.4            locl   895     1
1.1.1.5/32         user     1 10.1.5.17          ucst  1243    12 ge-2/1/10.18
10.1.2.4/30        user     0                    ulst 1048580     2
                              10.1.3.9          ucst   999     8 ge-2/1/10.10
                              10.1.5.17         ucst  1243    12 ge-2/1/10.18
10.1.3.8/30        intf     0                    rslv   899     1 ge-2/1/10.10
10.1.3.8/32        dest     0 10.1.3.8           recv   897     1 ge-2/1/10.10
10.1.3.9/32        dest     0 0.6.80.3.0.21.59.d.c5.d9.0.21.59.d.c5.da.8.0
                                                 ucst   999     8 ge-2/1/10.10
10.1.3.10/32       intf     0 10.1.3.10          locl   898     2
10.1.3.10/32       dest     0 10.1.3.10          locl   898     2
10.1.3.11/32       dest     0 10.1.3.11          bcst   896     1 ge-2/1/10.10
10.1.4.12/30       user     0 10.1.5.17          ucst  1243    12 ge-2/1/10.18
10.1.5.16/30       intf     0                    rslv   903     1 ge-2/1/10.18
10.1.5.16/32       dest     0 10.1.5.16          recv   901     1 ge-2/1/10.18
10.1.5.17/32       dest     0 0.e.80.3.0.21.59.d.c5.d9.0.21.59.d.c5.da.8.0
                                                 ucst  1243    12 ge-2/1/10.18
10.1.5.18/32       intf     0 10.1.5.18          locl   902     2
10.1.5.18/32       dest     0 10.1.5.18          locl   902     2
10.1.5.19/32       dest     0 10.1.5.19          bcst   900     1 ge-2/1/10.18
224.0.0.0/4        perm     2                    mdsc   592     1
224.0.0.1/32       perm     0 224.0.0.1          mcst   576     3
```

```
224.0.0.5/32       user     1 224.0.0.5          mcst   576     3
255.255.255.255/32 perm     0                    bcst   577     1

Router: PE3
Routing table: __master.anon__.inet
Internet:
Destination        Type RtRef Next hop           Type Index NhRef Netif
default            perm     0                    rjct   909     1
0.0.0.0/32         perm     0                    dscd   907     1
224.0.0.0/4        perm     0                    mdsc   908     1
224.0.0.1/32       perm     0 224.0.0.1          mcst   904     1
255.255.255.255/32 perm     0                    bcst   905     1

Router:  PE3
Routing table: purple.inet
Internet:
Destination        Type RtRef Next hop           Type Index NhRef Netif
default            perm     0                    rjct   918     1
0.0.0.0/32         perm     0                    dscd   916     1
1.1.1.1/32         user     0                    indr 1048576     3
                              10.1.3.9          Push 16  1187     2 ge-2/1/10.10
1.1.1.6/32         user     0                    ulst 1048587     2
                              10.1.7.26          ucst  1239     4 ge-2/1/10.25
                                                 indr 1048579     3
                              10.1.5.17         Push 16, Push 299776(top)  1306
    2 ge-2/1/10.18
1.1.1.7/32         user     0 10.1.8.30          ucst  1299     4 ge-2/1/10.29
299808(S=0)        user     0 10.1.5.17         Pop    1304     2 ge-2/1/10.18
...
```

In the **default.inet** routing table, which is the internal routing table, the line of output that starts with **10.1.2.4/30** shows that for a route to Device PE2 in the same AS, two next hops are installed in the table: **10.1.3.9** and **10.1.5.17**.

In the **purple.inet** routing table, which is the external routing table, the line of output that starts with **1.1.1.6/32** shows that for a route to Device CE2 in AS3, an internal next hop of **10.1.5.17** and an external next hop of **10.1.7.26** are installed in the table. This indicates that both internal and external BGP routes are operational in the network.

```
user@PE3> show route forwarding-table destination 10.1.2.6

Router: PE3
Routing table: default.inet
Internet:
Destination        Type RtRef Next hop           Type Index NhRef Netif
10.1.2.4/30        user     0                    ulst 1048580     2
                              10.1.3.9          ucst   999     8 ge-2/1/10.10
                              10.1.5.17         ucst  1243    12 ge-2/1/10.18

Router:  PE3
Routing table: __master.anon__.inet
Internet:
Destination        Type RtRef Next hop           Type Index NhRef Netif
default            perm     0                    rjct   909     1

Router:  PE3
Routing table: purple.inet
Internet:
```

```
Destination        Type RtRef Next hop           Type Index NhRef Netif
default            perm     0                    rjct   918     1
```

The line of output that starts with **10.1.2.4/30** shows that for a route from Device PE3 to Device PE2 in the same AS, two next hops are installed in the table: **10.1.3.9** through the **ge-2/1/10.10** interface, and **10.1.5.17** through the **ge-2/1/10.18** interface.

```
user@CE2> show route forwarding-table

Router: CE2
Routing table: default.inet
Internet:
Destination        Type RtRef Next hop           Type Index NhRef Netif
default            perm     0                    rjct  1194     1
0.0.0.0/32         perm     0                    dscd  1192     1
1.1.1.1/32         user     0                    ulst 1048584     4
                              10.1.6.21          ucst  1310     5 ge-2/1/10.22
                              10.1.7.25          ucst  1240     4 ge-2/1/10.26
1.1.1.6/32         intf     0 1.1.1.6            locl   600     1
1.1.1.7/32         user     0                    ulst 1048584     4
                              10.1.6.21          ucst  1310     5 ge-2/1/10.22
                              10.1.7.25          ucst  1240     4 ge-2/1/10.26
10.1.1.0/30        user     0                    ulst 1048584     4
                              10.1.6.21          ucst  1310     5 ge-2/1/10.22
                              10.1.7.25          ucst  1240     4 ge-2/1/10.26
10.1.6.20/30       intf     0                    rslv  1202     1 ge-2/1/10.22
10.1.6.20/32       dest     0 10.1.6.20          recv   602     1 ge-2/1/10.22
10.1.6.21/32       dest     1 0.8.80.3.0.21.59.d.c5.d9.0.21.59.d.c5.da.8.0
                                                 ucst  1310     5 ge-2/1/10.22
10.1.6.22/32       intf     0 10.1.6.22          locl   603     2
10.1.6.22/32       dest     0 10.1.6.22          locl   603     2
10.1.6.23/32       dest     0 10.1.6.23          bcst   601     1 ge-2/1/10.22
10.1.7.24/30       intf     0                    rslv   629     1 ge-2/1/10.26
10.1.7.24/32       dest     0 10.1.7.24          recv   627     1 ge-2/1/10.26
10.1.7.25/32       dest     1 0.17.80.3.0.21.59.d.c5.d9.0.21.59.d.c5.da.8.0
                                                 ucst  1240     4 ge-2/1/10.26
10.1.7.26/32       intf     0 10.1.7.26          locl   628     2
10.1.7.26/32       dest     0 10.1.7.26          locl   628     2
10.1.7.27/32       dest     0 10.1.7.27          bcst   608     1 ge-2/1/10.26
10.1.8.28/30       user     0 10.1.6.21          ucst  1310     5 ge-2/1/10.22
224.0.0.0/4        perm     0                    mdsc  1193     1
224.0.0.1/32       perm     0 224.0.0.1          mcst  1189     1
255.255.255.255/32 perm     0                    bcst  1190     1

Router: CE2
Routing table: __master.anon__.inet
Internet:
Destination        Type RtRef Next hop           Type Index NhRef Netif
default            perm     0                    rjct  1208     1
0.0.0.0/32         perm     0                    dscd  1206     1
224.0.0.0/4        perm     0                    mdsc  1207     1
224.0.0.1/32       perm     0 224.0.0.1          mcst  1001     1
255.255.255.255/32 perm     0                    bcst  1204     1
...
```

The lines of output starting with **1.1.1.1/32** and **10.1.1.0/30** show the external BGP routes from Device CE2 in AS3 to Device CE1 in AS1 through next hops of **10.1.6.21** and **10.1.7.25** in AS2.

```
user@CE2> show route forwarding-table destination 10.1.1.1
```

```
Router: CE2
Routing table: default.inet
Internet:
Destination        Type RtRef Next hop           Type Index NhRef Netif
10.1.1.0/30        user     0                    ulst 1048584     4
                              10.1.6.21          ucst  1310     5 ge-2/1/10.22
                              10.1.7.25          ucst  1240     4 ge-2/1/10.26

Router: CE2
Routing table: __master.anon__.inet
Internet:
Destination        Type RtRef Next hop           Type Index NhRef Netif
default            perm     0                    rjct  1208     1
```

The line of output starting with **10.1.1.0/30** shows the external BGP routes to Device CE1 through next hops of **10.1.6.21** through the **ge-2/1/10.22** interface, and **10.1.7.25** through the **ge-2/1/10.26** interface.

Meaning Multiple next hops for a route, including external BGP routes, are installed in the forwarding tables.

Verifying Load Balancing Using IP Header Filtering

Purpose Verify that filtered traffic reaches the egress CE devices after load balancing has been configured on the PE devices.

Action Configure a firewall filter on Device PE3 on the interface connecting to Device CE2.

```
[edit firewall family inet filter filterPE3 term a]
user@PE3# set from protocol tcp
user@PE3# set from source-port-except bgp
user@PE3# set from destination-port-except bgp
user@PE3# set then count filterPE3
user@PE3# set then accept

[edit firewall family inet filter filterPE3 term b]
user@PE3# set then accept

[edit interfaces ge-2/1/10 unit 25]
user@PE3# set family inet filter output filterPE3
```

Similarly, configure a firewall filter on Device PE3 on the interface facing Device CE3, and another on Device PE2 on the interface facing Device CE2.

Count the packets exiting the egress interfaces on Devices PE2 and PE3 by using the **show firewall filter** *<filter name>* **counter** *<counter name>* operational mode command. The output confirms if load balancing takes place with IP header filtering configured (enabled by the **vrf-table-label** statement). If all transmitted packets have been load-balanced between the paths PE3->CE2, PE3->CE3, and PE2->CE2, then it means that the IP header filtering feature works in a load-balanced Layer 3 network.

You can clear the counter by using the **clear firewall filter** *<filter name>* **counter** *<counter name>* operational mode command.

Meaning Load balancing takes place with IP header filtering configured.

Related Documentation

- Configuring Load Balancing for Layer 3 VPNs on page 83
- Configuring VRF Table Labels
- Example: Load Balancing BGP Traffic
- Load Balancing and IP Header Filtering for Layer 3 VPNs on page 51
- load-balance
- multipath
- vpn-unequal-cost on page 361
- vrf-table-label

Configuring a Label Allocation and Substitution Policy for VPNs

You can control label-advertisements on MPLS ingress and AS border routers (ASBRs). Labels can be assigned on a per–next-hop (by default) or on a per-table basis (by configuring the vrf-table-label statement). This choice affects all routes of a given routing instance. You can also configure a policy to generate labels on a per-route basis by specifying a label allocation policy.

To specify a label allocation policy for the routing instance, configure the **label** statement and specify a label allocation policy using the **allocation** option:

```
label {
  allocation label-allocation-policy;
}
```

You can configure this statement at the following hierarchy levels:

- **[edit routing-instances *routing-instance-name* routing-options]**
- **[edit logical-systems *logical-system-name* routing-instances *routing-instance-name* routing-options]**

To configure the label allocation policy, include the **label-allocation** statement at the **[edit policy-options policy-statement *policy-statement-name* term *term-name* then]** hierarchy level. You can configure the label allocation mode as either **per-nexthop** or **per-table**.

For a VPN option B ASBR, labels for transit routes are substituted for a local virtual tunnel label or vrf-table-label label. When a VRF table is configured on the ASBR (this type of configuration is uncommon for the option B model), the ASBR does not generate MPLS swap or swap and push state for transit routes. Instead, the ASBR re-advertises a local virtual-tunnel or vrf-table-label label and forwards that transit traffic based on IP forwarding tables. The label substitution helps to conserve labels on Juniper Networks routers.

However, this type of label substitution effectively breaks the MPLS forwarding path, which becomes visible when using an MPLS OAM command such as LSP ping. You can configure the way in which labels are substituted on a per-route basis by specifying a label substitution policy.

To specify a label substitution policy for the routing instance, configure the **label** statement and specify a label substitution policy using the **substitution** option:

```
label {
  substitution label-substitution-policy;
}
```

You can configure this statement at the following hierarchy levels:

- **[edit routing-instances *routing-instance-name* routing-options]**
- **[edit logical-systems *logical-system-name* routing-instances *routing-instance-name* routing-options]**

The label substitution policy is used to determine whether or not a label should be substituted on an ASBR router. The results of the policy operation are either **accept** (label substitution is performed) or **reject** (label substitution is not performed). The default behavior is **accept**. The following set command example illustrates how you can configure a **reject** label substitution policy: **set policy-options policy-statement no-label-substitution term default then reject**.

Configuring Logical Units on the Loopback Interface for Routing Instances in Layer 3 VPNs

For Layer 3 VPNs (VRF routing instances), you can configure a logical unit on the loopback interface into each VRF routing instance that you have configured on the router. Associating a VRF routing instance with a logical unit on the loopback interface allows you to easily identify the VRF routing instance.

Doing this is useful for troubleshooting:

- It allows you to ping a remote CE router from a local PE router in a Layer 3 VPN. For more information, see "Pinging the Remote CE Router from the Local PE Router" on page 378.
- It ensures that a path maximum transmission unit (MTU) check on traffic originating on a VRF or virtual-router routing instance functions properly. For more information, see Configuring Path MTU Checks for VPNs.

You can also configure a firewall filter for the logical unit on the loopback interface; this configuration allows you to filter traffic for the VRF routing instance associated with it.

The following describes how firewall filters affect the VRF routing instance depending on whether they are configured on the default loopback interface, the VRF routing instance, or some combination of the two. The "default loopback interface" refers to **lo0.0** (associated with the default routing table), and the "VRF loopback interface" refers to **lo0.*n***, which is configured in the VRF routing instance.

- If you configure Filter A on the default loopback interface and Filter B on the VRF loopback interface, the VRF routing instance uses Filter B.
- If you configure Filter A on the default loopback interface but do not configure a filter on the VRF loopback interface, the VRF routing instance does not use a filter.
- If you configure Filter A on the default loopback interface but do not even configure a VRF loopback interface, the VRF routing instance uses Filter A.

To configure a logical unit on the loopback interface, include the **unit** statement:

```
unit number {
  family inet {
    address address;
  }
}
```

You can include this statement at the following hierarchy levels:

- **[edit interfaces lo0]**
- **[edit logical-systems *logical-system-name* interfaces lo0]**

To associate a firewall filter with the logical unit on the loopback interface, include the **filter** statement:

```
filter {
  input filter-name;
}
```

You can include this statement at the following hierarchy levels:

- **[edit interfaces lo0 unit *unit-number* family inet]**
- **[edit logical-systems *logical-system-name* interfaces lo0 unit *unit-number* family inet]**

To include the **lo0.*n*** interface (where ***n*** specifies the logical unit) in the configuration for the VRF routing instance, include the following statement:

```
interface lo0.n;
```

You can include this statement at the following hierarchy levels:

- **[edit routing-instances *routing-instance-name*]**
- **[edit logical-systems *logical-system-name* routing-instances *routing-instance-name*]**

For more information about how to configure firewall filters, see the *Junos OS Policy Framework Configuration Guide*.

Configuring Multicast Layer 3 VPNs

You can configure two types of multicast Layer 3 VPNs using the Junos OS:

- Draft Rosen multicast VPNs—Draft Rosen multicast VPNs are described in RFC 4364, *BGP/MPLS IP Virtual Private Networks (VPNs)* and based on Section Two of the IETF

Internet draft draft-rosen-vpn-mcast-06.txt, *Multicast in MPLS/BGP VPNs* (expired April 2004).

- Next generation multicast VPNs—Next generation multicast VPNs are described in Internet drafts draft-ietf-l3vpn-2547bis-mcast-bgp-03.txt, *BGP Encodings for Multicast in MPLS/BGP IP VPNs* and draft-ietf-l3vpn-2547bis-mcast-02.txt, *Multicast in MPLS/BGP IP VPNs*.

This section describes how to configure draft Rosen multicast VPNs. This information is provided to you in case you already have dual PIM multicast VPNs configured on your network. For information about BGP MPLS multicast VPNs (also known as next generation multicast VPNs), see MBGP Multicast VPN Sites.

NOTE: **Draft-rosen multicast VPNs are not supported in a logical system environment even though the configuration statements can be configured under the logical-systems hierarchy.**

You can configure a Layer 3 VPN to support multicast traffic using the Protocol Independent Multicast (PIM) routing protocol. To support multicast, you need to configure PIM on routers within the VPN and within the service provider's network.

Each PE router configured to run multicast over Layer 3 VPNs must have a Tunnel Services PIC. A Tunnel Services PIC is also required on the P routers that act as rendezvous points (RPs). Tunnel Services PICs are also needed on all the CE routers acting as designated routers (first-hop/last-hop routers) or as RPs, just as they are in non-VPN PIM environments.

Configure the master PIM instance at the **[edit protocols pim]** hierarchy level on the CE and PE routers. This master PIM instance configuration on the PE router should match the configuration on the service providers core routers.

You also need to configure a PIM instance for the Layer 3 VPN at the **[edit routing-instances *routing-instance-name* protocols pim]** hierarchy level on the PE router. This creates a PIM instance for the indicated routing instance. The configuration of the PIM instance on the PE router should match the PIM instance configured on the CE router the PE router is connected to.

For information about how to configure PIM, see the *Junos OS Multicast Protocols Configuration Guide*.

Include the **vpn-apply-export** statement to configure the group address designated for the VPN in the service provider's network. This address must be unique for each VPN and configured on the VRF routing instance of all PE routers connecting to the same VPN. It ensures that multicast traffic is transmitted only to the specified VPN.

Include the **vpn-apply-export** statement:

vpn-apply-export *address*;

You can include this statement at the following hierarchy levels:

- **[edit routing-instances *routing-instance-name* protocols pim]**
- **[edit logical-systems *logical-system-name* routing-instances *routing-instance-name* protocols pim]**

The rest of the Layer 3 VPN configuration for multicast is conventional and is described in other sections of this manual. Most of the specific configuration tasks needed to activate multicast in a VPN environment involve PIM. For more information about how to configure PIM and multicast in Junos, including an example of how to configure multicast over Layer 3 VPNs, see the *Junos OS Multicast Protocols Configuration Guide*.

Configuring Packet Forwarding for Layer 3 VPNs

You can configure the router to support packet forwarding for IPv4 traffic in Layer 2 and Layer 3 VPNs. Packet forwarding is handled in one of the following ways, depending on the type of helper service configured:

- BOOTP service—Clients send Bootstrap Protocol (BOOTP) requests through the router configured with BOOTP service to a server in the specified routing instance. The server recognizes the client address and sends a response back to the router configured with BOOTP service. This router forwards the reply to the correct client address in the specified routing instance.
- Other services—Clients send requests through the router configured with the service to a server in the specified routing instance. The server recognizes the client address and sends a response to the correct client address in the specified routing instance.

To enable packet forwarding for VPNs, include the **helpers** statement:

```
helpers {
  service {
    description description-of-service;
    server {
      address address {
        routing-instance routing-instance-names;
      }
    }
    interface interface-name {
      description description-of-interface;
      no-listen;
      server {
        address address {
          routing-instance routing-instance-names;
        }
      }
    }
  }
}
```

You can include this statement at the following hierarchy levels:

- **[edit forwarding-options]**

- [edit logical-systems *logical-system-name* forwarding-options]
- [edit routing-instances *routing-instance-name* forwarding-options]

NOTE: You can enable packet forwarding for multiple VPNs. However, the client and server must be within the same VPN. Any Juniper Networks routing platforms with packet forwarding enabled along the path between the client and server must also reside within the same VPN.

The address and routing instance together constitute a unique server. This has implications for routers configured with BOOTP service, which can accept multiple servers.

For example, a BOOTP service can be configured as follows:

```
[edit forwarding-options helpers bootp]
server address 10.2.3.4 routing-instance [instance-A instance-B];
```

Even though the addresses are identical, the routing instances are different. A packet coming in for BOOTP service on **instance-A** is forwarded to **10.2.3.4** in the **instance-A** routing instance, while a packet coming in on **instance-B** is forwarded in the **instance-B** routing instance. Other services can only accept a single server, so this configuration does not apply in those cases.

For more information about the statements configured at the **[edit forwarding-options]** hierarchy level, see the *Junos OS Policy Framework Configuration Guide*.

Configuring GRE Tunnels for Layer 3 VPNs

Junos OS allows you to configure a generic routing encapsulation (GRE) tunnel between the PE and CE routers for a Layer 3 VPN. The GRE tunnel can have one or more hops. You can configure the tunnel from the PE router to a local CE router (as shown in Figure 14 on page 74) or to a remote CE router (as shown in Figure 15 on page 75).

Figure 14: GRE Tunnel Configured Between the Local CE Router and the PE Router

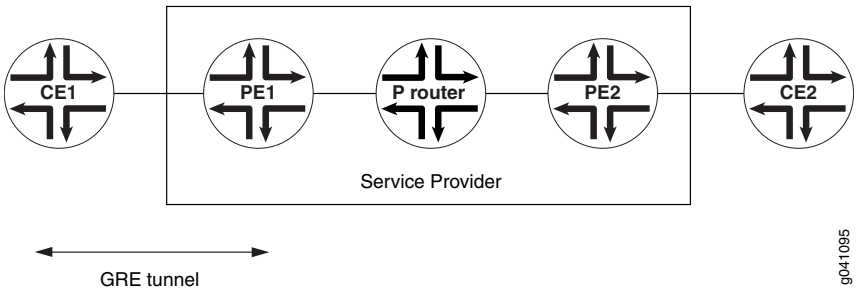
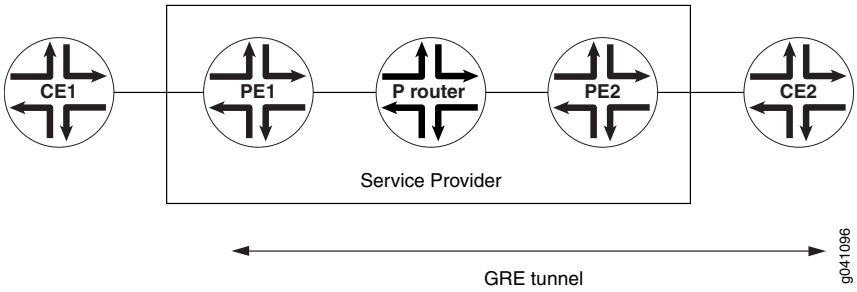


Figure 15: GRE Tunnel Configured Between the Remote CE Router and the PE Router



For more information about how to configure tunnel interfaces, see the *Junos OS Services Interfaces Configuration Guide*.

You can configure the GRE tunnels manually or configure the Junos OS to instantiate GRE tunnels dynamically.

The following sections describe how to configure GRE tunnels manually and dynamically:

- Configuring GRE Tunnels Manually Between PE and CE Routers on page 75
- Configuring GRE Tunnels Dynamically on page 76

Configuring GRE Tunnels Manually Between PE and CE Routers

You can manually configure a GRE tunnel between a PE router and either a local CE router or a remote CE router for a Layer 3 VPN as explained in the following sections:

- Configuring the GRE Tunnel Interface on the PE Router on page 75
- Configuring the GRE Tunnel Interface on the CE Router on page 76

Configuring the GRE Tunnel Interface on the PE Router

You configure the GRE tunnel as a logical interface on the PE router. To configure the GRE tunnel interface, include the **unit** statement:

```
unit logical-unit-number {
  tunnel {
    source source-address;
    destination destination-address;
    routing-instance {
      destination routing-instance-name;
    }
  }
  family inet {
    address address;
  }
}
```

You can include this statement at the following hierarchy levels:

- **[edit interfaces** *interface-name***]**
- **[edit logical-systems** *logical-system-name* **interfaces** *interface-name***]**

As part of the GRE tunnel interface configuration, you need to include the following statements:

- **source** *source-address*—Specify the source or origin of the GRE tunnel, typically the PE router.
- **destination** *destination-address*—Specify the destination or end point of the GRE tunnel. The destination can be a Provider router, the local CE router, or the remote CE router.

By default, the tunnel destination address is assumed to be in the default Internet routing table, **inet.0**. If the tunnel destination address is not in **inet.0**, you need to specify which routing table to search for the tunnel destination address by configuring the **routing-instance** statement. This is the case if the tunnel encapsulating interface is also configured under the routing instance.

- **destination** *routing-instance-name*—Specify the name of the routing instance when configuring the GRE tunnel interface on the PE router.

To complete the GRE tunnel interface configuration, include the **interface** statement for the GRE interface under the appropriate routing instance:

```
interface interface-name;
```

You can include this statement at the following hierarchy levels:

- **[edit routing-instances** *routing-instance-name***]**
- **[edit logical-systems** *logical-system-name* **routing-instances** *routing-instance-name***]**

Configuring the GRE Tunnel Interface on the CE Router

You can configure either the local or the remote CE router to act as the endpoint for the GRE tunnel.

To configure the GRE tunnel interface on the CE router, include the **unit** statement:

```
unit logical-unit-number {
    tunnel {
        source address;
        destination address;
    }
    family inet {
        address address;
    }
}
```

You can include this statement at the following hierarchy levels:

- **[edit interfaces** *interface-name***]**
- **[edit logical-systems** *logical-system-name* **interfaces** *interface-name***]**

Configuring GRE Tunnels Dynamically

When the router receives a VPN route to a BGP next-hop address, but no MPLS path is available, a GRE tunnel can be dynamically generated to carry the VPN traffic across the

BGP network. The GRE tunnel is generated and then its routing information is copied into the inet.3 routing table. IPv4 routes are the only type of routes supported for dynamic GRE tunnels. Also, the routing platform must have a tunnel PIC.

NOTE: When configuring a dynamic GRE tunnel to a remote CE router, do not configure OSPF over the tunnel interface. It creates a routing loop forcing the router to take the GRE tunnel down. The router attempts to reestablish the GRE tunnel, but will be forced to take it down again when OSPF becomes active on the tunnel interface and discovers a route to the tunnel endpoint. This is not an issue when configuring static GRE tunnels to a remote CE router.

To generate GRE tunnels dynamically, include the **dynamic-tunnels** statement:

```
dynamic-tunnels tunnel-name {
    destination-networks prefix;
    source-address address;
    tunnel-type gre;
}
```

You can include this statement at the following hierarchy levels:

- **[edit routing-options]**
- **[edit logical-systems *logical-system-name* routing-options]**

Specify the IPv4 prefix range (for example, **10/8** or **11.1/16**) for the destination network by including the **destination-networks** statement. Only tunnels within the specified IPv4 prefix range are allowed to be initiated.

```
destination-networks prefix;
```

You can include this statement at the following hierarchy levels:

- **[edit routing-options dynamic-tunnels *tunnel-name*]**
- **[edit logical-systems *logical-system-name* routing-options dynamic-tunnels *tunnel-name*]**

Specify the source address for the GRE tunnels by including the **source-address** statement. The source address specifies the address used as the source for the local tunnel endpoint. This could be any local address on the router (typically the router ID or the loopback address).

```
source-address address;
```

You can include this statement at the following hierarchy levels:

- **[edit routing-options dynamic-tunnels *tunnel-name*]**
- **[edit logical-systems *logical-system-name* routing-options dynamic-tunnels *tunnel-name*]**

Specify the type of tunnel to be dynamically created by including the **tunnel-type** statement. The only currently valid value is **gre** (for GRE tunnels).

```
tunnel-type gre;
```

You can include this statement at the following hierarchy levels:

- **[edit routing-options dynamic-tunnels** *tunnel-name***]**
- **[edit logical-systems** *logical-system-name* **routing-options dynamic-tunnels** *tunnel-name***]**

Related Documentation

- Example: Configuring a Two-Tiered Virtualized Data Center for Large Enterprise Networks

Configuring an ES Tunnel Interface for Layer 3 VPNs

An ES tunnel interface allows you to configure an IP Security (IPsec) tunnel between the PE and CE routers of a Layer 3 VPN. The IPsec tunnel can include one or more hops.

The following sections explain how to configure an ES tunnel interface between the PE and CE routers of a Layer 3 VPN:

- Configuring the ES Tunnel Interface on the PE Router on page 78
- Configuring the ES Tunnel Interface on the CE Router on page 79

Configuring the ES Tunnel Interface on the PE Router

To configure the ES tunnel interface on the PE router, include the **unit** statement:

```
unit logical-unit-number {
    tunnel {
        source source-address;
        destination destination-address;
    }
    family inet {
        address address;
        ipsec-sa security-association-name;
    }
}
```

You can include this statement at the following hierarchy levels:

- **[edit interfaces** *interface-name***]**
- **[edit logical-systems** *logical-system-name* **interfaces** *interface-name***]**

By default, the tunnel destination address is assumed to be in the default Internet routing table, **inet.0**. For IPsec tunnels using manual security association (SA), if the tunnel destination address is not in the default **inet.0** routing table, you need to specify which routing table to search for the tunnel destination address by configuring the **routing-instance** statement. This is the case if the tunnel encapsulating interface is also configured under the routing instance.

```
unit logical-unit-number {
    tunnel {
        source address;
        destination address;
        routing-instance {
```

```
        destination routing-instance-name;
      }
      family inet {
        address address;
        ipsec-sa security-association-name;
      }
      family mpls;
    }
  }
```

You can include these statements at the following hierarchy levels:

- **[edit interfaces *interface-name*]**
- **[edit logical-systems *logical-system-name* interfaces *interface-name*]**

NOTE: For IPsec tunnels using dynamic SA, the tunnel destination address must be in the default Internet routing table, **inet.0**.

To complete the ES tunnel interface configuration, include the **interface** statement for the ES interface under the appropriate routing instance:

```
interface interface-name;
```

You can include this statement at the following hierarchy levels:

- **[edit routing-instances *routing-instance-name*]**
- **[edit logical-systems *logical-system-name* routing-instances *routing-instance-name*]**

Configuring the ES Tunnel Interface on the CE Router

To configure the ES tunnel interface on the CE router, include the **unit** statement:

```
unit 0 {
  tunnel {
    source address;
    destination address;
  }
  family inet {
    address address;
    ipsec-sa security-association-name;
  }
}
```

You can include this statement at the following hierarchy levels:

- **[edit interfaces *interface-name*]**
- **[edit logical-systems *logical-system-name* interfaces *interface-name*]**

For more information about how to configure tunnel interfaces, see the *Junos OS Services Interfaces Configuration Guide*.

For more information about how to configure IPsec interfaces, see the *Junos OS System Basics Configuration Guide*.

Configuring IPsec Tunnels Instead of MPLS LSPs Between PE Routers in Layer 3 VPNs

A conventional Layer 3 BGP/MPLS VPN requires the configuration of MPLS label-switched paths (LSPs) between the PE routers. When a PE router receives a packet from a CE router, it performs a lookup in a specific VRF table for the IP destination address and obtains a corresponding MPLS label stack. The label stack is used to forward the packet to the egress PE router, where the bottom label is removed and the packet is forwarded to the specified CE router.

You can provide Layer 3 BGP/MPLS VPN service without an MPLS backbone. Instead of configuring MPLS LSPs between the PE routers, you configure GRE and IPsec tunnels between the PE routers. The MPLS information for the VPN (the VPN label) is encapsulated within an IP header and an IPsec header. The source address of the IP header is the address of the ingress PE router. The destination address has the BGP next hop, the address of the egress PE router.

NOTE: The IPsec tunnel requires the use of an ES PIC. The GRE tunnel requires the use of a Tunnel Services PIC.

To configure IPsec between PE routers, follow these steps:

1. Configure an IPsec tunnel between the PE routers. The source address is that of the ingress PE router, and the destination address is that of the egress PE router:

```
es-interface-name {
  unit unit-number {
    tunnel {
      source source-address;
      destination destination-address;
    }
    family inet {
      ipsec-sa sa-esp-dynamic;
      address address;
    }
    family mpls;
  }
}
```

   You can include these statements at the following hierarchy levels:

   - **[edit interfaces]**
   - **[edit logical-systems *logical-system-name* interfaces]**

2. Configure IPsec on the PE router. For information about how to configure IPsec, see the *Junos OS System Basics Configuration Guide*.
3. Configure a GRE tunnel between the PE routers. Again, the source address is that of the ingress PE router, and the destination address is that of the egress PE router:

```
gr-interface-name {
  unit unit-number {
    family inet {
      address address;
    }
    family mpls;
    tunnel {
      source source-address;
      destination destination-address;
    }
  }
}
```

You can include these statements at the following hierarchy levels:

- **[edit interfaces]**
- **[edit logical-systems *logical-system-name* interfaces]**

4. Configure BGP between the PE routers:

```
bgp {
  group pe {
    type internal;
    local-address local-address;
    family inet {
      unicast;
    }
    family inet-vpn {
      unicast;
    }
    peer-as as-number;
    neighbor address;
  }
}
```

You can include these statements at the following hierarchy levels:

- **[edit protocols]**
- **[edit logical-systems *logical-system-name* protocols]**

5. Configure the routing instance:

```
instance-type vrf;
interface interface-name;
route-distinguisher address;
vrf-import import-policy-name;
vrf-export export-policy-name;
protocols {
  bgp {
    group routing-instance-name {
      type external;
      peer-as as-number;
      as-override;
      neighbor address;
    }
  }
```

```
}
```

You can include these statements at the following hierarchy levels:

- **[edit routing-instances** *routing-instance-name***]**
- **[edit logical-systems** *logical-system-name* **routing-instances** *routing-instance-name***]**

6. Configure the policy options:

```
policy-statement import-policy-name {
  term 1 {
    from {
      protocol bgp;
      community community-name;
    }
    then accept;
  }
  term 2 {
    then reject;
  }
}
policy-statement export-policy-name {
  term 1 {
    from protocol [ bgp direct ];
    then {
      community add community-name;
      accept;
    }
  }
  term 2 {
    then reject;
  }
}
community community-name members target:target;
```

You can include these statements at the following hierarchy levels:

- **[edit policy-options]**
- **[edit logical-systems** *logical-system-name* **policy-options]**

7. Configure routing table groups to enable VPN route resolution in the **inet.3** routing table:

```
interface-routes {
  rib-group inet if-rib;
}
rib inet.3 {
  static {
    route BGP-address-for-remote-PE next-hop gre-interface-name;
  }
}
rib-groups {
  if-rib {
    import-rib [ inet.0 inet.3 ];
  }
}
```

You can include these statements at the following hierarchy levels:

- **[edit routing-options]**
- **[edit logical-systems *logical-system-name* routing-options]**

Configuring Protocol-Independent Load Balancing in Layer 3 VPNs

Protocol-independent load balancing for Layer 3 VPNs allows the forwarding next hops of both the active route and alternative paths to be used for load balancing. Protocol-independent load balancing works in conjunction with Layer 3 VPNs. It supports the load balancing of VPN routes independently of the assigned route distinguisher. When protocol-independent load balancing is enabled, both routes to other PE routers and routes to directly connected CE routers are load-balanced.

When load-balancing information is created for a given route, the active path is marked as **Routing Use Only** in the output of the **show route table** command.

The following sections describe how to configure protocol-independent load balancing and how this configuration can affect routing policies:

- Configuring Load Balancing for Layer 3 VPNs on page 83
- Configuring Load Balancing and Routing Policies on page 84

Configuring Load Balancing for Layer 3 VPNs

The configuration of protocol independent load balancing for Layer 3 VPNs is a little different for IPv4 versus IPv6:

- IPv4—You only need to configure the **multipath** statement at either the **[edit routing-instances *routing-instance-name* routing-options]** hierarchy level or at the **[edit routing-instances *routing-instance-name* routing-options rib *routing-table-name*]** hierarchy level.
- IPv6—You need to configure the **multipath** statement at both the **[edit routing-instances *routing-instance-name* routing-options]** hierarchy level and the **[edit routing-instances *routing-instance-name* routing-options rib *routing-table-name*]** hierarchy level.

To configure protocol-independent load balancing for Layer 3 VPNs, include the **multipath** statement:

```
multipath {
  vpn-unequal-cost equal-external-internal;
}
```

When you include the **multipath** statement at the following hierarchy levels, protocol-independent load balancing is applied to the default routing table for that routing instance (***routing-instance-name*.inet.0**):

- **[edit routing-instances *routing-instance-name* routing-options]**
- **[edit logical-systems *logical-system-name* routing-instances *routing-instance-name* routing-options]**

When you include the **multipath** statement at the following hierarchy levels, protocol-independent load balancing is applied to the specified routing table:

- **[edit routing-instances** *routing-instance-name* **routing-options rib** *routing-table-name***]**
- **[edit logical-systems** *logical-system-name* **routing-instances** *routing-instance-name* **routing-options rib** *routing-table-name***]**

The **vpn-unequal-cost** statement is optional:

- When you include it, protocol-independent load balancing is applied to VPN routes that are equal until the IGP metric with regard to route selection.
- When you do not include it, protocol-independent load balancing is applied to VPN routes that are equal until the router identifier with regard to route selection.

The **equal-external-internal** statement is also optional. When you include it, protocol-independent load balancing is applied to both internal and external BGP paths. You can configure this in conjunction with egress IP header filtering (enabled with the **vrf-table-label** statement). For more information, see "Load Balancing and IP Header Filtering for Layer 3 VPNs" on page 51.

NOTE: You can include the **vpn-unequal-cost equal-external-internal** statement and the **l3vpn-composite-nexthop** statement simultaneously. However, if you do this, EBGP does not work. This means that when there are both paths with chained nexthops and paths with nonchained nexthops as candidates for EBGP equal-cost multipath (ECMP), the paths using chained nexthops are excluded. In a typical case, the excluded paths are the internal paths.

Configuring Load Balancing and Routing Policies

If you enable protocol-independent load balancing for Layer 3 VPNs by including the **multipath** statement and if you also include the **load-balance per-packet** statement in the routing policy configuration, packets are not load-balanced.

For example, a PE router has the following VRF routing instance configured:

```
[edit routing-instances]
load-balance-example {
   instance-type vrf;
   interface fe-0/1/1.0;
   interface fe-0/1/1.1;
   route-distinguisher 2222:2;
   vrf-target target:2222:2;
   routing-options {
      multipath;
   }
   protocols {
      bgp {
         group group-example {
            import import-policy;
            family inet {
```

```
            unicast;
          }
          export export-policy;
          peer-as 4444;
          local-as 3333;
          multipath;
          as-override;
          neighbor 10.12.33.22;
        }
      }
    }
  }
```

The PE router also has the following policy statement configured:

```
[edit policy-options policy-statement export-policy]
from protocol bgp;
then {
  load-balance per-packet;
}
```

When you include the **multipath** statement in the VRF routing instance configuration, the paths are no longer marked as BGP paths but are instead marked as multipath paths. Packets from the PE router are not load-balanced.

To ensure that VPN load-balancing functions as expected, do not include the **from protocol** statement in the policy statement configuration. The policy statement should be configured as follows:

```
[edit policy-options policy-statement export-policy]
then {
  load-balance per-packet;
}
```

For more information about how to configure per-packet load balancing, see the *Junos OS Policy Framework Configuration Guide*.

Configuring the Algorithm That Determines the Active Route to Evaluate AS Numbers in AS Paths for VPN Routes

By default, the third step of the algorithm that determines the active route evaluates the length of the AS path but not the contents of the AS path. In some VPN scenarios with BGP multiple path routes, it can also be useful to compare the AS numbers of the AS paths and to have the algorithm select the route whose AS numbers match.

To configure the algorithm that selects the active path to evaluate the AS numbers in AS paths for VPN routes:

- Include the **as-path-compare** statement at the **[edit routing-instances *routing-instance-name* routing-options multipath]** hierarchy level.

NOTE: The as-path-compare statement is not supported for the default routing instance.

Related Documentation
- as-path-compare on page 345

Configuring Traffic Policing in Layer 3 VPNs

You can use policing to control the amount of traffic flowing over the interfaces servicing a Layer 3 VPN. If policing is disabled on an interface, all the available bandwidth on a Layer 3 VPN tunnel can be used by a single CCC or TCC interface.

For more information about the **policer** statement, see the *Junos OS Policy Framework Configuration Guide*.

To enable Layer 3 VPN policing on an interface, include the **policer** statement:

```
policer {
    input policer-template-name;
    output policer-template-name;
}
```

If you configure CCC encapsulation, you can include the **policer** statement at the following hierarchy levels:

- **[edit interfaces** *interface-name* **unit** *logical-unit-number* **family ccc]**
- **[edit logical-systems** *logical-system-name* **interfaces** *interface-name* **unit** *logical-unit-number* **family ccc]**

If you configure TCC encapsulation, you can include the **policer** statement at the following hierarchy levels:

- **[edit interfaces** *interface-name* **unit** *logical-unit-number* **family tcc]**
- **[edit logical-systems** *logical-system-name* **interfaces** *interface-name* **unit** *logical-unit-number* **family tcc]**

Accepting BGP Updates with Unique Inner VPN Labels in Layer 3 VPNs

For Layer 3 VPNs configured on Juniper Networks routers, Junos OS normally allocates one inner VPN label for each VPN network on the customer edge (CE)-facing interfaces of a provider edge (PE) router. However, other vendors allocate one VPN label for each BGP route on the CE-facing interfaces of a PE router. This practice increases the number of VPN labels exponentially, which leads to slow system processing and slow convergence time. To account for this difference, configure the **l3vpn-composite-nexthop** statement at the **[edit routing-options]** hierarchy level on the Juniper Networks routers participating in a mixed vendor network. The **l3vpn-composite-nexthop** statement is disabled by default.

Next-hop chaining (also known as composite next hop) is a composition function that concatenates the partial rewrite strings associated with individual next hops to form a larger rewrite string that is added to a packet.

When you configure the **l3vpn-composite-nexthop** statement, the number of BGP routes with unique inner VPN labels that can be processed by a Juniper Networks router is increased substantially. This statement functions by combining the common elements of the BGP route updates associated with Layer 3 VPNs (hence the statement name). This functionality reduces the number of route updates and reduces the amount of state the router needs to maintain, leading to enhanced scaling and convergence performance on Juniper Networks routers.

We recommend that you configure the **l3vpn-composite-nexthop** statement whenever you have deployed Juniper Networks routers in mixed vendor networks to support Layer 3 VPNs. This feature can also enhance the Layer 3 VPN performance of Juniper Networks routers in networks where only Juniper Networks routers are deployed. We recommend configuring the **l3vpn-composite-nexthop** statement in these networks as well.

You can configure the **l3vpn-composite-nexthop** statement on the following routers:

- MX Series routers
- M120 routers
- M320 routers with an Enhanced III FPC
- T Series routers (for Junos OS Release 10.4 and later)

To accept larger numbers of Layer 3 VPN BGP updates with unique inner VPN labels, configure the **l3vpn-composite-nexthop** statement. This statement is supported on indirectly connected PE routers only. Although you can configure this statement on a router that is directly connected to a PE router, there is no benefit to doing so. There is no problem with configuring the **l3vpn-composite-nexthop** statement on a router with a mix of links to both directly connected and indirectly connected PE routers.

You can include the l3vpn-composite-nexthop statement at the following hierarchy levels:

- **[edit routing-options]**
- **[edit logical-systems *logical-system-name* routing-options]**

When you have configured the **l3vpn-composite-nexthop** statement, you can determine whether or not a Layer 3 VPN route is a part of a composite next hop by examining the display output of the following commands:

- **show route *route-value* extensive**
- **show route forwarding-table destination *destination-value* extensive**

To enhance memory allocation to support a larger number of Layer 3 VPN labels, include the **vpn-label** statement at the **[edit chassis memory-enhanced]** hierarchy level. For more information about configuring more memory for Layer VPN labels, see the *Junos OS System Basics Configuration Guide.*

Related Documentation

- Chained Composite Next Hops for Transit Devices

CHAPTER 4

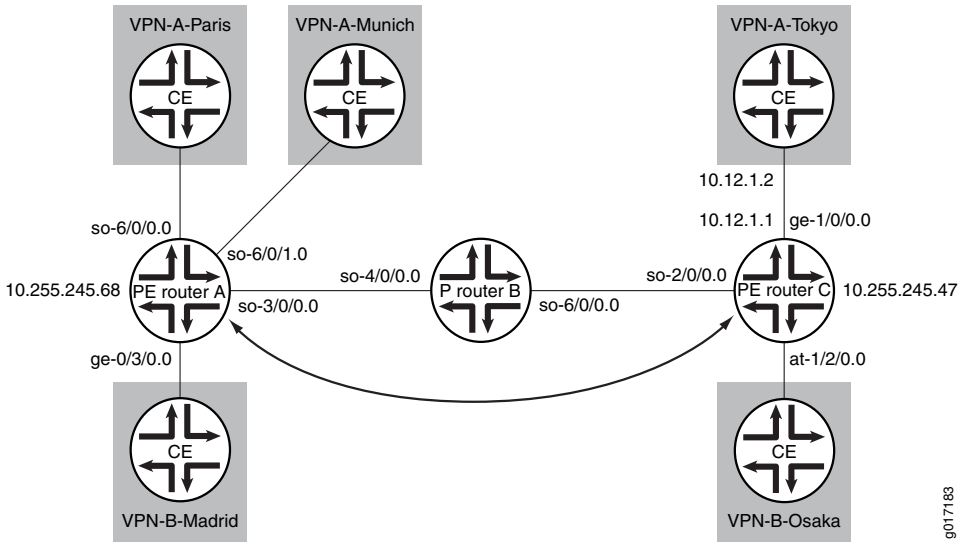
Layer 3 VPN Configuration Examples

Configuring a Simple Full-Mesh VPN Topology

This example shows how to set up a simple full-mesh service provider VPN configuration, which consists of the following components (see Figure 16 on page 90):

- Two separate VPNs (VPN-A and VPN-B)
- Two provider edge (PE) routers, both of which service VPN-A and VPN-B
- RSVP as the signaling protocol
- One RSVP label-switched path (LSP) that tunnels between the two PE routers through one provider (P) router

Figure 16: Example of a Simple VPN Topology



In this configuration, route distribution in VPN A from Router VPN-A-Paris to Router VPN-A-Tokyo occurs as follows:

1. The customer edge (CE) router VPN-A-Paris announces routes to the PE router Router A.
2. Router A installs the received announced routes into its VPN routing and forwarding (VRF) table, **VPN-A.inet.0**.
3. Router A creates an MPLS label for the interface between it and Router VPN-A-Paris.
4. Router A checks its VRF export policy.
5. Router A converts the Internet Protocol version 4 (IPv4) routes from Router VPN-A-Paris into VPN IPv4 format using its route distinguisher and announces these routes to PE Router C over the IBGP between the two PE routers.
6. Router C checks its VRF import policy and installs all routes that match the policy into its **bgp.l3vpn.0** routing table. (Any routes that do not match are discarded.)
7. Router C checks its VRF import policy and installs all routes that match into its **VPN-A.inet.0** routing table. The routes are installed in IPv4 format.
8. Router C announces its routes to the CE router Router VPN-A-Tokyo, which installs them into its master routing table. (For routing platforms running Junos OS, the master routing table is **inet.0**.)
9. Router C uses the LSP between it and Router A to route all packets from Router VPN-A-Tokyo that are destined for Router VPN-A-Paris.

The final section in this example consolidates the statements needed to configure VPN functionality on each of the service P routers shown in Figure 16 on page 90.

NOTE: In this example, a private autonomous system (AS) number is used for the route distinguisher and the route target. This number is used for illustration only. When you are configuring VPNs, you should use an assigned AS number.

The following sections explain how to configure the VPN functionality on the PE and P routers. The CE routers have no information about the VPN, so you configure them normally.

Enabling an IGP on the PE and P Routers

To allow the PE and P routers to exchange routing information among themselves, you must configure an interior gateway protocol (IGP) on all these routers or you must configure static routes. You configure the IGP on the master instance of the routing protocol process (**rpd**) (that is, at the **[edit protocols]** hierarchy level), not within the VPN routing instance (that is, not at the **[edit routing-instances]** hierarchy level).

You configure the IGP in the standard way. This configuration example does not include this portion of the configuration.

Enabling RSVP and MPLS on the P Router

On the P router, Router B, you must configure RSVP and MPLS because this router exists on the MPLS LSP path between the two PE routers, Router A and Router C:

```
[edit]
protocols {
   rsvp {
      interface so-4/0/0.0;
      interface so-6/0/0.0;
   }
   mpls {
      interface so-4/0/0.0;
      interface so-6/0/0.0;
   }
}
```

Configuring the MPLS LSP Tunnel Between the PE Routers

In this configuration example, RSVP is used for VPN signaling. Therefore, in addition to configuring RSVP, you must enable traffic engineering support in an IGP and you must create an MPLS LSP to tunnel the VPN traffic.

On PE Router A, enable RSVP and configure one end of the MPLS LSP tunnel. In this example, traffic engineering support is enabled for OSPF. When configuring the MPLS LSP, include **interface** statements for all interfaces participating in MPLS, including the interfaces to the PE and CE routers. The statements for the interfaces between the PE and CE routers are needed so that the PE router can create an MPLS label for the private interface. In this example, the first **interface** statement configures MPLS on the interface connected to the LSP, and the remaining three configure MPLS on the interfaces that connect the PE router to the CE routers.

```
[edit]
protocols {
    rsvp {
        interface so-3/0/0.0;
    }
    mpls {
        label-switched-path RouterA-to-RouterC {
            to 10.255.245.47;
        }
        interface so-3/0/0.0;
        interface so-6/0/0.0;
        interface so-6/0/1.0;
        interface ge-0/3/0.0;
    }
    ospf {
        traffic-engineering;
        area 0.0.0.0 {
            interface so-3/0/0.0;
        }
    }
}
```

On PE Router C, enable RSVP and configure the other end of the MPLS LSP tunnel. Again, traffic engineering support is enabled for OSPF, and you configure MPLS on the interfaces to the LSP and the CE routers.

```
[edit]
protocols {
    rsvp {
        interface so-2/0/0.0;
    }
    mpls {
        label-switched-path RouterC-to-RouterA {
            to 10.255.245.68;
        }
        interface so-2/0/0.0;
        interface ge-1/0/0.0;
        interface at-1/2/0.0;
    }
```

```
    ospf {
      traffic-engineering;
      area 0.0.0.0 {
        interface so-2/0/0.0;
      }
    }
}
```

Configuring IBGP on the PE Routers

On the PE routers, configure an IBGP session with the following properties:

- VPN family—To indicate that the IBGP session is for the VPN, include the **family inet-vpn** statement.
- Loopback address—Include the **local-address** statement, specifying the local PE router's loopback address. The IBGP session for VPNs runs through the loopback address. You must also configure the **lo0** interface at the **[edit interfaces]** hierarchy level. The example does not include this part of the router's configuration.
- Neighbor address—Include the **neighbor** statement, specifying the IP address of the neighboring PE router, which is its loopback (**lo0**) address.

On PE Router A, configure IBGP:

```
[edit]
protocols {
  bgp {
    group PE-RouterA-to-PE-RouterC {
      type internal;
      local-address 10.255.245.68;
      family inet-vpn {
        unicast;
      }
      neighbor 10.255.245.47;
    }
  }
}
```

On PE Router C, configure IBGP:

```
[edit]
protocols {
  bgp {
    group PE-RouterC-to-PE-RouterA {
      type internal;
      local-address 10.255.245.47;
      family inet-vpn {
        unicast;
      }
      neighbor 10.255.245.68;
    }
  }
}
```

Configuring Routing Instances for VPNs on the PE Routers

Both PE routers service VPN-A and VPN-B, so you must configure two routing instances on each router, one for each VPN. For each VPN, you must define the following in the routing instance:

- Route distinguisher, which must be unique for each routing instance on the PE router.
- It is used to distinguish the addresses in one VPN from those in another VPN.
- Instance type of **vrf**, which creates the VRF table on the PE router.
- Interfaces connected to the CE routers.
- VRF import and export policies, which must be the same on each PE router that services the same VPN. Unless an import policy contains only a **then reject** statement, it must include reference to a community. Otherwise, when you try to commit the configuration, the commit fails.

NOTE: In this example, a private AS number is used for the route distinguisher. This number is used for illustration only. When you are configuring VPNs, you should use an assigned AS number.

- Routing between the PE and CE routers, which is required for the PE router to distribute VPN-related routes to and from connected CE routers. You can configure a routing protocol—BGP, OSPF, or RIP—or you can configure static routing.

On PE Router A, configure the following routing instance for VPN-A. In this example, Router A uses static routes to distribute routes to and from the two CE routers to which it is connected.

```
[edit]
routing-instance {
   VPN-A-Paris-Munich {
      instance-type vrf;
      interface so-6/0/0.0;
      interface so-6/0/1.0;
      route-distinguisher 65535:0;
      vrf-import VPN-A-import;
      vrf-export VPN-A-export;
      routing-options {
         static {
            route 172.16.0.0/16 next-hop so-6/0/0.0;
            route 172.17.0.0/16 next-hop so-6/0/1.0;
         }
      }
   }
}
```

On PE Router C, configure the following routing instance for VPN-A. In this example, Router C uses BGP to distribute routes to and from the CE router to which it is connected.

```
[edit]
```

```
routing-instance {
    VPN-A-Tokyo {
        instance-type vrf;
        interface ge-1/0/0.0;
        route-distinguisher 65535:1;
        vrf-import VPN-A-import;
        vrf-export VPN-A-export;
        protocols {
            bgp {
                group VPN-A-Site2 {
                    peer-as 1;
                    neighbor 10.12.1.2;
                }
            }
        }
    }
}
```

On PE Router A, configure the following routing instance for VPN-B. In this example, Router A uses OSPF to distribute routes to and from the CE router to which it is connected.

```
[edit]
routing-instance {
    VPN-B-Madrid {
        instance-type vrf;
        interface ge-0/3/0.0;
        route-distinguisher 65535:2;
        vrf-import VPN-B-import;
        vrf-export VPN-B-export;
        protocols {
            ospf {
                export bgp-to-ospf;
                area 0.0.0.0 {
                    interface ge-0/3/0;
                }
            }
        }
    }
}
```

On PE Router C, configure the following routing instance for VPN-B. In this example, Router C uses RIP to distribute routes to and from the CE router to which it is connected.

```
[edit]
routing-instance {
    VPN-B-Osaka {
        instance-type vrf;
        interface at-1/2/0.0;
        route-distinguisher 65535:3;
        vrf-import VPN-B-import;
        vrf-export VPN-B-export;
        protocols {
            rip {
                group PE-C-to-VPN-B {
                    export bgp-to-rip;
                    neighbor at-1/2/0;
```

```
        }
      }
    }
  }
}
```

Configuring VPN Policy on the PE Routers

Configure the VPN import and export policies on each PE router so that the appropriate routes are installed in the PE router's VRF tables. The VRF table is used to forward packets within a VPN. For VPN-A, the VRF table is **VPN-A.inet.0**, and for VPN-B it is **VPN-B.inet.0**.

In the VPN policy, you also configure VPN target communities.

In the following example, a private AS number is used for the route target. This number is used for illustration only. When you are configuring VPNs, you should use an assigned AS number. The policy qualifiers shown in this example are only those needed for the VPN to function. You can configure additional qualifiers, as needed, for any policies that you configure.

On PE Router A, configure the following VPN import and export policies:

```
[edit]
policy-options {
  policy-statement VPN-A-import {
    term a {
      from {
        protocol bgp;
        community VPN-A;
      }
      then accept;
    }
    term b {
      then reject;
    }
  }
  policy-statement VPN-A-export {
    term a {
      from protocol static;
      then {
        community add VPN-A;
        accept;
      }
    }
    term b {
      then reject;
    }
  }
  policy-statement VPN-B-import {
    term a {
      from {
        protocol bgp;
        community VPN-B;
      }
      then accept;
```

```
        }
        term b {
            then reject;
        }
    }
    policy-statement VPN-B-export {
        term a {
            from protocol ospf;
            then {
                community add VPN-B;
                accept;
            }
        }
        term b {
            then reject;
        }
    }
    community VPN-A members target:65535:4;
    community VPN-B members target:65535:5;
}
```

On PE Router C, configure the following VPN import and export policies:

```
[edit]
policy-options {
    policy-statement VPN-A-import {
        term a {
            from {
                protocol bgp;
                community VPN-A;
            }
            then accept;
        }
        term b {
            then reject;
        }
    }
    policy-statement VPN-A-export {
        term a {
            from protocol bgp;
            then {
                community add VPN-A;
                accept;
            }
        }
        term b {
            then reject;
        }
    }
    policy-statement VPN-B-import {
        term a {
            from {
                protocol bgp;
                community VPN-B;
            }
            then accept;
```

```
        }
        term b {
            then reject;
        }
    }
    policy-statement VPN-B-export {
        term a {
            from protocol rip;
            then {
                community add VPN-B;
                accept;
            }
        }
        term b {
            then reject;
        }
    }
    community VPN-A members target:65535:4;
    community VPN-B members target:65535:5;
}
```

To apply the VPN policies on the routers, include the **vrf-export** and **vrf-import** statements when you configure the routing instance. For both VPNs, the VRF import and export policies handle the route distribution across the IBGP session running between the PE routers.

To apply the VPN policies on PE Router A, include the following statements:

```
[edit]
routing-instance {
    VPN-A-Paris-Munich {
        vrf-import VPN-A-import;
        vrf-export VPN-A-export;
    }
    VPN-B-Madrid {
        vrf-import VPN-B-import;
        vrf-export VPN-B-export;
    }
}
```

To apply the VPN policies on PE Router C, include the following statements:

```
[edit]
routing-instance {
    VPN-A-Tokyo {
        vrf-import VPN-A-import;
        vrf-export VPN-A-export;
    }
    VPN-B-Osaka {
        vrf-import VPN-B-import;
        vrf-export VPN-B-export;
    }
}
```

Simple VPN Configuration Summarized by Router

Router A (PE Router)

Routing Instance for VPN-A

```
routing-instance {
   VPN-A-Paris-Munich {
      instance-type vrf;
      interface so-6/0/0.0;
      interface so-6/0/1.0;
      route-distinguisher 65535:0;
      vrf-import VPN-A-import;
      vrf-export VPN-A-export;
   }
}
```

Instance Routing Protocol

```
routing-options {
   static {
      route 172.16.0.0/16 next-hop so-6/0/0.0;
      route 172.17.0.0/16 next-hop so-6/0/1.0;
   }
}
```

Routing Instance for VPN-B

```
routing-instance {
   VPN-B-Madrid {
      instance-type vrf;
      interface ge-0/3/0.0;
      route-distinguisher 65535:2;
      vrf-import VPN-B-import;
      vrf-export VPN-B-export;
   }
}
```

Instance Routing Protocol

```
protocols {
   ospf {
      area 0.0.0.0 {
         interface ge-0/3/0;
      }
   }
}
```

Master Protocol Instance

```
protocols {
}
```

Enable RSVP

```
rsvp {
   interface so-3/0/0.0;
}
```

Configure an MPLS LSP

```
mpls {
   label-switched-path RouterA-to-RouterC {
      to 10.255.245.47;
   }
   interface so-3/0/0.0;
   interface so-6/0/0.0;
   interface so-6/0/1.0;
```

```
    interface ge-0/3/0.0;
}
```

Configure IBGP

```
bgp {
  group PE-RouterA-to-PE-RouterC {
    type internal;
    local-address 10.255.245.68;
    family inet-vpn {
      unicast;
    }
    neighbor 10.255.245.47;
  }
}
```

Configure OSPF for Traffic Engineering Support

```
ospf {
  traffic-engineering;
  area 0.0.0.0 {
    interface so-3/0/0.0;
  }
}
```

Configure VPN Policy

```
policy-options {
  policy-statement VPN-A-import {
    term a {
      from {
        protocol bgp;
        community VPN-A;
      }
      then accept;
    }
    term b {
      then reject;
    }
  }
  policy-statement VPN-A-export {
    term a {
      from protocol static;
      then {
        community add VPN-A;
        accept;
      }
    }
    term b {
      then reject;
    }
  }
  policy-statement VPN-B-import {
    term a {
      from {
        protocol bgp;
        community VPN-B;
      }
      then accept;
    }
    term b {
```

```
        then reject;
      }
    }
    policy-statement VPN-B-export {
      term a {
        from protocol ospf;
        then {
          community add VPN-B;
          accept;
        }
      }
      term b {
        then reject;
      }
    }
    community VPN-A members target:65535:4;
    community VPN-B members target:65535:5;
}
```

Router B (P Router)

Master Protocol Instance

```
protocols {
}
```

Enable RSVP

```
rsvp {
  interface so-4/0/0.0;
  interface so-6/0/0.0;
}
```

Enable MPLS

```
mpls {
  interface so-4/0/0.0;
  interface so-6/0/0.0;
}
```

Router C (PE Router)

Routing Instance for VPN-A

```
routing-instance {
  VPN-A-Tokyo {
    instance-type vrf;
    interface ge-1/0/0.0;
    route-distinguisher 65535:1;
    vrf-import VPN-A-import;
    vrf-export VPN-A-export;
  }
}
```

Instance Routing Protocol

```
protocols {
  bgp {
    group VPN-A-Site2 {
      peer-as 1;
      neighbor 10.12.1.2;
    }
  }
}
```

Routing Instance for VPN-B

```
VPN-B-Osaka {
  instance-type vrf;
  interface at-1/2/0.0;
  route-distinguisher 65535:3;
  vrf-import VPN-B-import;
  vrf-export VPN-B-export;
}
```

Instance Routing Protocol

```
protocols {
  rip {
    group PE-C-to-VPN-B {
      neighbor at-1/2/0;
    }
  }
}
```

Master Protocol Instance

```
protocols {
}
```

Enable RSVP

```
rsvp {
  interface so-2/0/0.0;
}
```

Configure an MPLS LSP

```
mpls {
  label-switched-path RouterC-to-RouterA {
    to 10.255.245.68;
  }
  interface so-2/0/0.0;
  interface ge-1/0/0.0;
  interface at-1/2/0.0;
}
```

Configure IBGP

```
bgp {
  group PE-RouterC-to-PE-RouterA {
    type internal;
    local-address 10.255.245.47;
    family inet-vpn {
      unicast;
    }
    neighbor 10.255.245.68;
  }
}
```

Configure OSPF for Traffic Engineering Support

```
ospf {
  traffic-engineering;
  area 0.0.0.0 {
    interface so-2/0/0.0;
  }
}
```

Configure VPN Policy

```
policy-options {
  policy-statement VPN-A-import {
    term a {
      from {
```

```
            protocol bgp;
            community VPN-A;
          }
          then accept;
        }
        term b {
          then reject;
        }
      }
      policy-statement VPN-A-export {
        term a {
          from protocol bgp;
          then {
            community add VPN-A;
            accept;
          }
        }
        term b {
          then reject;
        }
      }
      policy-statement VPN-B-import {
        term a {
          from {
            protocol bgp;
            community VPN-B;
          }
          then accept;
        }
        term b {
          then reject;
        }
      }
      policy-statement VPN-B-export {
        term a {
          from protocol rip;
          then {
            community add VPN-B;
            accept;
          }
        }
        term b {
          then reject;
        }
      }
      community VPN-A members target:65535:4;
      community VPN-B members target:65535:5;
    }
```

Configuring a Full-Mesh VPN Topology with Route Reflectors

This example is a variation of the full-mesh VPN topology example (described in "Configuring a Simple Full-Mesh VPN Topology" on page 89) in which one of the PE routers is a BGP route reflector. In this variation, Router C in Figure 16 on page 90 is a route

reflector. The only change to its configuration is that you need to include the **cluster** statement when configuring the BGP group:

```
[edit]
protocols {
   bgp {
      group PE-RouterC-to-PE-RouterA {
         type internal;
         local-address 10.255.245.47;
         family inet-vpn {
            unicast;
         }
         neighbor 10.255.245.68;
         cluster 4.3.2.1;
      }
   }
}
```

For the complete configuration example of Router C, see "Configuring a Full-Mesh VPN Topology with Route Reflectors" on page 103.

Configuring Hub-and-Spoke VPN Topologies: One Interface

Use a one-interface configuration to advertise a default route from a hub or hubs.

Figure 17: Example of a Hub-and-Spoke VPN Topology with One Interface

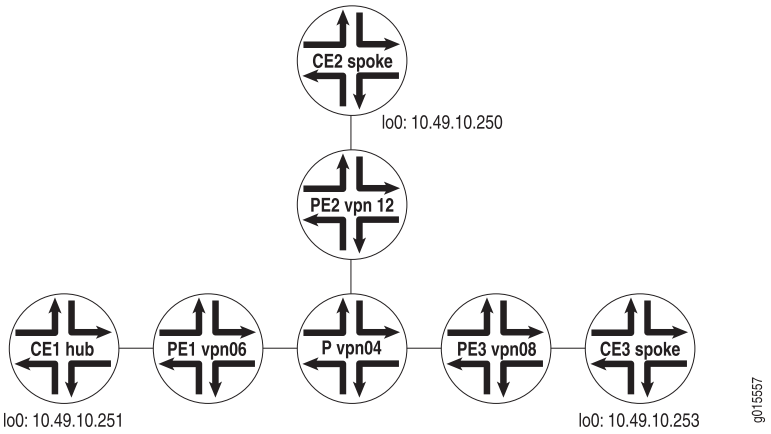


Figure 17 on page 104 illustrates a Layer 3 VPN hub-and-spoke application where there is only one interface between the hub CE (CE1) and the hub PE (PE1). This is the recommended way of configuring hub-and-spoke topologies.

In this configuration, a default route is advertised from the hub to the spokes. If more specific spoke CE routes need to be exchanged between spoke CE routers, then two interfaces are needed between the hub CE and hub PE. See "Configuring Hub-and-Spoke VPN Topologies: Two Interfaces" on page 116 for a two-interface example.

In this configuration example, spoke route distribution is as follows:

1. Spoke CE2 advertises its routes to spoke PE2.
2. Spoke PE2 installs routes from CE2 into its VPN routing and forwarding (VRF) table.
3. Spoke PE2 checks its VRF export policy, adds the route target community, and announces the routes to hub PE1.
4. Hub PE1 checks its VRF import policy and installs routes that match the import policy into table **bgp.l3vpn.0**.
5. Hub PE1 installs routes from table **bgp.l3vpn.0** into the hub VRF table.
6. Hub PE1 announces routes from the hub VRF table to the hub CE1.

In this configuration example, default route distribution is as follows:

1. Hub CE1 announces a default route to hub PE1.
2. Hub PE1 installs the default route into the hub VRF table.
3. Hub PE1 checks its VRF export policy, adds the route target community and announces the default route to spoke PE2 and PE3.
4. Spoke PE2 and PE3 check their VRF import policy and install the default route into table **bgp.l3vpn.0**.
5. Spoke PE2 and PE3 install the routes from table **bgp.l3vpn.0** into their spoke VRF tables.
6. Spoke PE2 and PE3 announce the default route from the spoke VRF table to spoke CE2 and CE3.

The following sections describe how to configure a hub-and-spoke topology with one interface based on the topology illustrated in Figure 17 on page 104:

- Configuring Hub CE1 on page 105
- Configuring Hub PE1 on page 106
- Configuring the P Router on page 107
- Configuring Spoke PE2 on page 107
- Configuring Spoke PE3 on page 109
- Configuring Spoke CE2 on page 110
- Configuring Spoke CE3 on page 110
- Enabling Egress Features on the Hub PE Router on page 112

Configuring Hub CE1

Configure hub CE1 as follows:

```
[edit routing-options]
static {
   route 0.0.0.0/0 discard;
}
```

```
autonomous-system 100;
[edit protocols]
bgp {
  group hub {
    type external;
    export default;
    peer-as 200;
    neighbor 10.49.4.1;
  }
}
[edit policy-statement]
default {
  term 1 {
    from {
      protocol static;
      route-filter 0.0.0.0/0 exact;
    }
    then accept;
  }
  term 2 {
    then reject;
  }
}
```

Configuring Hub PE1

Configure hub PE1 as follows:

```
[edit]
routing-instances {
  hub {
    instance-type vrf;
    interface t3-0/0/0 {
      encapsulation frame-relay;
      unit 0 {
        dlci 16;
        family inet {
          address 10.49.4.1/30;
        }
      }
    }
    vrf-target {
      import target:200:100;
      export target:200:101;
    }
    protocols {
      bgp {
        group hub {
          type external;
          peer-as 100;
          as-override;
          neighbor 10.49.4.2;
        }
      }
    }
  }
```

```
}
```

Configuring the P Router

Configure the P Router as follows:

```
[edit]
interfaces {
   t3-0/1/1 {
      unit 0 {
         family inet {
            address 10.49.2.1/30;
         }
         family mpls;
      }
   }
   t3-0/1/3 {
      unit 0 {
         family inet {
            address 10.49.0.2/30;
         }
         family mpls;
      }
   }
   t1-0/2/0 {
      unit 0 {
         family inet {
            address 10.49.1.2/30;
         }
         family mpls;
      }
   }
}
[edit]
protocols {
   ospf {
      area 0.0.0.0 {
         interface t3-0/1/3.0;
         interface t1-0/2/0.0;
         interface t3-0/1/1.0;
         interface lo0.0 {
            passive;
         }
      }
   }
   ldp {
      interface t3-0/1/1.0;
      interface t3-0/1/3.0;
      interface t1-0/2/0.0;
   }
}
```

Configuring Spoke PE2

Configure spoke PE2 as follows:

```
[edit]
interfaces {
   t3-0/0/0 {
      unit 0 {
         family inet {
            address 10.49.0.1/30;
         }
         family mpls;
      }
   }
   t1-0/1/2 {
      unit 0 {
         family inet {
            address 10.49.3.1/30;
         }
      }
   }
}
[edit protocols]
bgp {
   group ibgp {
      type internal;
      local-address 10.255.14.182;
      peer-as 200;
      neighbor 10.255.14.176 {
         family inet-vpn {
            unicast;
         }
      }
   }
}
ospf {
   area 0.0.0.0 {
      interface t3-0/0/0.0;
      interface lo0.0 {
         passive;
      }
   }
}
ldp {
   interface t3-0/0/0.0;
}
[edit]
routing-instances {
   spoke {
      instance-type vrf;
      interface t1-0/1/2.0;
      vrf-target {
         import target:200:101;
         export target:200:100;
      }
      protocols {
         bgp {
            group spoke {
               type external;
               peer-as 100;
```

```
            as-override;
            neighbor 10.49.3.2;
          }
        }
      }
    }
}
```

Configuring Spoke PE3

Configure spoke PE3 as follows:

```
[edit]
interfaces {
  t3-0/0/0 {
    unit 0 {
      family inet {
        address 10.49.6.1/30;
      }
    }
  }
  t3-0/0/1 {
    unit 0 {
      family inet {
        address 10.49.2.2/30;
      }
      family mpls;
    }
  }
}
[edit protocols}
bgp {
  group ibgp {
    type internal;
    local-address 10.255.14.178;
    peer-as 200;
    neighbor 10.255.14.176 {
      family inet-vpn {
        unicast;
      }
    }
  }
}
ospf {
  area 0.0.0.0 {
    interface t3-0/0/1.0;
    interface lo0.0 {
      passive;
    }
  }
}
ldp {
  interface t3-0/0/1.0;
}
[edit]
routing-instances {
```

```
    spoke {
      instance-type vrf;
      interface t3-0/0/0.0;
      vrf-target {
        import target:200:101;
        export target:200:100;
      }
      protocols {
        bgp {
          group spoke {
            type external;
            peer-as 100;
            as-override;
            neighbor 10.49.6.2;
          }
        }
      }
    }
}
```

Configuring Spoke CE2

Configure spoke CE2 as follows:

```
[edit routing-options]
autonomous-system 100;
{edit protocols]
bgp {
  group spoke {
    type external;
    export loopback;
    peer-as 200;
    neighbor 10.49.3.1;
  }
}
```

Configuring Spoke CE3

Configure spoke CE3 as follows:

```
[edit routing-options]
autonomous-system 100;
[edit protocols]
bgp {
  group spoke {
    type external;
    export loopback;
    peer-as 200;
    neighbor 10.49.6.1;
  }
}
```

In this configuration example, traffic forwarding is as follows between spoke CE2 and hub CE1:

1. Spoke CE2 forwards traffic using the default route learned from spoke PE2 through BGP.

```
        0.0.0.0/0          *[BGP/170] 02:24:15, localpref 100
                             AS path: 200 200 I
                            > to 10.49.3.1 via t1-3/0/1.0
```

2. Spoke PE2 performs a route lookup in the spoke VRF table and forwards the traffic to hub PE2 (through the P router—PE2 pushes two labels) using the default route learned through BGP.

```
        0.0.0.0/0          *[BGP/170] 01:35:45, localpref 100, from
    10.255.14.176
                             AS path: 100 I
                            > via t3-0/0/1.0, Push 100336, Push 100224(top)
```

3. Hub PE1 does a route lookup in the **mpls.0 table** for the VPN label **100336**.

```
        100336             *[VPN/170] 01:37:03
                            > to 10.49.4.2 via t3-0/0/0.0, Pop
```

4. Hub PE1 forwards the traffic out the interface **t3-0/0/0.0** to hub CE1.

In this configuration example, traffic forwarding is as follows between hub CE1 and spoke CE2:

1. Hub CE1 forwards traffic to the hub PE1 using the route learned through BGP.

```
        10.49.10.250/32    *[BGP/170] 02:28:46, localpref 100
                             AS path: 200 200 I
                            > to 10.49.4.1 via t3-3/1/0.0
```

2. Hub PE1 does a route lookup in the hub VRF table and forwards the traffic to spoke PE2 (through the P router—PE1 pushes two labels).

```
        10.49.10.250/32    *[BGP/170] 01:41:05, localpref 100, from
    10.255.14.182
                             AS path: 100 I
                            > via t1-0/1/0.0, Push 100352, Push 100208(top)
```

3. Spoke PE2 does a route lookup in the **mpls.0 table** for the VPN label **100352**.

```
        100352             *[VPN/170] 02:31:39
                            > to 10.49.3.2 via t1-0/1/2.0, Pop
```

4. Spoke PE2 forwards the traffic out the interface **t1-0/1/2.0** to spoke CE2.

In this configuration example, traffic forwarding is as follows between spoke CE2 and spoke CE3:

1. Spoke CE2 forwards traffic using the default route learned from spoke PE2 through BGP.

```
0.0.0.0/0          *[BGP/170] 02:24:15, localpref 100
                      AS path: 200 200 I
                    > to 10.49.3.1 via t1-3/0/1.0
```

2. Spoke PE2 does a route lookup in the spoke VRF table and forwards the traffic to hub PE1 (through the P router—PE2 pushes two labels) using the default route learned through BGP.

```
    0.0.0.0/0          *[BGP/170] 01:35:45, localpref 100, from
10.255.14.176
                         AS path: 100 I
                       > via t3-0/0/1.0, Push 100336, Push 100224(top)
```

3. Hub PE1 does a route lookup in the **mpls.0 table** for the VPN label **100336**.

```
100336             *[VPN/170] 01:37:03
                    > to 10.49.4.2 via t3-0/0/0.0, Pop
```

4. Hub PE1 forwards the traffic out the interface **t3-0/0/0.0** to the hub CE1.

5. Hub CE1 forwards the traffic to hub PE1 using the router learned through BGP.

```
10.49.10.253/32    *[BGP/170] 02:40:03, localpref 100
                      AS path: 200 200 I
                    > to 10.49.4.1 via t3-3/1/0.0
```

6. Hub PE1 does a route lookup in the hub VRF table and forwards the traffic to spoke PE3 (through the P router—PE1 pushes two labels).

```
    10.49.10.253/32    *[BGP/170] 01:41:05, localpref 100, from
10.255.14.178
                         AS path: 100 I
                       > via t1-0/1/0.0, Push 100128, Push 100192(top)
```

7. Spoke PE3 does a route lookup in the **mpls.0 table** for VPN label **100128**.

```
100128             *[VPN/170] 02:41:30
                    > to 10.49.6.2 via t3-0/0/0.0, Pop
```

8. Spoke PE3 forwards the traffic out the interface **t3-0/0/0.0** to spoke CE3.

If egress features are needed on the hub PE that require an IP forwarding lookup on the hub VRF routing table, see "Enabling Egress Features on the Hub PE Router" on page 112.

Enabling Egress Features on the Hub PE Router

This example is provided in conjunction with "Configuring Hub-and-Spoke VPN Topologies: One Interface" on page 104. This example also uses the topology illustrated in Figure 17 on page 104.

If egress features are needed on the hub PE that require an IP forwarding lookup on the hub VRF routing table, the configuration detailed in "Configuring Hub-and-Spoke VPN Topologies: One Interface" on page 104 will not work. Applying the **vrf-table-label** statement on the hub routing instance forces traffic from a remote spoke PE to be forwarded to the hub PE and forces an IP lookup to be performed. Because specific spoke routes are in the hub VRF table, traffic will be forwarded to a spoke PE without going through the hub CE.

The hub PE advertises the default route as follows, using VPN label 1028:

```
hub.inet.0: 7 destinations, 7 routes (7 active, 0 holddown, 0 hidden)
* 0.0.0.0/0 (1 entry, 1 announced)
 BGP group ibgp type Internal
     Route Distinguisher: 10.255.14.176:2
     VPN Label: 1028
     Nexthop: Self
     Localpref: 100
     AS path: 100 I
     Communities: target:200:101
```

Incoming traffic is forwarded using VPN label 1028. The **mpls.0 table** shows that an IP lookup in the table **hub.inet.0** is required:

```
1028               *[VPN/0] 00:00:27
                      to table hub.inet.0, Pop
```

However, the hub VRF table **hub.inet.0** contains specific spoke routes:

```
10.49.10.250/32    *[BGP/170] 00:00:05, localpref 100, from 10.255.14.182
                      AS path: 100 I
                    > via t1-0/1/0.0, Push 100352, Push 100208(top)
10.49.10.253/32    *[BGP/170] 00:00:05, localpref 100, from 10.255.14.178
                      AS path: 100 I
                    > via t1-0/1/0.0, Push 100128, Push 100192(top)
```

Because of this, traffic is forwarded directly to the spoke PEs without going through the hub CE. To prevent this, you must configure a secondary routing instance for downstream traffic in the hub PE1.

Configuring Hub PE1

Configure hub PE1 as follows:

```
[edit]
routing-instances {
  hub {
    instance-type vrf;
    interface t3-0/0/0.0;
    vrf-target {
      import target:200:100;
      export target:200:101;
    }
    no-vrf-advertise;
    routing-options {
      auto-export;
    }
    protocols {
      bgp {
        group hub {
          type external;
          peer-as 100;
          as-override;
          neighbor 10.49.4.2;
        }
      }
    }
  }
```

```
    hub-downstream {
      instance-type vrf;
      vrf-target target:200:101;
      vrf-table-label;
      routing-options {
        auto-export;
      }
    }
  }
```

When the **no-vrf-advertise** statement is used at the **[edit routing-instances hub]** hierarchy level, no routing table groups or VRF export policies are required. The **no-vrf-advertise** statement configures the hub PE not to advertise VPN routes from the primary routing-instance **hub**. These routes are instead advertised from the secondary routing instance **hub_downstream**. See the routing instances configuration guidelines in the *Junos OS Routing Protocols Configuration Guide* for more information about the **no-vrf-advertise** statement.

The **auto-export** statement at the **[edit routing-instances hub-downstream routing-options]** hierarchy level identifies routes exported from the hub instance to the hub-downstream instance by looking at the route targets defined for each routing instance. See the routing instances configuration guidelines in the *Junos OS Routing Protocols Configuration Guide* for more information about using the **auto-export** statement. See Configuring Overlapping VPNs Using Automatic Route Export for more examples of export policy.

With this configuration on hub PE, spoke-to-spoke CE traffic goes through the hub CE and permits egress features (such as filtering) to be enabled on the hub PE.

In this configuration example, traffic forwarding is as follows between spoke CE2 and spoke CE3:

1. Spoke CE2 forwards traffic using the default route learned from spoke PE2 through BGP.

```
0.0.0.0/0          *[BGP/170] 02:24:15, localpref 100
                      AS path: 200 200 I
                    > to 10.49.3.1 via t1-3/0/1.0
```

2. Spoke PE2 does a route lookup in the spoke VRF table and forwards the traffic to hub PE1 (through the P router—PE2 pushes two labels) using the default route learned through BGP.

```
spoke.inet.0: 5 destinations, 5 routes (5 active, 0 holddown, 0 hidden)

+ = Active Route, - = Last Active, * = Both

0.0.0.0/0          *[BGP/170] 00:00:09, localpref 100, from
10.255.14.176
                      AS path: 100 I
                    > via t3-0/0/0.0, Push 1029, Push 100224(top)
```

3. Hub PE1 does a route lookup in the **mpls.0 table** for the VPN label **1029**.

```
mpls.0: 5 destinations, 5 routes (5 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both
```

```
1029               *[VPN/0] 00:11:49
                      to table hub_downstream.inet.0, Pop
```

The VPN label **1029** is advertised because:

a. The **vrf-table-label** statement is applied at the **[edit routing-instances hub_downsteam]** hierarchy level in the hub PE1 configuration.

b. The **no-vrf-advertise** statement is applied at the **[edit routing-instances hub]** hierarchy level, instructing the router to advertise the route from the secondary table.

Therefore, IP lookups are performed in the **hub_downstream.inet.0** table, not in the **hub.inet.0** table.

Issue the **show route advertising-protocol** command on the hub PE to a spoke PE to verify the VPN label **1029** advertisement:

```
user@host> show route advertising-protocol

   hub_downstream.inet.0: 2 destinations, 2 routes (2 active, 0 holddown,
0 hidden)
    * 0.0.0.0/0 (1 entry, 1 announced)
     BGP group ibgp type Internal
         Route Distinguisher: 10.255.14.176:3
         VPN Label: 1029
         Nexthop: Self
         Localpref: 100
         AS path: 100 I
         Communities: target:200:101
```

4. Hub PE1 performs an IP lookup in the **hub_downstream.inet.0** table and forwards the traffic out interface **t3-0/0/0.0** to hub CE1.

```
   hub_downstream.inet.0: 2 destinations, 2 routes (2 active, 0 holddown,
0 hidden)
    0.0.0.0/0 (1 entry, 1 announced)
            *BGP    Preference: 170/-101
                    Next-hop reference count: 4
                    Source: 10.49.4.2
                    Next hop: 10.49.4.2 via t3-0/0/0.0, selected
                    State: <Secondary Active Ext>
                    Peer AS:   100
                    Age: 3:03
                    Task: BGP_100.10.49.4.2+1707
                    Announcement bits (2): 0-KRT 2-BGP.0.0.0.0+179
                    AS path: 100 I
                    Communities: target:200:101
                    Localpref: 100
                    Router ID: 10.49.10.251
                    Primary Routing Table hub.inet.0
```

The primary routing table is **hub.inet.0**, indicating that this route was exported from table **hub.inet.0** into this hub_downstream.inet.0 table as a result of the **no-vrf-advertise** statement at the **[edit routing-instances hub]** hierarchy level and the **auto-export** statement at the **[edit routing-instances hub-downstream routing-options]** hierarchy level in the hub PE1 configuration.

5. Hub CE1 forwards the traffic back to hub PE1 using the router learned through BGP.

```
10.49.10.253/32    *[BGP/170] 02:40:03, localpref 100
                      AS path: 200 200 I
                    > to 10.49.4.1 via t3-3/1/0.0
```

6. Hub PE1 performs a route lookup in the hub VRF table and forwards the traffic to spoke PE3 (through the P router—PE1 pushes two labels).

```
    10.49.10.253/32    *[BGP/170] 01:41:05, localpref 100, from
10.255.14.178
                          AS path: 100 I
                        > via t1-0/1/0.0, Push 100128, Push 100192(top)
```

7. Spoke PE3 performs a route lookup in the mpls.0 table for VPN label **100128**.

```
100128             *[VPN/170] 02:41:30
                    > to 10.49.6.2 via t3-0/0/0.0, Pop
```

8. Spoke PE3 forwards traffic out interface **t3-0/0/0.0** to spoke CE3.

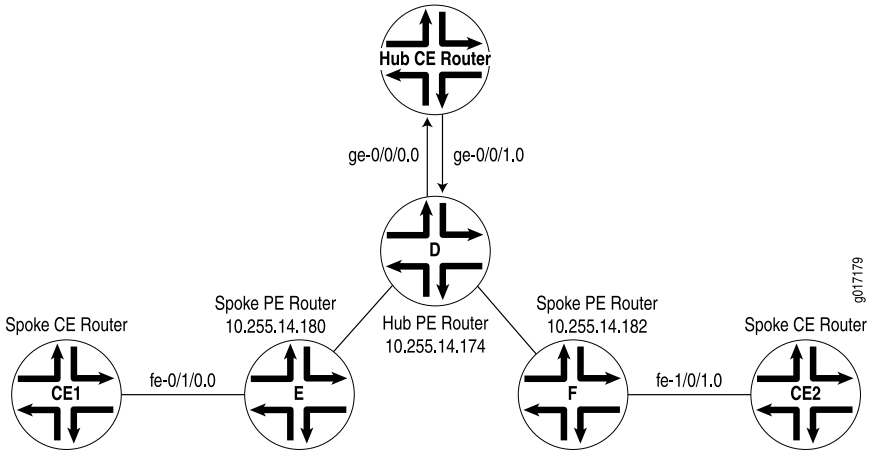
Configuring Hub-and-Spoke VPN Topologies: Two Interfaces

Use a two-interface configuration to propagate routes from spoke to spoke.

The example in this section configures a hub-and-spoke topology with two interfaces using the following components (see Figure 18 on page 117):

- One hub PE router (Router D).
- One hub CE router connected to the hub PE router. For this hub-and-spoke VPN topology to function properly, there must be two interfaces connecting the hub PE router to the hub CE router, and each interface must have its own VRF table on the PE router:
  - The first interface (here, interface **ge-0/0/0.0**) is used to announce spoke routes to the hub CE router. The VRF table associated with this interface contains the routes being announced by the spoke PE routers to the hub CE router.
  - The second interface (here, interface **ge-0/0/1.0**) is used to receive route announcements from the hub CE that are destined for the hub-and-spoke routers. The VRF table associated with this interface contains the routes announced by the hub CE router to the spoke PE routers. For this example, two separate physical interfaces are used. It would also work if you were to configure two separate logical interfaces sharing the same physical interface between the hub PE router and the hub CE router.
- Two spoke PE routers (Router E and Router F).
- Two spoke CE routers (CE1 and CE2), one connected to each spoke PE router.
- LDP as the signaling protocol.

Figure 18: Example of a Hub-and-Spoke VPN Topology with Two Interfaces



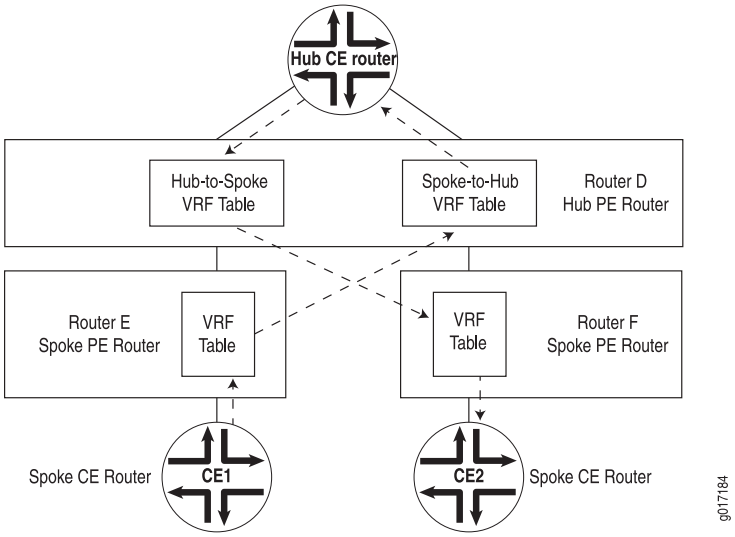
In this configuration, route distribution from spoke CE Router CE1 occurs as follows:

1. Spoke Router CE1 announces its routes to spoke PE Router E.
2. Router E installs the routes from CE1 into its VRF table.
3. After checking its VRF export policy, Router E adds the spoke target community to the routes from Router CE1 that passed the policy and announces them to the hub PE router, Router D.
4. Router D checks the VRF import policy associated with interface **ge-0/0/0.0** and places all routes from spoke PE routers that match the policy into its **bgp.l3vpn** routing table. (Any routes that do not match are discarded.)
5. Router D checks its VRF import policy associated with interface **ge-0/0/0.0** and installs all routes that match into its spoke VRF table. The routes are installed with the spoke target community.
6. Router D announces routes to the hub CE over interface **ge-0/0/0**.
7. The hub CE router announces the routes back to the hub PE Router D over the second interface to the hub router, interface **ge-0/0/1**.
8. The hub PE router installs the routes learned from the hub CE router into its hub VRF table, which is associated with interface **ge-0/0/1**.
9. The hub PE router checks the VRF export policy associated with interface **ge-0/0/1.0** and announces all routes that match to all spokes after adding the hub target community.

Figure 19 on page 118 illustrates how routes are distributed from this spoke router to the other spoke CE router, Router CE2. The same path is followed if you issue a **traceroute** command from Router CE1 to Router CE2.

The final section in this example, "Hub-and-Spoke VPN Configuration Summarized by Router" on page 126, consolidates the statements needed to configure VPN functionality for each of the service provider routers shown in Figure 18 on page 117.

Figure 19: Route Distribution Between Two Spoke Routers



The following sections explain how to configure the VPN functionality for a hub-and-spoke topology on the hub-and-spoke PE routers. The CE routers do not have any information about the VPN, so you configure them normally.

- Enabling an IGP on the Hub-and-Spoke PE Routers on page 118
- Configuring LDP on the Hub-and-Spoke PE Routers on page 119
- Configuring IBGP on the PE Routers on page 119
- Configuring VPN Routing Instances on the Hub-and-Spoke PE Routers on page 120
- Configuring VPN Policy on the PE Routers on page 123
- Hub-and-Spoke VPN Configuration Summarized by Router on page 126

Enabling an IGP on the Hub-and-Spoke PE Routers

To allow the hub-and-spoke PE routers to exchange routing information, you must configure an IGP on all these routers or you must configure static routes. You configure the IGP on the master instance of the routing protocol process (**rpd**) (that is, at the **[edit protocols]** hierarchy level), not within the routing instance (that is, not at the **[edit routing-instances]** hierarchy level).

You configure the IGP in the standard way. This configuration example does not include this portion of the configuration.

In the route distribution in a hub-and-spoke topology, if the protocol used between the CE and PE routers at the hub site is BGP, the hub CE router announces all routes received from the hub PE router and the spoke routers back to the hub PE router and all the spoke routers. This means that the hub-and-spoke PE routers receive routes that contain their AS number. Normally, when a route contains this information, it indicates that a routing loop has occurred and the router rejects the routes. However, for the VPN configuration to work, the hub PE router and the spoke routers must accept these routes. To enable this, include the **loops** option when configuring the AS at the **[edit routing-options]**

hierarchy level on the hub PE router and all the spoke routers. For this example configuration, you specify a value of 1. You can specify a number from 0 through 10.

```
[edit routing-options]
autonomous-system as-number loops 1;
```

Configuring LDP on the Hub-and-Spoke PE Routers

Configure LDP on the interfaces between the hub-and-spoke PE routers that participate in the VPN.

On hub PE Router D, configure LDP:

```
[edit protocols]
ldp {
  interface so-1/0/0.0;
  interface t3-1/1/0.0;
}
```

On spoke PE Router E, configure LDP:

```
[edit protocols]
ldp {
  interface fe-0/1/2.0;
}
```

On spoke PE router Router F, configure LDP:

```
[edit protocols]
ldp {
  interface fe-1/0/0.0;
}
```

Configuring IBGP on the PE Routers

On the hub-and-spoke PE routers, configure an IBGP session with the following properties:

- VPN family—To indicate that the IBGP session is for the VPN, include the **family inet-vpn** statement.
- Loopback address—Include the **local-address** statement, specifying the local PE router's loopback address. The IBGP session for VPNs runs through the loopback address. You must also configure the **lo0** interface at the **[edit interfaces]** hierarchy level. The example does not include this part of the router's configuration.
- Neighbor address—Include the **neighbor** statement. On the hub router, specify the IP address of each spoke PE router, and on the spoke router, specify the address of the hub PE router.

For the hub router, you configure an IBGP session with each spoke, and for each spoke router, you configure an IBGP session with the hub. There are no IBGP sessions between the two spoke routers.

On hub Router D, configure IBGP. The first **neighbor** statement configures an IBGP session to spoke Router E, and the second configures a session to spoke Router F.

```
[edit protocols]
bgp {
  group Hub-to-Spokes {
    type internal;
    local-address 10.255.14.174;
    family inet-vpn {
      unicast;
    }
    neighbor 10.255.14.180;
    neighbor 10.255.14.182;
  }
}
```

On spoke Router E, configure an IBGP session to the hub router:

```
[edit protocols]
bgp {
  group Spoke-E-to-Hub {
    type internal;
    local-address 10.255.14.180;
    neighbor 10.255.14.174 {
      family inet-vpn {
        unicast;
      }
    }
  }
}
```

On spoke Router F, configure an IBGP session to the hub router:

```
[edit protocols]
bgp {
  group Spoke-F-to-Hub {
    type internal;
    local-address 10.255.14.182;
    neighbor 10.255.14.174 {
      family inet-vpn {
        unicast;
      }
    }
  }
}
```

Configuring VPN Routing Instances on the Hub-and-Spoke PE Routers

For the hub PE router to be able to distinguish between packets going to and coming from the spoke PE routers, you must configure it with two routing instances:

- One routing instance (in this example, **Spokes-to-Hub-CE**) is associated with the interface that carries packets from the hub PE router to the hub CE router (in this example, interface **ge-0/0/0.0**). Its VRF table contains the routes being announced by the spoke PE routers and the hub PE router to the hub CE router.
- The second routing instance (in this example, **Hub-CE-to-Spokes**) is associated with the interface that carries packets from the hub CE router to the hub PE router (in this

example, interface **ge-0/0/1.0**). Its VRF table contains the routes being announced from the hub CE router to the hub-and-spoke PE routers.

On each spoke router, you must configure one routing instance.

You must define the following in the routing instance:

- Route distinguisher, which is used to distinguish the addresses in one VPN from those in another VPN.
- Instance type of **vrf**, which creates the VRF table on the PE router.
- Interfaces that are part of the VPN and that connect the PE routers to their CE routers.
- VRF import and export policies. Both import policies must include reference to a community. Otherwise, when you try to commit the configuration, the commit fails. (The exception to this is if the import policy contains only a **then reject** statement.) In the VRF export policy, spoke PE routers attach the spoke target community.
- Routing between the PE and CE routers, which is required for the PE router to distribute VPN-related routes to and from connected CE routers. You can configure a routing protocol—BGP, OSPF, or RIP—or you can configure static routing.

For a hub-and-spoke topology, you must configure different policies in each routing instance on the hub CE router. For the routing instance associated with the interface that carries packets from the hub PE router to the hub CE router (in this example, **Spokes-to-Hub-CE**), the import policy must accept all routes received on the IBGP session between the hub-and-spoke PE routers, and the export policy must reject all routes received from the hub CE router. For the routing instance associated with the interface that carries packets from the hub CE router to the hub PE router (in this example, **Hub-CE-to-Spokes**), the import policy must reject all routes received from the spoke PE routers, and the export policy must export to all the spoke routers.

On hub PE Router D, configure the following routing instances. Router D uses OSPF to distribute routes to and from the hub CE router.

```
[edit]
routing-instance {
   Spokes-to-Hub-CE {
      instance-type vrf;
      interface ge-0/0/0.0;
      route-distinguisher 10.255.1.174:65535;
      vrf-import spoke;
      vrf-export null;
      protocols {
         ospf {
            export redistribute-vpn;
            domain-vpn-tag 0;
            area 0.0.0.0 {
               interface ge-0/0/0;
            }
         }
      }
   }
   Hub-CE-to-Spokes {
```

```
    instance-type vrf;
    interface ge-0/0/1.0;
    route-distinguisher 10.255.1.174:65535;
    vrf-import null;
    vrf-export hub;
    protocols {
      ospf {
        export redistribute-vpn;
        area 0.0.0.0 {
          interface ge-0/0/1.0;
        }
      }
    }
  }
}
```

On spoke PE Router E, configure the following routing instances. Router E uses OSPF to distribute routes to and from spoke CE Router CE1.

```
[edit]
routing-instance {
  Spoke-E-to-Hub {
    instance-type vrf;
    interface fe-0/1/0.0;
    route-distinguisher 10.255.14.80:65535;
    vrf-import hub;
    vrf-export spoke;
    protocols {
      ospf {
        export redistribute-vpn;
        area 0.0.0.0 {
          interface fe-0/1/0.0;
        }
      }
    }
  }
}
```

On spoke PE Router F, configure the following routing instances. Router F uses OSPF to distribute routes to and from spoke CE Router CE2.

```
[edit]
routing-instance {
  Spoke-F-to-Hub {
    instance-type vrf;
    interface fe-1/0/1.0;
    route-distinguisher 10.255.14.182:65535;
    vrf-import hub;
    vrf-export spoke;
    protocols {
      ospf {
        export redistribute-vpn;
        area 0.0.0.0 {
          interface fe-1/0/1.0;
        }
      }
```

```
    }
  }
}
```

Configuring VPN Policy on the PE Routers

You must configure VPN import and export policies on each of the hub-and-spoke PE routers so that they install the appropriate routes in the VRF tables, which they use to forward packets within each VPN.

On the spoke routers, you define policies to exchange routes with the hub router.

On the hub router, you define policies to accept routes from the spoke PE routers and distribute them to the hub CE router, and vice versa. The hub PE router has two VRF tables:

- Spoke-to-hub VRF table—Handles routes received from spoke routers and announces these routes to the hub CE router. For this VRF table, the import policy must check that the spoke target name is present and that the route was received from the IBGP session between the hub PE and the spoke PE routers. This VRF table must not export any routes, so its export policy should reject everything.
- Hub-to-spoke VRF table—Handles routes received from the hub CE router and announces them to the spoke routers. For this VRF table, the export policy must add the hub target community. This VRF table must not import any routes, so its import policy should reject everything.

In the VPN policy, you also configure the VPN target communities.

On hub PE Router D, configure the following policies to apply to the VRF tables:

- **spoke**—Accepts routes received from the IBGP session between it and the spoke PE routers that contain the community target **spoke**, and rejects all other routes.
- **hub**—Adds the community target hub to all routes received from OSPF (that is, from the session between it and the hub CE router). It rejects all other routes.
- **null**—Rejects all routes.
- **redistribute-vpn**—Redistributes OSPF routes to neighbors within the routing instance.

```
[edit]
policy-options {
  policy-statement spoke {
    term a {
      from {
        protocol bgp;
        community spoke;
      }
      then accept;
    }
    term b {
      then reject;
    }
  }
  policy-statement hub {
```

```
        term a {
          from protocol ospf;
          then {
            community add hub;
            accept;
          }
        }
        term b {
          then reject;
        }
      }
      policy-statement null {
        then reject;
      }
      policy-statement redistribute-vpn {
        term a {
          from protocol bgp;
          then accept;
        }
        term b {
          then reject;
        }
      }
      community hub members target:65535:1;
      community spoke members target:65535:2;
    }
```

To apply the VRF policies on Router D, include the **vrf-export** and **vrf-import** statements when you configure the routing instances:

```
[edit]
routing-instance {
  Spokes-to-Hub-CE {
    vrf-import spoke;
    vrf-export null;
  }
  Hub-CE-to-Spokes {
    vrf-import null;
    vrf-export hub;
  }
}
```

On spoke PE Router E and Router F, configure the following policies to apply to the VRF tables:

- **hub**—Accepts routes received from the IBGP session between it and the hub PE routers that contain the community target **hub**, and rejects all other routes.
- **spoke**—Adds the community target spoke to all routes received from OSPF (that is, from the session between it and the hub CE router) rejects all other routes.
- **redistribute-vpn**—Redistributes OSPF routes to neighbors within the routing instance.

On spoke PE Router E and Router F, configure the following VPN import and export policies:

```
[edit]
policy-options {
  policy-statement hub {
    term a {
      from {
        protocol bgp;
        community hub;
      }
      then accept;
    }
    term b {
      then reject;
    }
  }
  policy-statement spoke {
    term a {
      from protocol ospf;
      then {
        community add spoke;
        accept;
      }
    }
    term b {
      then reject;
    }
  }
  policy-statement redistribute-vpn {
    term a {
      from protocol bgp;
      then accept;
    }
    term b {
      then reject;
    }
  }
  community hub members target:65535:1;
  community spoke members target 65535:2;
}
```

To apply the VRF policies on the spoke routers, include the **vrf-export** and **vrf-import** statements when you configure the routing instances:

```
[edit]
routing-instance {
  Spoke-E-to-Hub {
    vrf-import hub;
    vrf-export spoke;
  }
}
[edit]
routing-instance {
  Spoke-F-to-Hub {
    vrf-import hub;
    vrf-export spoke;
  }
}
```

Hub-and-Spoke VPN Configuration Summarized by Router

Router D (Hub PE Router)

Routing Instance for Distributing Spoke Routes to Hub CE

```
Spokes-to-Hub-CE {
  instance-type vrf;
  interface fe-0/0/0.0;
  route-distinguisher 10.255.1.174:65535;
  vrf-import spoke;
  vrf-export null;
}
```

Instance Routing Protocol

```
protocols {
  ospf {
    domain-vpn-tag 0;
    export redistribute-vpn;
    area 0.0.0.0 {
      interface ge-0/0/0.0;
    }
  }
}
```

Routing Instance for Distributing Hub CE Routes to Spokes

```
Hub-CE-to-Spokes {
  instance-type vrf;
  interface ge-0/0/1.0;
  route-distinguisher 10.255.1.174:65535;
  vrf-import null;
  vrf-export hub;
}
```

Routing Instance Routing Protocols

```
protocols {
  ospf {
    export redistribute-vpn;
    area 0.0.0.0 {
      interface ge-0/0/1.0;
    }
  }
}
```

Routing Options (Master Instance)

```
routing-options {
  autonomous-system 1 loops 1;
}
```

Protocols (Master Instance)

```
protocols {
}
```

Enable LDP

```
ldp {
  interface so-1/0/0.0;
  interface t3-1/1/0.0;
}
```

Configure IBGP

```
bgp {
  group Hub-to-Spokes {
    type internal;
```

```
    local-address 10.255.14.174;
    family inet-vpn {
      unicast;
    }
    neighbor 10.255.14.180;
    neighbor 10.255.14.182;
  }
}
```

Configure VPN Policy

```
policy-options {
  policy-statement spoke {
    term a {
      from {
        protocol bgp;
        community spoke;
      }
      then accept;
    }
    term b {
      then reject;
    }
  }
  policy-statement hub {
    term a {
      from protocol ospf;
      then {
        community add hub;
        accept;
      }
    }
    term b {
      then reject;
    }
  }
  policy-statement null {
    then reject;
  }
  policy-statement redistribute-vpn {
    term a {
      from protocol bgp;
      then accept;
    }
    term b {
      then reject;
    }
  }
  community hub members target:65535:1;
  community spoke members target:65535:2;
}
```

Router E (Spoke PE Router)

Routing Instance

```
routing-instance {
  Spoke-E-to-Hub {
    instance-type vrf;
```

```
    interface fe-0/1/0.0;
    route-distinguisher 10.255.14.80:65535;
    vrf-import hub;
    vrf-export spoke;
  }
}
```

Instance Routing Protocol

```
protocols {
  ospf {
    export redistribute-vpn;
    area 0.0.0.0 {
      interface fe-0/1/0.0;
    }
  }
}
```

Routing Options (Master Instance)

```
routing-options {
  autonomous-system 1 loops 1;
}
```

Protocols (Master Instance)

```
protocols {
}
```

Enable LDP

```
ldp {
  interface fe-0/1/2.0;
}
```

Configure IBGP

```
bgp {
  group Spoke-E-to-Hub {
    type internal;
    local-address 10.255.14.180;
    neighbor 10.255.14.174 {
      family inet-vpn {
        unicast;
      }
    }
  }
}
```

Configure VPN Policy

```
policy-options {
  policy-statement hub {
    term a {
      from {
        protocol bgp;
        community hub;
      }
      then accept;
    }
    term b {
      then reject;
    }
  }
  policy-statement spoke {
    term a {
      from protocol ospf;
```

```
        then {
          community add spoke;
          accept;
        }
      }
      term b {
        then reject;
      }
    }
    policy-statement redistribute-vpn {
      term a {
        from protocol bgp;
        then accept;
      }
      term b {
        then reject;
      }
    }
    community hub members target:65535:1;
    community spoke members target:65535:2;
}
```

Router F (Spoke PE Router)

Routing Instance

```
routing-instance {
  Spoke-F-to-Hub {
    instance-type vrf;
    interface fe-1/0/1.0;
    route-distinguisher 10.255.14.182:65535;
    vrf-import hub;
    vrf-export spoke;
  }
}
```

Instance Routing Protocol

```
protocols {
  ospf {
    export redistribute-vpn;
    area 0.0.0.0 {
      interface fe-1/0/1.0;
    }
  }
}
```

Routing Options (Master Instance)

```
routing-options {
  autonomous-system 1 loops 1;
}
```

Protocols (Master Instance)

```
protocols {
}
```

Enable LDP

```
ldp {
  interface fe-1/0/0.0;
}
```

Configure IBGP

```
bgp {
  group Spoke-F-to-Hub {
    type internal;
    local-address 10.255.14.182;
    neighbor 10.255.14.174 {
      family inet-vpn {
        unicast;
      }
    }
  }
}
```

Configure VPN Policy

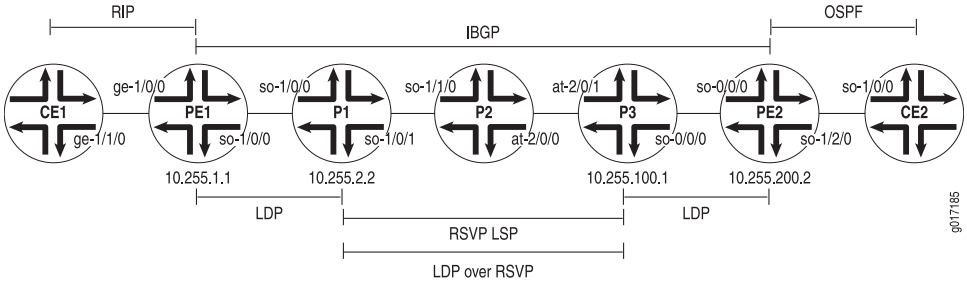
```
policy-options {
  policy-statement hub {
    term a {
      from {
        protocol bgp;
        community hub;
      }
      then accept;
    }
    term b {
      then reject;
    }
  }
  policy-statement spoke {
    term a {
      from protocol ospf;
      then {
        community add spoke;
        accept;
      }
    }
    term b {
      then reject;
    }
  }
  policy-statement redistribute-vpn {
    term a {
      from {
        protocol bgp;
      }
      then accept;
    }
    term b {
      then reject;
    }
  }
  community hub members target:65535:1;
  community spoke members target:65535:2;
}
```

Configuring an LDP-over-RSVP VPN Topology

This example shows how to set up a VPN topology in which LDP packets are tunneled over an RSVP LSP. This configuration consists of the following components (see Figure 20 on page 131):

- One VPN (VPN-A)
- Two PE routers
- LDP as the signaling protocol between the PE routers and their adjacent P routers
- An RSVP LSP between two of the P routers over which LDP is tunneled

Figure 20: Example of an LDP-over-RSVP VPN Topology



The following steps describe how this topology is established and how packets are sent from CE Router CE2 to CE Router CE1:

1. The P routers P1 and P3 establish RSVP LSPs between each other and install their loopback addresses in their **inet.3** routing tables.
2. PE Router PE1 establishes an LDP session with Router P1 over interface **so-1/0/0.0**.
3. Router P1 establishes an LDP session with Router P3's loopback address, which is reachable using the RSVP LSP.
4. Router P1 sends its label bindings, which include a label to reach Router PE1, to Router P3. These label bindings allow Router P3 to direct LDP packets to Router PE1.
5. Router P3 establishes an LDP session with Router PE2 over interface **so0-0/0/0.0** and establishes an LDP session with Router P1's loopback address.
6. Router P3 sends its label bindings, which include a label to reach Router PE2, to Router P1. These label bindings allow Router P1 to direct LDP packets to Router PE2's loopback address.
7. Routers PE1 and PE2 establish IBGP sessions with each other.
8. When Router PE1 announces to Router PE2 routes that it learned from Router CE1, it includes its VPN label. (The PE router creates the VPN label and binds it to the interface between the PE and CE routers.) Similarly, when Router PE2 announces routes that it learned from Router CE2, it sends its VPN label to Router PE1.

When Router PE2 wants to forward a packet to Router CE1, it pushes two labels onto the packet's label stack: first the VPN label that is bound to the interface between Router

PE1 and Router CE1, then the LDP label used to reach Router PE1. Then it forwards the packets to Router P3 over interface **so-0/0/1.0**.

1. When Router P3 receives the packets from Router PE2, it swaps the LDP label that is on top of the stack (according to its LDP database) and also pushes an RSVP label onto the top of the stack so that the packet can now be switched by the RSVP LSP. At this point, there are three labels on the stack: the inner (bottom) label is the VPN label, the middle is the LDP label, and the outer (top) is the RSVP label.
2. Router P2 receives the packet and switches it to Router P1 by swapping the RSVP label. In this topology, because Router P2 is the penultimate-hop router in the LSP, it pops the RSVP label and forwards the packet over interface **so-1/1/0.0** to Router P1. At this point, there are two labels on the stack: The inner label is the VPN label, and the outer one is the LDP label.
3. When Router P1 receives the packet, it pops the outer label (the LDP label) and forwards the packet to Router PE1 using interface **so-1/0/0.0**. In this topology, Router PE1 is the egress LDP router, so Router P1 pops the LDP label instead of swapping it with another label. At this point, there is only one label on the stack, the VPN label.
4. When Router PE1 receives the packet, it pops the VPN label and forwards the packet as an IPv4 packet to Router CE1 over interface **ge-1/1/0.0**.

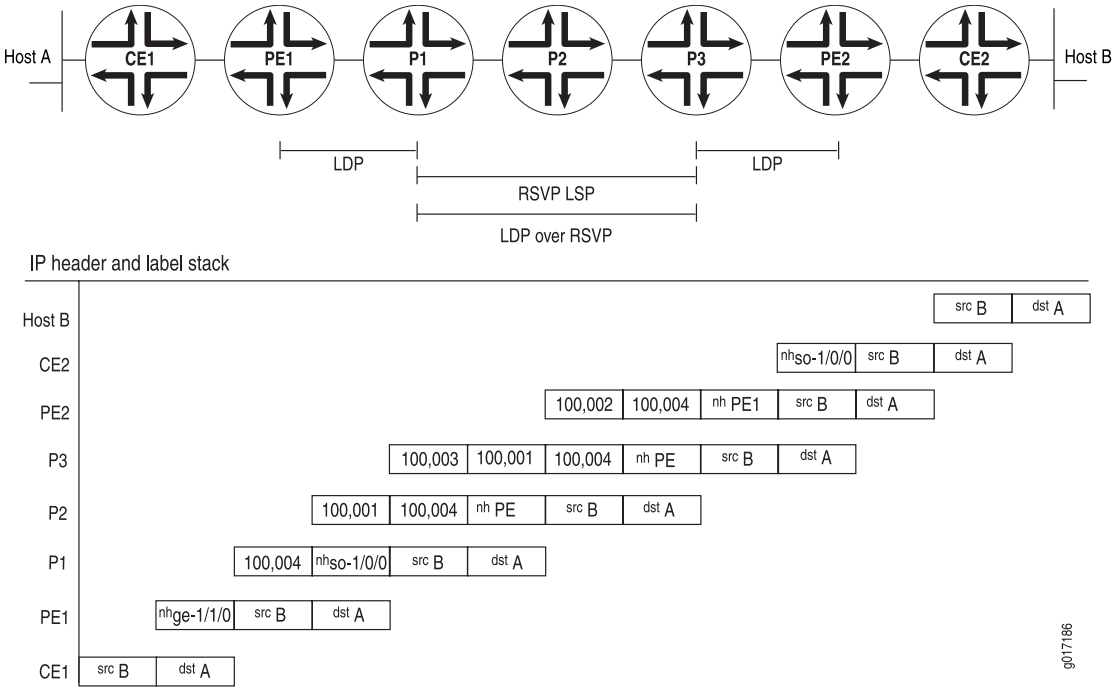
A similar set of operations occurs for packets sent from Router CE1 that are destined for Router CE2.

The following list explains how, for packets being sent from Router CE2 to Router CE1, the LDP, RSVP, and VPN labels are announced by the various routers. These steps include examples of label values (illustrated in Figure 21 on page 133).

- LDP labels
  - Router PE1 announces LDP label 3 for itself to Router P1.
  - Router P1 announces LDP label 100,001 for Router PE1 to Router P3.
  - Router P3 announces LDP label 100,002 for Router PE1 to Router PE2.
- RSVP labels
  - Router P1 announces RSVP label 3 to Router P2.
  - Router P2 announces RSVP label 100,003 to Router P3.
- VPN label

- Router PE1 announces VPN label 100,004 to Router PE2 for the route from Router CE1 to Router CE2.

Figure 21: Label Pushing and Popping



For a packet sent from Host B in Figure 21 on page 133 to Host A, the packet headers and labels change as the packet travels to its destination:

1. The packet that originates from Host B has a source address of B and a destination address of A in its header.
2. Router CE2 adds to the packet a next hop of interface **so-1/0/0**.
3. Router PE2 swaps out the next hop of interface **so-1/0/0** and replaces it with a next hop of PE1. It also adds two labels for reaching Router PE1, first the VPN label (100,004), then the LDP label (100,002). The VPN label is thus the inner (bottom) label on the stack, and the LDP label is the outer label.
4. Router P3 swaps out the LDP label added by Router PE2 (100,002) and replaces it with its LDP label for reaching Router PE1 (100,001). It also adds the RSVP label for reaching Router P2 (100,003).
5. Router P2 removes the RSVP label (100,003) because it is the penultimate hop in the MPLS LSP.
6. Router P1 removes the LDP label (100,001) because it is the penultimate LDP router. It also swaps out the next hop of PE1 and replaces it with the next-hop interface, **so-1/0/0**.

7. Router PE1 removes the VPN label (100,004). It also swaps out the next-hop interface of **so-1/0/0** and replaces it with its next-hop interface, **ge-1/1/0**.
8. Router CE1 removes the next-hop interface of **ge-1/1/0**, and the packet header now contains just a source address of B and a destination address of A.

The final section in this example consolidates the statements needed to configure VPN functionality on each of the service P routers shown in Figure 20 on page 131.

NOTE: **In this example, a private AS number is used for the route distinguisher and the route target. This number is used for illustration only. When you are configuring VPNs, you should use an assigned AS number.**

The following sections explain how to configure the VPN functionality on the PE and P routers. The CE routers do not have any information about the VPN, so you configure them normally.

- Enabling an IGP on the PE and P Routers on page 134
- Enabling LDP on the PE and P Routers on page 134
- Enabling RSVP and MPLS on the P Router on page 136
- Configuring the MPLS LSP Tunnel Between the P Routers on page 136
- Configuring IBGP on the PE Routers on page 137
- Configuring Routing Instances for VPNs on the PE Routers on page 138
- Configuring VPN Policy on the PE Routers on page 139
- LDP-over-RSVP VPN Configuration Summarized by Router on page 141

Enabling an IGP on the PE and P Routers

To allow the PE and P routers to exchange routing information among themselves, you must configure an IGP on all these routers or you must configure static routes. You configure the IGP on the master instance of the routing protocol process (**rpd**) (that is, at the **[edit protocols]** hierarchy level), not within the VPN routing instance (that is, not at the **[edit routing-instances]** hierarchy level).

You configure the IGP in the standard way. This configuration example does not include this portion of the configuration.

Enabling LDP on the PE and P Routers

In this configuration example, the LDP is the signaling protocol between the PE routers. For the VPN to function, you must configure LDP on the two PE routers and on the P routers that are connected to the PE routers. You need to configure LDP only on the interfaces in the core of the service provider's network; that is, between the PE and P routers and between the P routers. You do not need to configure LDP on the interface between the PE and CE routers.

In this configuration example, you configure LDP on the P routers' loopback interfaces because these are the interfaces on which the MPLS LSP is configured.

On the PE routers, you must also configure **family inet** when you configure the logical interface.

On Router PE1, configure LDP:

```
[edit protocols]
ldp {
   interface so-1/0/0.0;
}
[edit interfaces]
so-1/0/0 {
   unit 0 {
      family mpls;
   }
}
```

On Router PE2, configure LDP:

```
[edit protocols]
ldp {
   interface so-0/0/0.0;
}
[edit interfaces]
so-0/0/1 {
   unit 0 {
      family mpls;
   }
}
```

On Router P1, configure LDP:

```
[edit protocols]
ldp {
   interface so-1/0/0.0;
   interface lo0;
}
```

On Router P3, configure LDP:

```
[edit protocols]
ldp {
   interface lo0;
   interface so-0/0/0.0;
}
```

On Router P2, although you do not need to configure LDP, you can optionally configure it to provide a fallback LDP path in case the RSVP LSP becomes nonoperational:

```
[edit protocols]
ldp {
   interface so-1/1/0.0;
   interface at-2/0/0.0;
}
```

Enabling RSVP and MPLS on the P Router

On the P Router P2 you must configure RSVP and MPLS because this router exists on the MPLS LSP path between the P Routers P1 and P3:

```
[edit]
protocols {
   rsvp {
      interface so-1/1/0.0;
      interface at-2/0/0.0;
   }
   mpls {
      interface so-1/1/0.0;
      interface at-2/0/0.0;
   }
}
```

Configuring the MPLS LSP Tunnel Between the P Routers

In this configuration example, LDP is tunneled over an RSVP LSP. Therefore, in addition to configuring RSVP, you must enable traffic engineering support in an IGP, and you must create an MPLS LSP to tunnel the LDP traffic.

On Router P1, enable RSVP and configure one end of the MPLS LSP tunnel. In this example, traffic engineering support is enabled for OSPF, and you configure MPLS on the interfaces to the LSP and to Router PE1. In the **to** statement, you specify the loopback address of Router P3.

```
[edit]
protocols {
   rsvp {
      interface so-1/0/1.0;
   }
   mpls {
      label-switched-path P1-to-P3 {
         to 10.255.100.1;
         ldp-tunneling;
      }
      interface so-1/0/0.0;
      interface so-1/0/1.0;
   }
   ospf {
      traffic-engineering;
      area 0.0.0.0 {
         interface so-1/0/0.0;
         interface so-1/0/1.0;
      }
   }
}
```

On Router P3, enable RSVP and configure the other end of the MPLS LSP tunnel. Again, traffic engineering support is enabled for OSPF, and you configure MPLS on the interfaces to the LSP and to Router PE2. In the **to** statement, you specify the loopback address of Router P1.

```
[edit]
protocols {
  rsvp {
    interface at-2/0/1.0;
  }
  mpls {
    label-switched-path P3-to-P1 {
      to 10.255.2.2;
      ldp-tunneling;
    }
    interface at-2/0/1.0;
    interface so-0/0/0.0;
  }
  ospf {
    traffic-engineering;
    area 0.0.0.0 {
      interface at-2/0/1.0;
      interface so-0/0/0.0;
    }
  }
}
```

Configuring IBGP on the PE Routers

On the PE routers, configure an IBGP session with the following properties:

- VPN family—To indicate that the IBGP session is for the VPN, include the **family inet-vpn** statement.
- Loopback address—Include the **local-address** statement, specifying the local PE router's loopback address. The IBGP session for VPNs runs through the loopback address. You must also configure the **lo0** interface at the **[edit interfaces]** hierarchy level. The example does not include this part of the router's configuration.
- Neighbor address—Include the **neighbor** statement, specifying the IP address of the neighboring PE router, which is its loopback (**lo0**) address.

On Router PE1, configure IBGP:

```
[edit]
protocols {
  bgp {
    group PE1-to-PE2 {
      type internal;
      local-address 10.255.1.1;
      family inet-vpn {
        unicast;
      }
      neighbor 10.255.200.2;
    }
  }
}
```

On Router PE2, configure IBGP:

```
[edit]
```

```
protocols {
  bgp {
    group PE2-to-PE1 {
      type internal;
      local-address 10.255.200.2;
      family inet-vpn {
        unicast;
      }
      neighbor 10.255.1.1;
    }
  }
}
```

Configuring Routing Instances for VPNs on the PE Routers

Both PE routers service VPN-A, so you must configure one routing instance on each router for the VPN in which you define the following:

- Route distinguisher, which must be unique for each routing instance on the PE router. It is used to distinguish the addresses in one VPN from those in another VPN.
- Instance type of **vrf**, which creates the VRF table on the PE router.
- Interfaces connected to the CE routers.
- VRF import and export policies, which must be the same on each PE router that services the same VPN. Unless the import policy contains only a **then reject** statement, it must include reference to a community. Otherwise, when you try to commit the configuration, the commit fails.

NOTE: In this example, a private AS number is used for the route distinguisher. This number is used for illustration only. When you are configuring VPNs, you should use an assigned AS number.

- Routing between the PE and CE routers, which is required for the PE router to distribute VPN-related routes to and from connected CE routers. You can configure a routing protocol—BGP, OSPF, or RIP—or you can configure static routing.

On Router PE1, configure the following routing instance for VPN-A. In this example, Router PE1 uses RIP to distribute routes to and from the CE router to which it is connected.

```
[edit]
routing-instance {
  VPN-A {
    instance-type vrf;
    interface ge-1/0/0.0;
    route-distinguisher 65535:0;
    vrf-import VPN-A-import;
    vrf-export VPN-A-export;
    protocols {
      rip {
        group PE1-to-CE1 {
          neighbor ge-1/0/0.0;
        }
```

```
        }
      }
    }
  }
```

On Router PE2, configure the following routing instance for VPN-A. In this example, Router PE2 uses OSPF to distribute routes to and from the CE router to which it is connected.

```
[edit]
routing-instance {
  VPN-A {
    instance-type vrf;
    interface so-1/2/0.0;
    route-distinguisher 65535:1;
    vrf-import VPN-A-import;
    vrf-export VPN-A-export;
    protocols {
      ospf {
        area 0.0.0.0 {
          interface so-1/2/0.0;
        }
      }
    }
  }
}
```

Configuring VPN Policy on the PE Routers

You must configure VPN import and export policies on each of the PE routers so that they install the appropriate routes in their VRF tables, which they use to forward packets within a VPN. For VPN-A, the VRF table is **VPN-A.inet.0**.

In the VPN policy, you also configure VPN target communities.

NOTE: In this example, a private AS number is used for the route target. This number is used for illustration only. When you are configuring VPNs, you should use an assigned AS number.

On Router PE1, configure the following VPN import and export policies:

NOTE: The policy qualifiers shown in this example are only those needed for the VPN to function. You can configure additional qualifiers, as needed, to any policies that you configure.

```
[edit]
policy-options {
  policy-statement VPN-A-import {
    term a {
      from {
        protocol bgp;
```

```
            community VPN-A;
          }
          then accept;
        }
        term b {
          then reject;
        }
      }
      policy-statement VPN-A-export {
        term a {
          from protocol rip;
          then {
            community add VPN-A;
            accept;
          }
        }
        term b {
          then reject;
        }
      }
      community VPN-A members target:65535:00;
    }
```

On Router PE2, configure the following VPN import and export policies:

```
[edit]
policy-options {
  policy-statement VPN-A-import {
    term a {
      from {
        protocol bgp;
        community VPN-A;
      }
      then accept;
    }
    term b {
      then reject;
    }
  }
  policy-statement VPN-A-export {
    term a {
      from protocol ospf;
      then {
        community add VPN-A;
        accept;
      }
    }
    term b {
      then reject;
    }
  }
  community VPN-A members target:65535:00;
}
```

To apply the VPN policies on the routers, include the **vrf-export** and **vrf-import** statements when you configure the routing instance on the PE routers. The VRF import and export

policies handle the route distribution across the IBGP session running between the PE routers.

LDP-over-RSVP VPN Configuration Summarized by Router

Router PE1

Routing Instance for VPN-A

```
routing-instance {
  VPN-A {
    instance-type vrf;
    interface ge-1/0/0.0;
    route-distinguisher 65535:0;
    vrf-import VPN-A-import;
    vrf-export VPN-A-export;
  }
}
```

Instance Routing Protocol

```
protocols {
  rip {
    group PE1-to-CE1 {
      neighbor ge-1/0/0.0;
    }
  }
}
```

Interfaces

```
interfaces {
  so-1/0/0 {
    unit 0 {
      family mpls;
    }
  }
  ge-1/0/0 {
    unit 0;
  }
}
```

Master Protocol Instance

```
protocols {
}
```

Enable LDP

```
ldp {
  interface so-1/0/0.0;
}
```

Enable MPLS

```
mpls {
  interface so-1/0/0.0;
  interface ge-1/0/0.0;
}
```

Configure IBGP

```
bgp {
  group PE1-to-PE2 {
    type internal;
    local-address 10.255.1.1;
    family inet-vpn {
      unicast;
    }
```

```
    neighbor 10.255.100.1;
  }
}
```

Configure VPN Policy

```
policy-options {
  policy-statement VPN-A-import {
    term a {
      from {
        protocol bgp;
        community VPN-A;
      }
      then accept;
    }
    term b {
      then reject;
    }
  }
  policy-statement VPN-A-export {
    term a {
      from protocol rip;
      then {
        community add VPN-A;
        accept;
      }
    }
    term b {
      then reject;
    }
  }
  community VPN-A members target:65535:00;
}
```

Router P1

Master Protocol Instance

```
protocols {
}
```

Enable RSVP

```
rsvp {
  interface so-1/0/1.0;
}
```

Enable LDP

```
ldp {
  interface so-1/0/0.0;
  interface lo0.0;
}
```

Enable MPLS

```
mpls {
  label-switched-path P1-to-P3 {
    to 10.255.100.1;
    ldp-tunneling;
  }
  interface so-1/0/0.0;
  interface so-1/0/1.0;
}
```

Configure OSPF for Traffic Engineering Support

```
ospf {
  traffic-engineering;
  area 0.0.0.0 {
    interface so-1/0/0.0;
    interface so-1/0/1.0;
  }
}
```

Router P2

Master Protocol Instance

```
protocols {
}
```

Enable RSVP

```
rsvp {
  interface so-1/1/0.0;
  interface at-2/0/0.0;
}
```

Enable MPLS

```
mpls {
  interface so-1/1/0.0;
  interface at-2/0/0.0;
}
```

Router P3

Master Protocol Instance

```
protocols {
}
```

Enable RSVP

```
rsvp {
  interface at-2/0/1.0;
}
```

Enable LDP

```
ldp {
  interface so-0/0/0.0;
  interface lo0.0;
}
```

Enable MPLS

```
mpls {
  label-switched-path P3-to-P1 {
    to 10.255.2.2;
    ldp-tunneling;
  }
  interface at-2/0/1.0;
  interface so-0/0/0.0;
}
```

Configure OSPF for Traffic Engineering Support

```
ospf {
  traffic-engineering;
  area 0.0.0.0 {
    interface at-2/0/1.0;
    interface at-2/0/1.0;
  }
}
```

Router PE2

Routing Instance for VPN-A

```
routing-instance {
  VPN-A {
    instance-type vrf;
    interface so-1/2/0.0;
    route-distinguisher 65535:1;
    vrf-import VPN-A-import;
    vrf-export VPN-A-export;
  }
}
```

Instance Routing Protocol

```
protocols {
  ospf {
    area 0.0.0.0 {
      interface so-1/2/0.0;
    }
  }
}
```

Interfaces

```
interfaces {
  so-0/0/0 {
    unit 0 {
      family mpls;
    }
  }
  so-1/2/0 {
    unit 0;
  }
}
```

Master Protocol Instance

```
protocols {
}
```

Enable LDP

```
ldp {
  interface so-0/0/0.0;
}
```

Enable MPLS

```
mpls {
  interface so-0/0/0.0;
  interface so-1/2/0.0;
}
```

Configure IBGP

```
bgp {
  group PE2-to-PE1 {
    type internal;
    local-address 10.255.200.2;
    family inet-vpn {
      unicast;
    }
    neighbor 10.255.1.1;
  }
}
```

Configure VPN Policy

```
policy-options {
  policy-statement VPN-A-import {
    term a {
      from {
        protocol bgp;
        community VPN-A;
      }
      then accept;
    }
    term b {
      then reject;
    }
  }
  policy-statement VPN-A-export {
    term a {
      from protocol ospf;
      then {
        community add VPN-A;
        accept;
      }
    }
    term b {
      then reject;
    }
  }
  community VPN-A members target:65535:01;
}
```

Configuring an Application-Based Layer 3 VPN Topology

This example illustrates an application-based mechanism for forwarding traffic into a Layer 3 VPN. Typically, one or more interfaces are associated with, or bound to, a VPN by including them in the configuration of the VPN routing instance. By binding the interface to the VPN, the VPN's VRF table is used to make forwarding decisions for any incoming traffic on that interface. Binding the interface also includes the interface local routes in the VRF table, which provides next-hop resolution for VRF routes.

In this example, a firewall filter is used to define which incoming traffic on an interface is forwarded by means of the standard routing table, **inet.0**, and which incoming traffic is forwarded by means of the VRF table. You can expand this example such that incoming traffic on an interface can be redirected to one or more VPNs. For example, you can define a configuration to support a VPN that forwards traffic based on source address, that forwards Hypertext Transfer Protocol (HTTP) traffic, or that forwards only streaming media.

For this configuration to work, the following conditions must be true:

- The interfaces that use filter-based forwarding must not be bound to the VPN.
- Static routing must be used as the means of routing.
- You must define an interface routing table group that is shared among **inet.0** and the VRF tables to provide local routes to the VRF table.

This example consists of two client hosts (Client D and Client E) that are in two different VPNs and that want to send traffic both within the VPN and to the Internet. The paths are defined as follows:

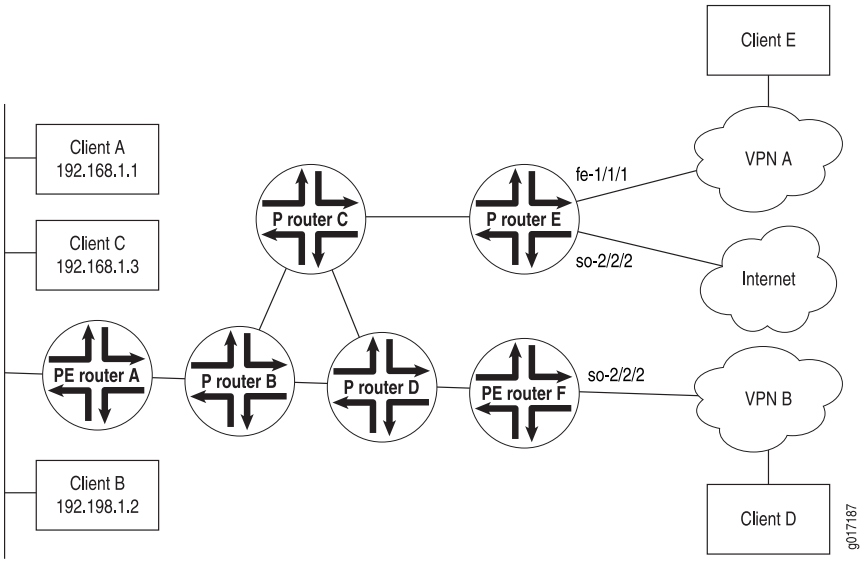
- Client A sends traffic to Client E over VPN A with a return path that also uses VPN A (using the VPN's VRF table).
- Client B sends traffic to Client D over VPN B with a return path that uses standard destination-based routing (using the **inet.0** routing table).
- Clients B and C send traffic to the Internet using standard routing (using the **inet.0** routing table), with a return path that also uses standard routing.

This example illustrates that there are a large variety of options in configuring an application-based Layer 3 VPN topology. This flexibility has application in many network implementations that require specific traffic to be forwarded in a constrained routing environment.

This configuration example shows only the portions of the configuration for the filter-based forwarding, routing instances, and policy. It does not illustrate how to configure a Layer 3 VPN.

Figure 22 on page 146 illustrates the network topology used in this example.

Figure 22: Application-Based Layer 3 VPN Example Configuration



- Configuration on Router A on page 147
- Configuration on Router E on page 148
- Configuration on Router F on page 149

Configuration on Router A

On Router A, you configure the interface to Clients A, B, and C. The configuration evaluates incoming traffic to determine whether it is to be forwarded by means of VPN or standard destination-based routing.

First, you apply an inbound filter and configure the interface:

```
[edit]
interfaces {
    fe-1/1/0 {
        unit 0 {
            family inet {
                filter {
                    input fbf-vrf;
                }
                address 192.168.1.1/24;
            }
        }
    }
}
```

Because the interfaces that use filter-based forwarding must not be bound to a VPN, you must configure an alternate method to provide next-hop routes to the VRF table. You do this by defining an interface routing table group and sharing this group among all the routing tables:

```
[edit]
routing-options {
    interface-routes {
        rib-group inet if-rib;
    }
    rib-groups {
        if-rib {
            import-rib [ inet.0 vpn-A.inet.0 vpn-B.inet.0 ];
        }
    }
}
```

You apply the following filter to incoming traffic on interface **fe-1/1/0.0**. The first term matches traffic from Client A and forwards it to the routing instance for VPN A. The second term matches traffic from Client B that is destined for Client D and forwards it to the routing instance for VPN B. The third term matches all other traffic, which is forwarded normally by means of destination-based forwarding according to the routes in **inet.0**.

```
[edit firewall family family-name]
filter fbf-vrf {
    term vpnA {
        from {
            source-address {
                192.168.1.1/32;
            }
        }
        then {
```

```
        routing-instance vpn-A;
      }
    }
    term vpnB {
      from {
        source-address {
          192.168.1.2/32;
        }
        destination-address {
          192.168.3.0/24;
        }
      }
      then routing-instance vpn-B;
    }
  }
  term internet {
    then accept;
  }
```

You then configure the routing instances for VPN A and VPN B. Notice that these statements include all the required statements to define a Layer 3 VPN except for the **interface** statement.

```
[edit]
routing-instances {
  vpn-A {
    instance-type vrf;
    route-distinguisher 172.21.10.63:100;
    vrf-import vpn-A-import;
    vrf-export vpn-A-export;
  }
  vpn-B {
    instance-type vrf;
    route-distinguisher 172.21.10.63:200;
    vrf-import vpn-B-import;
    vrf-export vpn-B-export;
  }
}
```

Configuration on Router E

On Router E, configure a default route to reach the Internet. You should inject this route into the local IBGP mesh to provide an exit point from the network.

```
[edit]
routing-options {
  static {
    route 0.0.0.0/0 next-hop so-2/2/2.0 discard
  }
}
```

Configure the interface to Client E so that all incoming traffic on interface **fe-1/1/1.0** that matches the VPN policy is forwarded over VPN A:

```
[edit]
routing-instances {
```

```
    vpn-A {
      interface fe-1/1/1.0
      instance-type vrf;
      route-distinguisher 172.21.10.62:100;
      vrf-import vpn-A-import;
      vrf-export vpn-A-export;
      routing-options {
        static {
          route 192.168.2.0/24 next-hop fe-1/1/1.0;
        }
      }
    }
}
```

Configuration on Router F

Again, because the interfaces that use filter-based forwarding must not be bound to a VPN, you configure an alternate method to provide next-hop routes to the VRF table by defining an interface routing table group and sharing this group among all the routing tables. To provide a route back to the clients for normal **inet.0** routing, you define a static route to include in **inet.0** and redistribute the static route into BGP:

```
[edit]
routing-options {
  interface-routes {
    rib-group inet if-rib;
  }
  rib-groups {
    if-rib {
      import-rib [ inet.0 vpn-B.inet.0 ];
    }
  }
}
```

To direct traffic from VPN B to Client D, you configure the routing instance for VPN B on Router F. All incoming traffic from Client D on interface **so-3/3/3.0** is forwarded normally by means of the destination address based on the routes in **inet.0**.

```
[edit]
routing-instances {
  vpn-B {
    instance-type vrf;
    route-distinguisher 172.21.10.64:200;
    vrf-import vpn-B-import;
    vrf-export vpn-B-export;
    routing-options {
      static {
        route 192.168.3.0/24 next-hop so-3/3/3.0;
      }
    }
  }
}
```

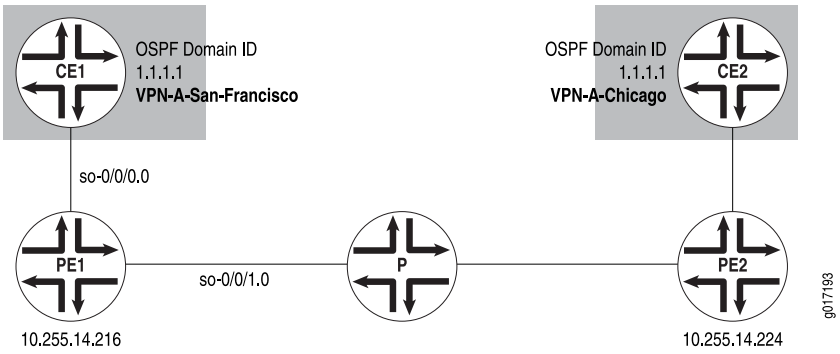
Configuring an OSPF Domain ID for a Layer 3 VPN

This example illustrates how to configure an OSPF domain ID for a VPN by using OSPF as the routing protocol between the PE and CE routers. Routes from an OSPF domain need an OSPF domain ID when they are distributed in BGP as VPN-IPv4 routes in VPNs with multiple OSPF domains. In a VPN connecting multiple OSPF domains, the routes from one domain might overlap with the routes of another.

For more information about OSPF domain IDs and Layer 3 VPNs, see "Configuring an OSPF Domain ID" on page 33.

Figure 23 on page 150 shows this example's configuration topology. Only the configuration for Router PE1 is provided. The configuration for Router PE2 can be similar to the configuration for Router PE1. There are no special configuration requirements for the CE routers.

Figure 23: Example of a Configuration Using an OSPF Domain ID



For configuration information, see the following sections:

- Configuring Interfaces on Router PE1 on page 150
- Configuring Routing Options on Router PE1 on page 151
- Configuring Protocols on Router PE1 on page 151
- Configuring Policy Options on Router PE1 on page 152
- Configuring the Routing Instance on Router PE1 on page 152
- Configuration Summary for Router PE1 on page 153

Configuring Interfaces on Router PE1

You need to configure two interfaces for Router PE1—the **so-0/0/0** interface for traffic to Router CE1 (San Francisco) and the **so-0/0/1** interface for traffic to a P router in the service provider's network.

Configure the interfaces for Router PE1:

```
[edit]
interfaces {
   so-0/0/0 {
      unit 0 {
```

```
        family inet {
          address 10.19.1.2/30;
        }
      }
    }
    so-0/0/1 {
      unit 0 {
        family inet {
          address 10.19.2.1/30;
        }
        family mpls;
      }
    }
}
```

Configuring Routing Options on Router PE1

At the **[edit routing-options]** hierarchy level, you need to configure the **router-id** and **autonomous-system** statements. The **router-id** statement identifies Router PE1.

Configure the routing options for Router PE1:

```
[edit]
routing-options {
  router-id 10.255.14.216;
  autonomous-system 69;
}
```

Configuring Protocols on Router PE1

On Router PE1, you need to configure MPLS, BGP, OSPF, and LDP at the **[edit protocols]** hierarchy level:

```
[edit]
protocols {
  mpls {
    interface so-0/0/1.0;
  }
  bgp {
    group San-Francisco-Chicago {
      type internal;
      preference 10;
      local-address 10.255.14.216;
      family inet-vpn {
        unicast;
      }
      neighbor 10.255.14.224;
    }
  }
  ospf {
    traffic-engineering;
    area 0.0.0.0 {
      interface so-0/0/1.0;
    }
  }
  ldp {
```

```
        interface so-0/0/1.0;
    }
}
```

Configuring Policy Options on Router PE1

On Router PE1, you need to configure policies at the **[edit policy-options]** hierarchy level. These policies ensure that the CE routers in the Layer 3 VPN exchange routing information. In this example, Router CE1 in San Francisco exchanges routing information with Router CE2 in Chicago.

Configure the policy options on the PE1 router:

```
[edit]
policy-options {
    policy-statement vpn-import-VPN-A {
        term term1 {
            from {
                protocol bgp;
                community import-target-VPN-A;
            }
            then accept;
        }
        term term2 {
            then reject;
        }
    }
    policy-statement vpn-export-VPN-A {
        term term1 {
            from protocol ospf;
            then {
                community add export-target-VPN-A;
                accept;
            }
        }
        term term2 {
            then reject;
        }
    }
    community export-target-VPN-A members [target:10.255.14.216:11
    domain-id:1.1.1.1:0];
    community import-target-VPN-A members target:10.255.14.224:31;
}
```

Configuring the Routing Instance on Router PE1

You need to configure a Layer 3 VPN routing instance on Router PE1. To indicate that the routing instance is for a Layer 3 VPN, add the **instance-type vrf** statement at the **[edit routing-instance *routing-instance-name*]** hierarchy level.

The **domain-id** statement is configured at the **[edit routing-instances routing-options protocols ospf]** hierarchy level. As shown in Figure 23 on page 150, the routing instance on Router PE2 must share the same domain ID as the corresponding routing instance on Router PE1 so that routes from Router CE1 to Router CE2 and vice versa are distributed as Type 3 LSAs. If you configure different OSPF domain IDs in the routing instances for

Router PE1 and Router PE2, the routes from each CE router will be distributed as Type 5 LSAs.

Configure the routing instance on Router PE1:

```
[edit]
routing-instances {
   VPN-A-San-Francisco-Chicago {
      instance-type vrf;
      interface so-0/0/0.0;
      route-distinguisher 10.255.14.216:11;
      vrf-import vpn-import-VPN-A;
      vrf-export vpn-export-VPN-A;
      routing-options {
         router-id 10.255.14.216;
         autonomous-system 69;
      }
      protocols {
         ospf {
            domain-id 1.1.1.1;
            export vpn-import-VPN-A;
            area 0.0.0.0 {
               interface so-0/0/0.0;
            }
         }
      }
   }
}
```

Configuration Summary for Router PE1

Configure Interfaces

```
interfaces {
   so-0/0/0 {
      unit 0 {
         family inet {
            address 10.19.1.2/30;
         }
      }
   }
   so-0/0/1 {
      unit 0 {
         family inet {
            address 10.19.2.1/30;
         }
         family mpls;
      }
   }
}
```

Configure Routing Options

```
routing-options {
   router-id 10.255.14.216;
   autonomous-system 69;
}
```

Configure Protocols

```
protocols {
   mpls {
```

```
    interface so-0/0/0.0;
  }
  bgp {
    group San-Francisco-Chicago {
      type internal;
      preference 10;
      local-address 10.255.14.216;
      family inet-vpn {
        unicast;
      }
      neighbor 10.255.14.224;
    }
  }
  ospf {
    traffic-engineering;
    area 0.0.0.0 {
      interface so-0/0/1.0;
    }
  }
  ldp {
    interface so-0/0/1.0;
  }
}
```

Configure VPN Policy

```
policy-options {
  policy-statement vpn-import-VPN-A {
    term term1 {
      from {
        protocol bgp;
        community import-target-VPN-A;
      }
      then accept;
    }
    term term2 {
      then reject;
    }
  }
  policy-statement vpn-export-VPN-A {
    term term1 {
      from protocol ospf;
      then {
        community add export-target-VPN-A;
        accept;
      }
    }
    term term2 {
      then reject;
    }
  }
  community export-target-VPN-B members [ target:10.255.14.216:11domain-id:1.1.1.1:0 ];
  community import-target-VPN-B members target:10.255.14.224:31;
}
```

Routing Instance for Layer 3 VPN

```
routing-instances {
  VPN-A-San-Francisco-Chicago {
```

```
        instance-type vrf;
        interface so-0/0/0.0;
        route-distinguisher 10.255.14.216:11;
        vrf-import vpn-import-VPN-A;
        vrf-export vpn-export-VPN-A;
        routing-options {
          router-id 10.255.14.216;
          autonomous-system 69;
        }
        protocols {
          ospf {
            domain-id 1.1.1.1;
            export vpn-import-VPN-A;
            area 0.0.0.0 {
              interface so-0/0/0.0;
            }
          }
        }
      }
    }
```

Configuring Overlapping VPNs Using Routing Table Groups

In Layer 3 VPNs, a CE router is often a member of more than one VPN. This example illustrates how to configure PE routers that support CE routers that support multiple VPNs. Support for this type of configuration uses a Junos OS feature called routing table groups (sometimes also called routing information base [RIB] groups), which allows a route to be installed into several routing tables. A routing table group is a list of routing tables into which the protocol should install its routes.

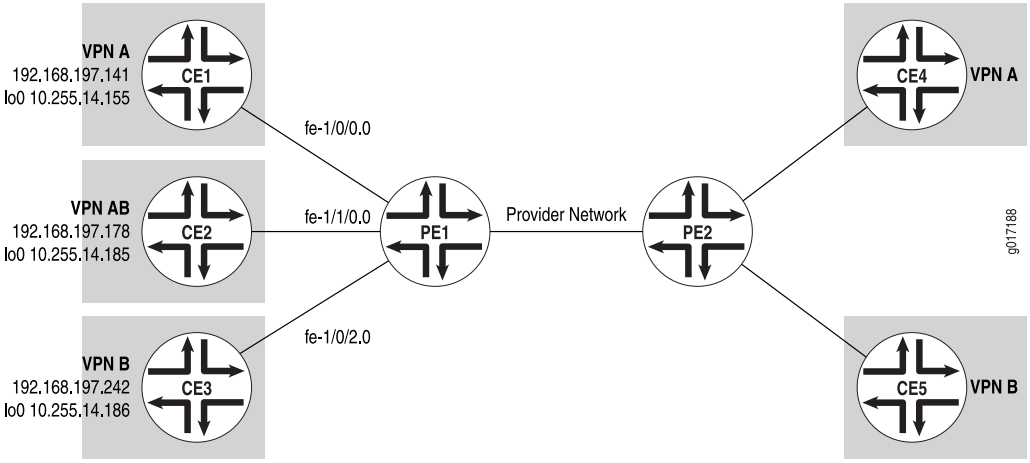
You define routing table groups at the **[edit routing-options]** hierarchy level for the default instance. You cannot configure routing table groups at the **[edit routing-instances routing-options]** hierarchy level; doing so results in a commit error.

After you define a routing table group, it can be used by multiple protocols. You can also apply routing table groups to static routing. The configuration examples in this section include both types of configurations.

Figure 24 on page 156 illustrates the topology for the configuration example in this section. The configurations in this section illustrate local connectivity between CE routers connected to the same PE router. If Router PE1 were connected only to Router CE2 (VPN AB), there would be no need for any extra configuration. The configuration statements in the sections that follow enable VPN AB Router CE2 to communicate with VPN A Router CE1 and VPN B Router CE3, which are directly connected to Router PE1. VPN routes that originate from the remote PE routers (the PE2 router in this case) are placed in a global Layer 3 VPN routing table (**bgp.l3vpn.inet.0**), and routes with appropriate route targets are imported into the routing tables as dictated by the VRF import policy configuration. The goal is to be able to choose routes from individual VPN routing tables that are locally populated.

Router PE1 is where all the filtering and configuration modification takes place. Therefore only VPN configurations for PE1 are shown. The CE routers do not have any information about the VPN, so you can configure them normally.

Figure 24: Example of an Overlapping VPN Topology



The following sections explain several ways to configure overlapping VPNs.

The following sections illustrate different scenarios for configuring overlapping VPNs, depending on the routing protocol used between the PE and CE routers. For all of these examples, you need to configure routing table groups.

- Configuring Routing Table Groups on page 156
- Configuring Static Routes Between the PE and CE Routers on page 157
- Configuring BGP Between the PE and CE Routers on page 162
- Configuring OSPF Between the PE and CE Routers on page 163
- Configuring Static, BGP, and OSPF Routes Between PE and CE Routers on page 165

Configuring Routing Table Groups

In this example, routing table groups are common in the four configuration scenarios. The routing table groups are used to install routes (including interface, static, OSPF, and BGP routes) into several routing tables for the default and other instances. In the routing table group definition, the first routing table is called the primary routing table. (Normally, the primary routing table is the table into which the route would be installed if you did not configure routing table groups. The other routing tables are called secondary routing tables.)

The routing table groups in this configuration install routes as follows:

- **vpna-vpnab** installs routes into routing tables **VPN-A.inet.0** and **VPN-AB.inet.0**.
- **vpnb-vpnab** installs routes into routing tables **VPN-B.inet.0** and **VPN-AB.inet.0**.
- **vpnab-vpna_and_vpnb** installs routes into routing tables **VPN-AB.inet.0**, **VPN-A.inet.0**, and **VPN-B.inet.0**.

Configure the routing table groups:

```
[edit]
routing-options {
   rib-groups {
      vpna-vpnab {
         import-rib [ VPN-A.inet.0 VPN-AB.inet.0 ];
      }
      vpnb-vpnab {
         import-rib [ VPN-B.inet.0 VPN-AB.inet.0 ];
      }
      vpnab-vpna_and_vpnb {
         import-rib [ VPN-AB.inet.0 VPN-A.inet.0 VPN-B.inet.0 ];
      }
   }
}
```

Configuring Static Routes Between the PE and CE Routers

To configure static routing between the PE1 router and the CE1, CE2, and CE3 routers, you must configure routing instances for VPN A, VPN B, and VPN AB (you configure static routing under each instance):

- Configuring the Routing Instance for VPN A on page 157
- Configuring the Routing Instance for VPN AB on page 158
- Configuring the Routing Instance for VPN B on page 158
- Configuring VPN Policy on page 159

Configuring the Routing Instance for VPN A

On Router PE1, configure VPN A:

```
[edit]
routing-instances {
   VPN-A {
      instance-type vrf;
      interface fe-1/0/0.0;
      route-distinguisher 10.255.14.175:3;
      vrf-import vpna-import;
      vrf-export vpna-export;
      routing-options {
         interface-routes {
            rib-group inet vpna-vpnab;
         }
         static {
            route 10.255.14.155/32 next-hop 192.168.197.141;
            route 10.255.14.185/32 next-hop 192.168.197.178;
         }
      }
   }
}
```

The **interface-routes** statement installs VPN A's interface routes into the routing tables defined in the routing table group **vpna-vpnab**.

The **static** statement configures the static routes that are installed in the **VPN-A.inet.0** routing table. The first static route is for Router CE1 (VPN A) and the second is for Router CE2 (in VPN AB).

Next hop **192.168.197.178** is not in VPN A. Route **10.255.14.185/32** cannot be installed in **VPN-A.inet.0** unless interface routes from routing instance VPN AB are installed in this routing table. Including the **interface-routes** statements in the VPN AB configuration provides this next hop. Similarly, including the **interface-routes** statement in the VPN AB configuration installs **192.168.197.141** into **VPN-AB.inet.0**.

Configuring the Routing Instance for VPN AB

On Router PE1, configure VPN AB:

```
[edit]
routing instances {
   VPN-AB {
      instance-type vrf;
      interface fe-1/1/0.0;
      route-distinguisher 10.255.14.175:9;
      vrf-import vpnab-import;
      vrf-export vpnab-export;
      routing-options {
         interface-routes {
            rib-group vpnab-vpna_and_vpnb;
         }
         static {
            route 10.255.14.185/32 next-hop 192.168.197.178;
            route 10.255.14.155/32 next-hop 192.168.197.141;
            route 10.255.14.186/32 next-hop 192.168.197.242;
         }
      }
   }
}
```

In this configuration, the following static routes are installed in the **VPN-AB.inet.0** routing table:

- **10.255.14.185/32** is for Router CE2 (in VPN AB)
- **10.255.14.155/32** is for Router CE1 (in VPN A)
- **10.255.14.186/32** is for Router CE3 (in VPN B)

Next hops **192.168.197.141** and **192.168.197.242** do not belong to VPN AB. Routes **10.255.14.155/32** and **10.255.14.186/32** cannot be installed in **VPN-AB.inet.0** unless interface routes from VPN A and VPN B are installed in this routing table. The interface route configurations in VPN A and VPN B routing instances provide these next hops.

Configuring the Routing Instance for VPN B

On Router PE1, configure VPN B:

```
[edit]
routing instances {
   VPN-B {
```

```
        instance-type vrf;
        interface fe-1/0/2.0;
        route-distinguisher 10.255.14.175:10;
        vrf-import vpnb-import;
        vrf-export vpnb-export;
        routing-options {
          interface-routes {
            rib-group inet vpnb-vpnab;
          }
          static {
            route 10.255.14.186/32 next-hop 192.168.197.242;
            route 10.255.14.185/32 next-hop 192.168.197.178;
          }
        }
      }
    }
```

When you configure the routing instance for VPN B, these static routes are placed in **VPNB.inet.0**:

- **10.255.14.186/32** is for Router CE3 (in VPN B)
- **10.255.14.185/32** is for Router CE2 (in VPN AB)

Next hop **192.168.197.178** does not belong to VPN B. Route **10.255.14.185/32** cannot be installed in **VPN-B.inet.0** unless interface routes from VPN AB are installed in this routing table. The interface route configuration in VPN AB provides this next hop.

Configuring VPN Policy

The **vrf-import** and **vrf-export** policy statements that you configure for overlapping VPNs are the same as policy statements for regular VPNs, except that you include the **from interface** statement in each VRF export policy. This statement forces each VPN to announce only those routes that originated from that VPN. For example, VPN A has routes that originated in VPN A and VPN AB. If you do not include the **from interface** statement, VPN A announces its own routes as well as VPN AB's routes, so the remote PE router receives multiple announcements for the same routes. Including the **from interface** statement restricts each VPN to announcing only the routes it originated and allows you to filter out the routes imported from other routing tables for local connectivity.

In this configuration example, the **vpnab-import** policy accepts routes from VPN A, VPN B, and VPN AB. The **vpna-export** policy exports only routes that originate in VPN A. Similarly, the **vpnb-export** and **vpnab-export** policies export only routes that originate within the respective VPNs.

On Router PE1, configure the following VPN import and export policies:

```
[edit]
policy-options {
  policy-statement vpna-import {
    term a {
      from {
        protocol bgp;
        community VPNA-comm;
      }
```

```
        then accept;
    }
    term b {
        then reject;
    }
}
policy-statement vpnb-import {
    term a {
        from {
            protocol bgp;
            community VPNB-comm;
        }
        then accept;
    }
    term b {
        then reject;
    }
}
policy-statement vpnab-import {
    term a {
        from {
            protocol bgp;
            community [ VPNA-comm VPNB-comm ];
        }
        then accept;
    }
    term b {
        then reject;
    }
}
policy-statement vpna-export {
    term a {
        from {
            protocol static;
            interface fe-1/0/0.0;
        }
        then {
            community add VPNA-comm;
            accept;
        }
    }
    term b {
        then reject;
    }
}
policy-statement vpnb-export {
    term a {
        from {
            protocol static;
            interface fe-1/0/2.0;
        }
        then {
            community add VPNB-comm;
            accept;
        }
    }
```

```
        term b {
            then reject;
        }
    }
    policy-statement vpnab-export {
        term a {
            from {
                protocol static;
                interface fe-1/1/0.0;
            }
            then {
                community add VPNB-comm;
                community add VPNA-comm;
                accept;
            }
        }
        term b {
            then reject;
        }
    }
    community VPNA-comm members target:69:1;
    community VPNB-comm members target:69:2;
}
```

On Router PE1, apply the VPN import and export policies:

```
[edit]
routing-instances {
    VPN-A {
        instance-type vrf;
        interface fe-1/0/0.0;
        route-distinguisher 10.255.14.175:3;
        vrf-import vpna-import;
        vrf-export vpna-export;
        routing-options {
            static {
                rib-group vpna-vpnab;
                route 10.255.14.155/32 next-hop 192.168.197.141;
                route 10.255.14.185/32 next-hop 192.168.197.178;
            }
        }
    }
    VPN-AB {
        instance-type vrf;
        interface fe-1/1/0.0;
        route-distinguisher 10.255.14.175:9;
        vrf-import vpnab-import;
        vrf-export vpnab-export;
        routing-options {
            static {
                rib-group vpnab-vpna_and_vpnb;
                route 10.255.14.185/32 next-hop 192.168.197.178;
            }
        }
    }
    VPN-B {
```

```
        instance-type vrf;
        interface fe-1/0/2.0;
        route-distinguisher 10.255.14.175:10;
        vrf-import vpnb-import;
        vrf-export vpnb-export;
        routing-options {
          static {
            rib-group vpnb-vpnab;
            route 10.255.14.186/32 next-hop 192.168.197.242;
          }
        }
      }
    }
```

For VPN A, include the **routing-options** statement at the **[edit routing-instances *routing-instance-name*]** hierarchy level to install the static routes directly into the routing tables defined in the routing table group **vpna-vpnab**. For VPN AB, the configuration installs the static route directly into the routing tables defined in the routing table group **vpnab-vpna** and **vpnab-vpnb**. For VPN B the configuration installs the static route directly into the routing tables defined in the routing table group **vpnb-vpnab**.

Configuring BGP Between the PE and CE Routers

In this configuration example, the **vpna-site1** BGP group for VPN A installs the routes learned from the BGP session into the routing tables defined in the **vpna-vpnab** routing table group. For VPN AB, the **vpnab-site1** group installs the routes learned from the BGP session into the routing tables defined in the **vpnab-vpna_and_vpnb** routing table group. For VPN B, the **vpnb-site1** group installs the routes learned from the BGP session into the routing tables defined in the **vpnb-vpnab** routing table group. Interface routes are not needed for this configuration.

The VRF import and export policies are similar to those defined in "Configuring Static Routes Between the PE and CE Routers" on page 157, except the export protocol is BGP instead of a static route. On all **vrf-export** policies, you use the **from protocol bgp** statement.

On Router PE1, configure BGP between the PE and CE routers:

```
[edit]
routing-instances {
  VPN-A {
    instance-type vrf;
    interface fe-1/0/0.0;
    route-distinguisher 10.255.14.175:3;
    vrf-import vpna-import;
    vrf-export vpna-export;
    protocols {
      bgp {
        group vpna-site1 {
          family inet {
            unicast {
              rib-group vpna-vpnab;
            }
          }
```

```
            peer-as 1;
            neighbor 192.168.197.141;
          }
        }
      }
    }
    VPN-AB {
      instance-type vrf;
      interface fe-1/1/0.0;
      route-distinguisher 10.255.14.175:9;
      vrf-import vpnab-import;
      vrf-export vpnab-export;
      protocols {
        bgp {
          group vpnab-site1 {
            family inet {
              unicast {
                rib-group vpnab-vpna_and_vpnb;
              }
            }
            peer-as 9;
            neighbor 192.168.197.178;
          }
        }
      }
    }
    VPN-B {
      instance-type vrf;
      interface fe-1/0/2.0;
      route-distinguisher 10.255.14.175:10;
      vrf-import vpnb-import;
      vrf-export vpnb-export;
      protocols {
        bgp {
          group vpnb-site1 {
            family inet {
              unicast {
                rib-group vpnb-vpnab;
              }
            }
            neighbor 192.168.197.242 {
              peer-as 10;
            }
          }
        }
      }
    }
  }
```

Configuring OSPF Between the PE and CE Routers

In this configuration example, routes learned from the OSPF session for VPN A are installed into the routing tables defined in the **vpna-vpnab** routing table group. For VPN AB, routes learned from the OSPF session are installed into the routing tables defined in the

vpnab-vpna_and_vpnb routing table group. For VPN B, routes learned from the OSPF session are installed into the routing tables defined in the **vpnb-vpnab** routing table group.

The VRF import and export policies are similar to those defined in "Configuring Static Routes Between the PE and CE Routers" on page 157 and "Configuring BGP Between the PE and CE Routers" on page 162, except the export protocol is OSPF instead of BGP or a static route. Therefore, on all **vrf-export** policies, you use the **from protocol ospf** statement instead of the **from protocol <static | bgp>** statement.

On Router PE1, configure OSPF between the PE and CE routers:

```
[edit]
routing-instances {
    VPN-A {
        instance-type vrf;
        interface fe-1/0/0.0;
        route-distinguisher 10.255.14.175:3;
        vrf-import vpna-import;
        vrf-export vpna-export;
        protocols {
            ospf {
                rib-group vpna-vpnab;
                export vpna-import;
                area 0.0.0.0 {
                    interface fe-1/0/0.0;
                }
            }
        }
    }
    VPN-AB {
        instance-type vrf;
        interface fe-1/1/0.0;
        route-distinguisher 10.255.14.175:9;
        vrf-import vpnab-import;
        vrf-export vpnab-export;
        protocols {
            ospf {
                rib-group vpnab-vpna_and_vpnb;
                export vpnab-import;
                area 0.0.0.0 {
                    interface fe-1/1/0.0;
                }
            }
        }
    }
    VPN-B {
        instance-type vrf;
        interface fe-1/0/2.0;
        route-distinguisher 10.255.14.175:10;
        vrf-import vpnb-import;
        vrf-export vpnb-export;
        protocols {
            ospf {
                rib-group vpnb-vpnab;
                export vpnb-import;
```

```
            area 0.0.0.0 {
              interface fe-1/0/2.0;
            }
          }
        }
      }
    }
```

Configuring Static, BGP, and OSPF Routes Between PE and CE Routers

This section shows how to configure the routes between the PE and CE routers by using a combination of static routes, BGP, and OSPF:

- The connection between Router PE1 and Router CE1 uses static routing.
- The connection between Router PE1 and Router CE2 uses BGP.
- The connection between Router PE1 and Router CE3 uses OSPF.

Here, the configuration for VPN AB also includes a static route to CE1.

On Router PE1, configure a combination of static routing, BGP, and OSPF between the PE and CE routers:

```
[edit]
routing-instances {
  VPN-A {
    instance-type vrf;
    interface fe-1/0/0.0;
    route-distinguisher 10.255.14.175:3;
    vrf-import vpna-import;
    vrf-export vpna-export;
    routing-options {
      static {
        rib-group vpna-vpnab;
        route 10.255.14.155/32 next-hop 192.168.197.141;
      }
    }
  }
  VPN-AB {
    instance-type vrf;
    interface fe-1/1/0.0;
    route-distinguisher 10.255.14.175:9;
    vrf-import vpnab-import;
    vrf-export vpnab-export;
    protocols {
      bgp {
        group vpnab-site1 {
          family inet {
            unicast {
              rib-group vpnab-vpna_and_vpnb;
            }
          }
          export to-vpnab-site1;
          peer-as 9;
          neighbor 192.168.197.178;
```

```
            }
          }
        }
      }
      VPN-B {
        instance-type vrf;
        interface fe-1/0/2.0;
        route-distinguisher 10.255.14.175:10;
        vrf-import vpnb-import;
        vrf-export vpnb-export;
        protocols {
          ospf {
            rib-group vpnb-vpnab;
            export vpnb-import;
            area 0.0.0.1 {
              interface t3-0/3/3.0;
            }
          }
        }
      }
    }
    policy-options {
      policy-statement to-vpnab-site1 {
        term a {
          from protocol static;
          then accept;
        }
        term b {
          from protocol bgp;
          then accept;
        }
        term c {
          then reject;
        }
      }
    }
```

Configuring Overlapping VPNs Using Automatic Route Export

A problem with multiple routing instances is how to export routes between routing instances. You can accomplish this in Junos OS by configuring routing table groups for each routing instance that needs to export routes to other routing tables. For information about how to configure overlapping VPNs by using routing table groups, see "Configuring Overlapping VPNs Using Routing Table Groups" on page 155.

However, using routing table groups has limitations:

- Routing table group configuration is complex. You must define a unique routing table group for each routing instance that will export routes.
- You must also configure a unique routing table group for each protocol that will export routes.

To limit and sometimes eliminate the need to configure routing table groups in multiple routing instance topologies, you can use the functionality provided by the **auto-export** statement.

The **auto-export** statement is particularly useful for configuring overlapping VPNs—VPN configurations where more than one VRF routing instance lists the same community route target in its **vrf-import** policy. The **auto-export** statement finds out which routing tables to export routes from and import routes to by examining the existing policy configuration.

The **auto-export** statement automatically exports routes between the routing instances referencing a given route target community. When the **auto-export** statement is configured, a VRF target tree is constructed based on the **vrf-import** and **vrf-export** policies configured on the system. If a routing instance references a route target in its **vrf-import** policy, the route target is added to the import list for the target. If it references a specific route target in its **vrf-export** policy, the route target is added to the export list for that target. Route targets where there is a single importer that matches a single exporter or with no importers or exporters are ignored.

Changes to routing tables that export route targets are tracked. When a route change occurs, the routing instance's **vpn-export** policy is applied to the route. If it is allowed, the route is imported to all the import tables (subject to the **vrf-import** policy) of the route targets set by the export policy.

The sections that follow describe how to configure overlapping VPNs by using the **auto-export** statement for inter-instance export in addition to routing table groups:

- Configuring Overlapping VPNs with BGP and Automatic Route Export on page 167
- Configuring Overlapping VPNs and Additional Tables on page 168
- Configuring Automatic Route Export for All VRF Instances on page 169

Configuring Overlapping VPNs with BGP and Automatic Route Export

The following example provides the configuration for an overlapping VPN where BGP is used between the PE and CE routers.

Configure routing instance **VPN-A**:

```
[edit]
routing-instances {
   VPN-A {
      instance-type vrf;
      interface fe-1/0/0.0;
      route-distinguisher 10.255.14.175:3;
      vrf-import vpna-import;
      vrf-export vpna-export;
      routing-options {
         auto-export;
      }
      protocols {
         bgp {
            group vpna-site1 {
```

```
            peer-as 1;
            neighbor 192.168.197.141;
          }
        }
      }
    }
  }
```

Configure routing instance **VPN-AB**:

```
[edit]
routing-instances {
  VPN-AB {
    instance-type vrf;
    interface fe-1/1/0.0;
    route-distinguisher 10.255.14.175:9;
    vrf-import vpnab-import;
    vrf-export vpnab-export;
    routing-options {
      auto-export;
    }
    protocols {
      bgp {
        group vpnab-site1 {
          peer-as 9;
          neighbor 192.168.197.178;
        }
      }
    }
  }
}
```

For this configuration, the **auto-export** statement replaces the functionality that was provided by a routing table group configuration. However, sometimes additional configuration is required.

Since the **vrf-import** policy and the **vrf-export** policy from which the **auto-export** statement deduces the import and export matrix are configured on a per-instance basis, you must be able to enable or disable them for unicast and multicast, in case multicast network layer reachability information (NLRI) is configured.

Configuring Overlapping VPNs and Additional Tables

You might need to use the **auto-export** statement between overlapping VPNs but require that a subset of the routes learned from a VRF table be installed into the **inet.0** table or in **routing-instance.inet.2**.

To support this type of scenario, where not all of the information needed is present in the **vrf-import** and **vrf-export** policies, you configure an additional list of routing tables by using an additional routing table group.

To add routes from **VPN-A** and **VPN-AB** to **inet.0** in the example described, you need to include the following additional configuration statements:

Configure the routing options:

```
[edit]
routing-options {
  rib-groups {
    inet-access {
      import-rib inet.0;
    }
  }
}
```

Configure routing instance **VPN-A**:

```
[edit]
routing-instances {
  VPN-A {
    routing-options {
      auto-export {
        family inet {
          unicast {
            rib-group inet-access;
          }
        }
      }
    }
  }
}
```

Configure routing instance **VPN-AB**:

```
[edit]
routing-instances {
  VPN-AB {
    routing-options {
      auto-export {
        family inet {
          unicast {
            rib-group inet-access;
          }
        }
      }
    }
  }
}
```

Routing table groups are used in this configuration differently from how they are generally used in Junos OS. Routing table groups normally require that the exporting routing table be referenced as the primary import routing table in the routing table group. For this configuration, the restriction does not apply. The routing table group functions as an additional list of tables to which to export routes.

Configuring Automatic Route Export for All VRF Instances

The following configuration allows you to configure the **auto-export** statement for all of the routing instances in a configuration group:

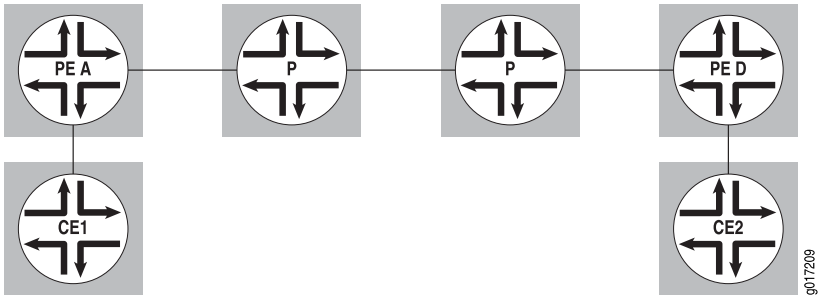
```
[edit]
groups {
  vrf-export-on {
```

```
        routing-instances {
          <*> {
            routing-options {
              auto-export;
            }
          }
        }
      }
    }
    apply-groups vrf-export-on;
```

Configuring a GRE Tunnel Interface Between PE Routers

This example shows how to configure a generic routing encapsulation (GRE) tunnel interface between PE routers to provide VPN connectivity. You can use this configuration to tunnel VPN traffic across a non-MPLS core network. The network topology used in this example is shown in Figure 25 on page 170. The P routers shown in this illustration do not run MPLS.

Figure 25: PE Routers A and D Connected by a GRE Tunnel Interface



For configuration information, see the following sections:

- Configuring the Routing Instance on Router A on page 170
- Configuring the Routing Instance on Router D on page 171
- Configuring MPLS, BGP, and OSPF on Router A on page 171
- Configuring MPLS, BGP, and OSPF on Router D on page 172
- Configuring the Tunnel Interface on Router A on page 172
- Configuring the Tunnel Interface on Router D on page 173
- Configuring the Routing Options on Router A on page 173
- Configuring the Routing Options on Router D on page 173
- Configuration Summary for Router A on page 174
- Configuration Summary for Router D on page 175

Configuring the Routing Instance on Router A

Configure a routing instance on Router A:

```
[edit routing-instances]
gre-config {
```

```
    instance-type vrf;
    interface fe-1/0/0.0;
    route-distinguisher 10.255.14.176:69;
    vrf-import import-config;
    vrf-export export-config;
    protocols {
      ospf {
        export import-config;
        area 0.0.0.0 {
          interface all;
        }
      }
    }
}
```

Configuring the Routing Instance on Router D

Configure a routing instance on Router D:

```
[edit routing-instances]
gre-config {
    instance-type vrf;
    interface fe-1/0/1.0;
    route-distinguisher 10.255.14.178:69;
    vrf-import import-config;
    vrf-export export-config;
    protocols {
      ospf {
        export import-config;
        area 0.0.0.0 {
          interface all;
        }
      }
    }
}
```

Configuring MPLS, BGP, and OSPF on Router A

Although you do not need to configure MPLS on the P routers in this example, it is needed on the PE routers for the interface between the PE and CE routers and on the GRE interface (**gr-1/1/0.0**) linking the PE routers (Router A and Router D). Configure MPLS, BGP, and OSPF on Router A:

```
[edit protocols]
mpls {
    interface all;
}
bgp {
    group pe-to-pe {
      type internal;
      neighbor 10.255.14.178 {
        family inet-vpn {
          unicast;
        }
      }
    }
```

```
}
ospf {
  area 0.0.0.0 {
    interface all;
    interface gr-1/1/0.0 {
      disable;
    }
  }
}
```

Configuring MPLS, BGP, and OSPF on Router D

Although you do not need to configure MPLS on the P routers in this example, it is needed on the PE routers for the interface between the PE and CE routers and on the GRE interface (**gr-1/1/0.0**) linking the PE routers (Router D and Router A). Configure MPLS, BGP, and OSPF on Router D:

```
[edit protocols]
mpls {
  interface all;
}
bgp {
  group pe-to-pe {
    type internal;
    neighbor 10.255.14.176 {
      family inet-vpn {
        unicast;
      }
    }
  }
}
ospf {
  traffic-engineering;
  area 0.0.0.0 {
    interface all;
    interface fxp0.0 {
      disable;
    }
    interface gr-1/1/0.0 {
      disable;
    }
  }
}
```

Configuring the Tunnel Interface on Router A

Configure the tunnel interface on Router A (the tunnel is unnumbered):

```
[edit interfaces interface-name]
unit 0 {
  tunnel {
    source 10.255.14.176;
    destination 10.255.14.178;
  }
  family inet;
  family mpls;
```

```
}
```

Configuring the Tunnel Interface on Router D

Configure the tunnel interface on Router D (the tunnel is unnumbered):

```
[edit interfaces interface-name]
unit 0 {
   tunnel {
      source 10.255.14.178;
      destination 10.255.14.176;
   }
   family inet;
   family mpls;
}
```

Configuring the Routing Options on Router A

As part of the routing options configuration for Router A, you need to configure routing table groups to enable VPN route resolution in the **inet.3** routing table.

Configure the routing options on Router A:

```
[edit routing-options]
interface-routes {
   rib-group inet if-rib;
}
rib inet.3 {
   static {
      route 10.255.14.178/32 next-hop gr-1/1/0.0;
   }
}
rib-groups {
   if-rib {
      import-rib [ inet.0 inet.3 ];
   }
}
```

Configuring the Routing Options on Router D

As part of the routing options configuration for Router D, you need to configure routing table groups to enable VPN route resolution in the **inet.3** routing table.

Configure the routing options on Router D:

```
[edit routing-options]
interface-routes {
   rib-group inet if-rib;
}
rib inet.3 {
   static {
      route 10.255.14.176/32 next-hop gr-1/1/0.0;
   }
}
rib-groups {
   if-rib {
```

```
    import-rib [ inet.0 inet.3 ];
  }
}
```

Configuration Summary for Router A

Configure the Routing Instance

```
gre-config {
  instance-type vrf;
  interface fe-1/0/0.0;
  route-distinguisher 10.255.14.176:69;
  vrf-import import-config;
  vrf-export export-config;
  protocols {
    ospf {
      export import-config;
      area 0.0.0.0 {
        interface all;
      }
    }
  }
}
```

Configure MPLS

```
mpls {
  interface all;
}
```

Configure BGP

```
bgp {
  traceoptions {
    file bgp.trace world-readable;
    flag update detail;
  }
  group pe-to-pe {
    type internal;
    neighbor 10.255.14.178 {
      family inet-vpn {
        unicast;
      }
    }
  }
}
```

Configure OSPF

```
ospf {
  area 0.0.0.0 {
    interface all;
    interface gr-1/1/0.0 {
      disable;
    }
  }
}
```

Configure the Tunnel Interface

```
interface-name {
  unit 0 {
    tunnel {
      source 10.255.14.176;
      destination 10.255.14.178;
```

```
    }
    family inet;
    family mpls;
  }
}
```

Configure Routing Options

```
interface-routes {
  rib-group inet if-rib;
}
rib inet.3 {
  static {
    route 10.255.14.178/32 next-hop gr-1/1/0.0;
  }
}
rib-groups {
  if-rib {
    import-rib [ inet.0 inet.3 ];
  }
}
```

Configuration Summary for Router D

Configure the Routing Instance

```
gre-config {
  instance-type vrf;
  interface fe-1/0/1.0;
  route-distinguisher 10.255.14.178:69;
  vrf-import import-config;
  vrf-export export-config;
  protocols {
    ospf {
      export import-config;
      area 0.0.0.0 {
        interface all;
      }
    }
  }
}
```

Configure MPLS

```
mpls {
  interface all;
}
```

Configure BGP

```
bgp {
  group pe-to-pe {
    type internal;
    neighbor 10.255.14.176 {
      family inet-vpn {
        unicast;
      }
    }
  }
}
```

Configure OSPF

```
ospf {
  traffic-engineering;
```

```
    area 0.0.0.0 {
      interface all;
      interface fxp0.0 {
        disable;
      }
      interface gr-1/1/0.0 {
        disable;
      }
    }
}
```

Configure the Tunnel Interface

```
interface-name {
    unit 0 {
        tunnel {
            source 10.255.14.178;
            destination 10.255.14.176;
        }
        family inet;
        family mpls;
    }
}
```

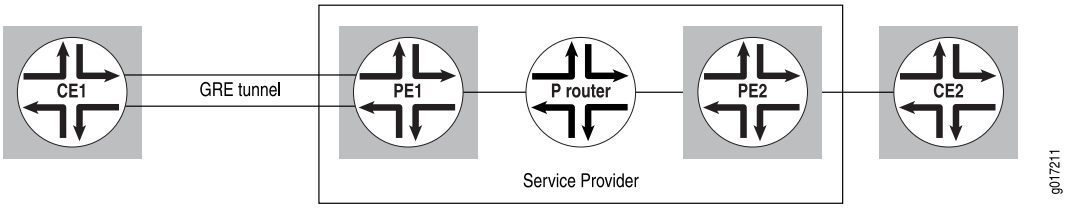
Configure the Routing Options

```
interface-routes {
    rib-group inet if-rib;
}
rib inet.3 {
    static {
        route 10.255.14.176/32 next-hop gr-1/1/0.0;
    }
}
rib-groups {
    if-rib {
        import-rib [ inet.0 inet.3 ];
    }
}
```

Configuring a GRE Tunnel Interface Between a PE and CE Router

This example shows how to configure a GRE tunnel interface between a PE router and a CE router. You can use this configuration to tunnel VPN traffic across a non-MPLS core network. The network topology used in this example is shown in Figure 26 on page 176.

Figure 26: GRE Tunnel Between the CE Router and the PE Router



For this example, complete the procedures described in the following sections:

- Configuring the Routing Instance Without the Encapsulating Interface on page 177
- Configuring the Routing Instance with the Encapsulating Interface on page 178
- Configuring the GRE Tunnel Interface on Router CE1 on page 179

Configuring the Routing Instance Without the Encapsulating Interface

You can configure the routing instance either with or without the encapsulating interface. The following sections explain how to configure the routing instance without it:

- Configuring the Routing Instance on Router PE1 on page 177
- Configuring the GRE Tunnel Interface on Router PE1 on page 177
- Configuring the Encapsulation Interface on Router PE1 on page 178

Configuring the Routing Instance on Router PE1

Configure the routing instance on Router PE1:

```
[edit routing-instances]
vpna {
   instance-type vrf;
   interface gr-1/2/0.0;
   route-distinguisher 10.255.14.174:1;
   vrf-import vpna-import;
   vrf-export vpna-export;
   protocols {
      bgp {
         group vpna {
            type external;
            peer-as 100;
            as-override;
            neighbor 10.49.2.1;
         }
      }
   }
}
```

Configuring the GRE Tunnel Interface on Router PE1

Configure the GRE tunnel interface on Router PE1:

```
[edit interfaces gr-1/2/0]
unit 0 {
   tunnel {
      source 192.168.197.249;
      destination 192.168.197.250;
   }
   family inet {
      address 10.49.2.2/30;
   }
}
```

In this example, interface **t3-0/1/3** acts as the encapsulating interface for the GRE tunnel.

Configuring the Encapsulation Interface on Router PE1

Configure the encapsulation interface on Router PE1:

```
[edit interfaces t3-0/1/3]
unit 0 {
   family inet {
      address 192.168.197.249/30;
   }
}
```

Configuring the Routing Instance with the Encapsulating Interface

If the tunnel-encapsulating interface, **t3-0/1/3**, is also configured under the routing instance, then you need to specify the name of that routing instance under the interface definition. The system uses this routing instance to search for the tunnel destination address.

To configure the routing instance with the encapsulating interface, you perform the steps in the following sections:

- Configuring the Routing Instance on Router PE1 on page 178
- Configuring the GRE Tunnel Interface on Router PE1 on page 178
- Configuring the Encapsulation Interface on Router PE1 on page 179

Configuring the Routing Instance on Router PE1

If you configure the tunnel-encapsulating interface under the routing instance, then configure the routing instance on Router PE1:

```
[edit routing-instances]
vpna {
   instance-type vrf;
   interface gr-1/2/0.0;
   interface t3-0/1/3.0;
   route-distinguisher 10.255.14.174:1;
   vrf-import vpna-import;
   vrf-export vpna-export;
   protocols {
      bgp {
         group vpna {
            type external;
            peer-as 100;
            as-override;
            neighbor 10.49.2.1;
         }
      }
   }
}
```

Configuring the GRE Tunnel Interface on Router PE1

Configure the GRE tunnel interface on Router PE1:

```
[edit interfaces gr-1/2/0]
unit 0 {
   tunnel {
      source 192.168.197.249;
      destination 192.168.197.250;
      routing-instance {
         destination vpna;
      }
   }
   family inet {
      address 10.49.2.2/30;
   }
}
```

Configuring the Encapsulation Interface on Router PE1

Configure the encapsulation interface on Router PE1:

```
[edit interfaces t3-0/1/3]
unit 0 {
   family inet {
      address 192.168.197.249/30;
   }
}
```

Configuring the GRE Tunnel Interface on Router CE1

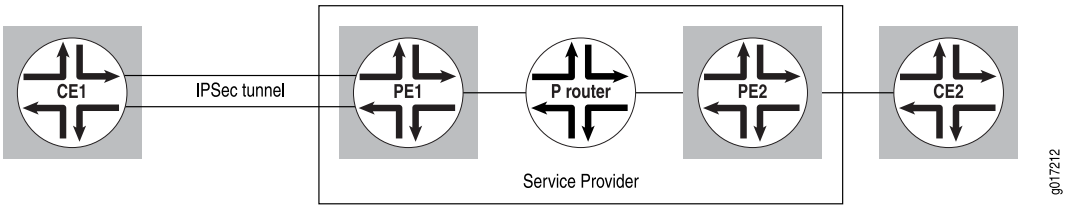
Configure the GRE tunnel interface on Router CE1:

```
[edit interfaces gr-1/2/0]
unit 0 {
   tunnel {
      source 192.168.197.250;
      destination 192.168.197.249;
   }
   family inet {
      address 10.49.2.1/30;
   }
}
```

Configuring an ES Tunnel Interface Between a PE and CE Router

This example shows how to configure an ES tunnel interface between a PE router and a CE router in a Layer 3 VPN. The network topology used in this example is shown in Figure 27 on page 179.

Figure 27: ES Tunnel Interface (IPsec Tunnel)



To configure this example, you perform the steps in the following sections:

Configuring IPsec on Router PE1

Configure IP Security (IPsec) on Router PE1:

```
[edit security]
ipsec {
   security-association sa-esp-manual {
      mode tunnel;
      manual {
         direction bidirectional {
            protocol esp;
            spi 16000;
            authentication {
               algorithm hmac-md5-96;
               key ascii-text
                  "$9$ABULt1heK87dsWLDk.P3nrevM7V24ZHkPaZ/tp0cSvWLNwgZUH";
            }
            encryption {
               algorithm des-cbc;
               key ascii-text "$9$/H8Q90IyrvL7VKMZjHqQzcyleLN";
            }
         }
      }
   }
}
```

Configuring the Routing Instance Without the Encapsulating Interface

You can configure the routing instance on Router PE1 with or without the encapsulating interface (**t3-0/1/3** in this example). The following sections explain how to configure the routing instance without it:

Configuring the Routing Instance on Router PE1

Configure the routing instance on Router PE1:

```
[edit routing-instances]
vpna {
   instance-type vrf;
   interface es-1/2/0.0;
   route-distinguisher 10.255.14.174:1;
```

```
    vrf-import vpna-import;
    vrf-export vpna-export;
    protocols {
      bgp {
        group vpna {
          type external;
          peer-as 100;
          as-override;
          neighbor 10.49.2.1;
        }
      }
    }
}
```

Configuring the ES Tunnel Interface on Router PE1

Configure the ES tunnel interface on Router PE1:

```
[edit interfaces es-1/2/0]
unit 0 {
  tunnel {
    source 192.168.197.249;
    destination 192.168.197.250;
  }
  family inet {
    address 10.49.2.2/30;
    ipsec-sa sa-esp-manual;
  }
}
```

Configuring the Encapsulating Interface for the ES Tunnel

For this example, interface **t3-0/1/3** is the encapsulating interface for the ES tunnel. Configure interface **t3-0/1/3**:

```
[edit interfaces t3-0/1/3]
unit 0 {
  family inet {
    address 192.168.197.249/30;
  }
}
```

Configuring the Routing Instance with the Encapsulating Interface

If the tunnel-encapsulating interface, **t3-0/1/3**, is also configured under the routing instance, you need to specify the routing instance name under the interface definition. The system uses this routing instance to search for the tunnel destination address for the IPsec tunnel using manual security association.

The following sections explain how to configure the routing instance with the encapsulating interface:

- Configuring the Routing Instance on Router PE1 on page 182
- Configuring the ES Tunnel Interface on Router PE1 on page 182
- Configuring the Encapsulating Interface on Router PE1 on page 182

Configuring the Routing Instance on Router PE1

Configure the routing instance on Router PE1 (including the tunnel encapsulating interface):

```
[edit routing-instances]
vpna {
   instance-type vrf;
   interface es-1/2/0.0;
   interface t3-0/1/3.0;
   route-distinguisher 10.255.14.174:1;
   vrf-import vpna-import;
   vrf-export vpna-export;
   protocols {
      bgp {
         group vpna {
            type external;
            peer-as 100;
            as-override;
            neighbor 10.49.2.1;
         }
      }
   }
}
```

Configuring the ES Tunnel Interface on Router PE1

Configure the ES tunnel interface on Router PE1:

```
[edit interfaces es-1/2/0]
unit 0 {
   tunnel {
      source 192.168.197.249;
      destination 192.168.197.250;
      routing-instance {
         destination vpna;
      }
   }
   family inet {
      address 10.49.2.2/30;
      ipsec-sa sa-esp-manual;
   }
}
```

Configuring the Encapsulating Interface on Router PE1

Configure the encapsulating interface on Router PE1:

```
[edit interfaces t3-0/1/3]
unit 0 {
   family inet {
      address 192.168.197.249/30;
   }
}
```

Configuring the ES Tunnel Interface on Router CE1

Configure the ES tunnel interface on Router CE1:

```
[edit interfaces es-1/2/0]
unit 0 {
   tunnel {
      source 192.168.197.250;
      destination 192.168.197.249;
   }
   family inet {
      address 10.49.2.1/30;
      ipsec-sa sa-esp-manual;
   }
}
```

Configuring IPsec on Router CE1

Configure IPsec on Router CE1:

```
[edit security]
ipsec {
   security-association sa-esp-manual {
      mode tunnel;
      manual {
         direction bidirectional {
            protocol esp;
            spi 16000;
            authentication {
               algorithm hmac-md5-96;
               key ascii-text
                  "$9$ABULt1heK87dsWLDk.P3nrevM7V24ZHkPaZ/tp0cSvWLNwgZUH";
            }
            encryption {
               algorithm des-cbc;
               key ascii-text "$9$/H8Q90IyrvL7VKMZjHqQzcyleLN";
            }
         }
      }
   }
}
```

Example: Disabling Normal TTL Decrementing in a VRF Routing Instance

This example shows how to disable TTL decrementing in a single VRF routing instance in a Layer 3 VPN scenario.

Requirements

Before you begin:

- Configure the router interfaces. See the *Network Interfaces Configuration Guide*.

Overview

To diagnose networking problems related to VPNs, it can be useful to disable normal time-to-live (TTL) decrementing. The IP header includes a TTL field that serves as a hop counter. At every routed hop, the TTL is decremented by one; if the TTL reaches zero before the packet reaches its destination, the packet is discarded and (optionally) an ICMP TTL exceeded message is sent to the source. MPLS labels also have a TTL field. MPLS routers copy the TTL of an IP packet when it enters a label-switched path (LSP). An IP packet with a TTL of 27 receives an MPLS label with a TTL of 27. Junos OS decrements the MPLS TTL of an MPLS-encapsulated packet in place of the IP TTL, at every label-switched hop. Because the MPLS TTL is copied (or propagated) from the IP TTL, a traceroute lists every hop in the path, be it routed or label-switched. When the packet exits the LSP, the decremented MPLS TTL is propagated back into the IP TTL field.

By default, TTL propagation is enabled. The global **no-propagate-ttl** statement disables TTL propagation at the router level and affects all RSVP-signalled or LDP-signalled LSPs. When a router acts as an ingress router for an LSP and the router configuration includes the **no-propagate-ttl** statement, the router pushes an MPLS header with a TTL value of 255, regardless of the IP packet TTL. When a router acts as the penultimate router, it pops the MPLS header without propagating the MPLS TTL into the IP packet. Thus the IP packet TTL value is preserved, regardless of the hop count of the LSP.

Instead of configuring TTL propagation behavior at the router level, you can configure the behavior for the routes in a VRF routing instance. This example shows how to disable TTL propagation for the routes in a single VRF routing instance instead of at the global router level.

The per-VRF configuration takes precedence over the global router configuration. If you disable TTL propagation on the router and explicitly enable TTL propagation for a single VRF routing instance, TTL propagation is in effect for that routing instance. To explicitly enable TTL propagation on a VRF routing instance, include the **vrf-propagate-ttl** statement in the routing instance.

When you change the TTL propagation behavior, old next hops for VRF routes are deleted from the inet.3 routing table and new next hops are added.

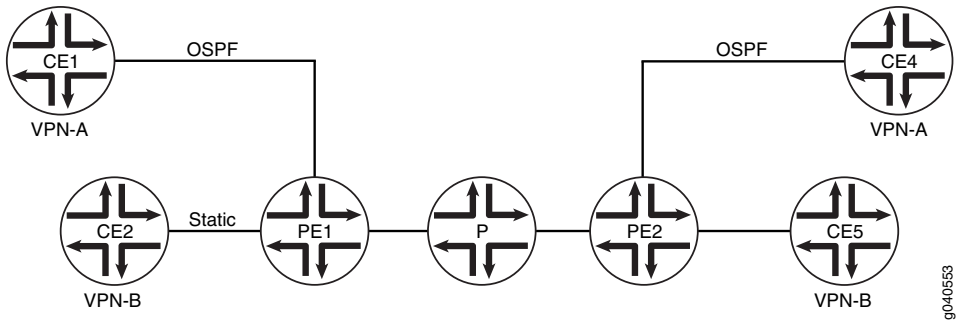
You need only configure the **vrf-propagate-ttl** or **no-vrf-propagate-ttl** statement on the ingress routers.

Topology Diagram

Figure 28 on page 185 shows the topology used in this example. Router PE1 and Router PE2 have two VPNs---VPN-A and VPN-B. Devices CE1 and CE4 belong to VPN-A. Devices CE2 and CE5 belong to VPN-B. In this example, Router PE1 has TTL propagation disabled

on VPN-A but not on VPN-B. Packets received by PE1 on the interface connected to CE1 have TTL propagation disabled. This example shows the configuration on Router PE1. You do not need to include the **no-vrf-propagate-ttl** statement on the egress router (PE2).

Figure 28: Disabling TTL Propagation for a Single VPN



Configuration

CLI Quick Configuration

To quickly disable TTL propagation in a VRF routing instance, copy the following commands and paste the commands into the CLI.

```
[edit]
set interfaces lo0 unit 0 family inet address 10.255.179.45/32 primary
set protocols mpls interface all
set protocols bgp group ibgp type internal
set protocols bgp group ibgp local-address 10.255.179.45
set protocols bgp group ibgp family inet-vpn unicast
set protocols bgp group ibgp neighbor 10.255.179.71
set protocols ospf area 0.0.0.0 interface fe-1/1/2.0
set protocols ospf area 0.0.0.0 interface fxp0.0 disable
set protocols ospf area 0.0.0.0 interface lo0.0
set protocols ldp interface all
set policy-options policy-statement VPN-A-export term a from protocol ospf
set policy-options policy-statement VPN-A-export term a from interface ge-1/2/0.0
set policy-options policy-statement VPN-A-export term a then community add VPN-A
set policy-options policy-statement VPN-A-export term a then accept
set policy-options policy-statement VPN-A-export term b then reject
set policy-options policy-statement VPN-A-import term a from protocol bgp
set policy-options policy-statement VPN-A-import term a from community VPN-A
set policy-options policy-statement VPN-A-import term a then accept
set policy-options policy-statement VPN-A-import term b then reject
set policy-options policy-statement VPN-B-export term a from protocol static
set policy-options policy-statement VPN-B-export term a then community add VPN-B
set policy-options policy-statement VPN-B-export term a then accept
set policy-options policy-statement VPN-B-export term b then reject
set policy-options policy-statement VPN-B-import term a from protocol bgp
set policy-options policy-statement VPN-B-import term a from community VPN-B
set policy-options policy-statement VPN-B-import term a then accept
set policy-options policy-statement VPN-B-import term b then reject
set policy-options policy-statement bgp-to-ospf from protocol bgp
set policy-options policy-statement bgp-to-ospf then accept
set policy-options community VPN-A members target:1:100
set policy-options community VPN-B members target:1:200
set routing-instances VPN-A instance-type vrf
```

```
set routing-instances VPN-A interface ge-1/2/0.0
set routing-instances VPN-A route-distinguisher 10.255.179.45:100
set routing-instances VPN-A interface ge-1/2/0.0
set routing-instances VPN-A no-vrf-propagate-ttl
set routing-instances VPN-A vrf-import VPN-A-import
set routing-instances VPN-A vrf-export VPN-A-export
set routing-instances VPN-A protocols ospf export bgp-to-ospf
set routing-instances VPN-A protocols ospf area 0.0.0.0 interface ge-1/2/0.0
set routing-instances VPN-B instance-type vrf
set routing-instances VPN-B interface so-0/1/0.0
set routing-instances VPN-B route-distinguisher 10.255.179.45:300
set routing-instances VPN-B vrf-import VPN-B-import
set routing-instances VPN-B vrf-export VPN-B-export
set routing-instances VPN-B routing-options static route 10.255.179.15/32 next-hop
    so-0/1/0.0
set routing-options autonomous-system 1
```

Step-by-Step Procedure

The following example requires you to navigate various levels in the configuration hierarchy. For information about navigating the CLI, see Using the CLI Editor in Configuration Mode.

To configure a flow map:

1. Configure the loopback interface.

```
[edit]
user@PE1# edit interfaces
[edit interfaces]
user@PE1# set lo0 unit 0 family inet address 10.255.179.45/32 primary
user@PE1# exit
```

2. Configure the routing protocols.

   The internal BGP neighbor address is the loopback interface address of Router PE2 in Figure 28 on page 185.

```
[edit]
user@PE1# edit protocols
[edit protocols]
user@PE1# set mpls interface all
user@PE1# set bgp group ibgp type internal
user@PE1# set bgp group ibgp local-address 10.255.179.45
user@PE1# set bgp group ibgp family inet-vpn unicast
user@PE1# set bgp group ibgp neighbor 10.255.179.71
user@PE1# set ospf area 0.0.0.0 interface fe-1/1/2.0
user@PE1# set ospf area 0.0.0.0 interface fxp0.0 disable
user@PE1# set ospf area 0.0.0.0 interface lo0.0
user@PE1# set ldp interface all
user@PE1# exit
```

3. Configure routing policies for VPN-A and VPN-B.

```
[edit]
user@PE1# edit policy-options
[edit policy-options]
user@PE1# set policy-statement VPN-A-export term a from protocol ospf
user@PE1# set policy-statement VPN-A-export term a from interface ge-1/2/0.0
```

```
user@PE1# set policy-statement VPN-A-export term a then community add VPN-A
user@PE1# set policy-statement VPN-A-export term a then accept
user@PE1# set policy-statement VPN-A-export term b then reject
user@PE1# set policy-statement VPN-A-import term a from protocol bgp
user@PE1# set policy-statement VPN-A-import term a from community VPN-A
user@PE1# set policy-statement VPN-A-import term a then accept
user@PE1# set policy-statement VPN-A-import term b then reject
user@PE1# set policy-statement VPN-B-export term a from protocol static
user@PE1# set policy-statement VPN-B-export term a then community add VPN-B
user@PE1# set policy-statement VPN-B-export term a then accept
user@PE1# set policy-statement VPN-B-export term b then reject
user@PE1# set policy-statement VPN-B-import term a from protocol bgp
user@PE1# set policy-statement VPN-B-import term a from community VPN-B
user@PE1# set policy-statement VPN-B-import term a then accept
user@PE1# set policy-statement VPN-B-import term b then reject
user@PE1# set policy-statement bgp-to-ospf from protocol bgp
user@PE1# set policy-statement bgp-to-ospf then accept
user@PE1# set community VPN-A members target:1:100
user@PE1# set community VPN-B members target:1:200
user@PE1# exit
```

4. Configure the VPN-A and VPN-B routing instances, including the **no-vrf-propagate-ttl** statement in VPN-A.

```
[edit]
user@PE1# edit routing-instances
[edit routing-instances]
user@PE1# set VPN-A instance-type vrf
user@PE1# set VPN-A interface ge-1/2/0.0
user@PE1# set VPN-A route-distinguisher 10.255.179.45:100
user@PE1# set VPN-A interface ge-1/2/0.0
user@PE1# set VPN-A no-vrf-propagate-ttl
user@PE1# set VPN-A vrf-import VPN-A-import
user@PE1# set VPN-A vrf-export VPN-A-export
user@PE1# set VPN-A protocols ospf export bgp-to-ospf
user@PE1# set VPN-A protocols ospf area 0.0.0.0 interface ge-1/2/0.0
user@PE1# set VPN-B instance-type vrf
user@PE1# set VPN-B interface so-0/1/0.0
user@PE1# set VPN-B route-distinguisher 10.255.179.45:300
user@PE1# set VPN-B vrf-import VPN-B-import
user@PE1# set VPN-B vrf-export VPN-B-export
user@PE1# set VPN-B routing-options static route 10.255.179.15/32 next-hop
  so-0/1/0.0
user@PE1# exit
```

5. Define the local autonomous system.

```
[edit]
user@PE1# edit routing-options
[edit routing-options]
user@PE1# set autonomous-system 1
user@PE1# exit
```

6. If you are done configuring the device, commit the configuration.

```
[edit]
user@PE1# commit
```

Results Confirm your configuration by entering the **show interfaces**, **show policy-options**, **show protocols**, **show routing-instances**, and **show routing-options** commands.

```
user@PE1# show interfaces
lo0 {
  unit 0 {
    family inet {
      address 10.255.179.45/32 {
        primary;
      }
    }
  }
}

user@PE1# show policy-options
policy-statement VPN-A-export {
  term a {
    from {
      protocol ospf;
      interface ge-1/2/0.0;
    }
    then {
      community add VPN-A;
      accept;
    }
  }
  term b {
    then reject;
  }
}
policy-statement VPN-A-import {
  term a {
    from {
      protocol bgp;
      community VPN-A;
    }
    then accept;
  }
  term b {
    then reject;
  }
}
policy-statement VPN-B-export {
  term a {
    from protocol static;
    then {
      community add VPN-B;
      accept;
    }
  }
  term b {
    then reject;
  }
}
policy-statement VPN-B-import {
  term a {
```

```
      from {
        protocol bgp;
        community VPN-B;
      }
      then accept;
    }
    term b {
      then reject;
    }
}
policy-statement bgp-to-ospf {
    from protocol bgp;
    then accept;
}
community VPN-A members target:1:100;
community VPN-B members target:1:200;
```

```
user@PE1# show protocols
mpls {
    interface all;
}
bgp {
    group ibgp {
        type internal;
        local-address 10.255.179.45;
        family inet-vpn {
            unicast;
        }
        neighbor 10.255.179.71;
    }
}
ospf {
    area 0.0.0.0 {
        interface fe-1/1/2.0;
        interface fxp0.0 {
            disable;
        }
        interface lo0.0;
    }
}
ldp {
    interface all;
}
```

```
user@PE1# show routing-instances
VPN-A {
    instance-type vrf;
    interface ge-1/2/0.0;
    no-vrf-propagate-ttl;
    route-distinguisher 10.255.179.45:100;
    vrf-import VPN-A-import;
    vrf-export VPN-A-export;
    protocols {
        ospf {
            export bgp-to-ospf;
            area 0.0.0.0 {
                interface ge-1/2/0.0;
```

```
        }
      }
    }
  }
  VPN-B {
    instance-type vrf;
    interface so-0/1/0.0;
    route-distinguisher 10.255.179.45:300;
    vrf-import VPN-B-import;
    vrf-export VPN-B-export;
    routing-options {
      static {
        route 10.255.179.15/32 next-hop so-0/1/0.0;
      }
    }
  }
```

```
user@PE1# show routing-options
autonomous-system 1;
```

Verification

To verify the operation, run the following commands:

- See the **TTL Action** field in the output of the **show route extensive table VPN-A** command.
- See the **TTL Action** field in the output of the **show route extensive table VPN-B** command.
- On Device CE1, run the **traceroute** command to Device CE4's loopback address.
- On Device CE4, run the **traceroute** command to Device CE1's loopback address.

Related Documentation

- Disabling Normal TTL Decrementing in the *Junos OS MPLS Applications Configuration Guide*

Example: Configuring Layer 3 VPN Protocol Family Qualifiers for Route Filters

This example shows how to control the scope of BGP import policies by configuring a family qualifier for the BGP import policy. The family qualifier specifies routes of type **inet**, **inet6**, **inet-vpn**, or **inet6-vpn**.

Requirements

This example uses Junos OS Release 10.0 or later.

Before you begin:

- Configure the router interfaces.

- Configure an interior gateway protocol. See the *Junos OS Routing Protocols Configuration Guide*.
- Configure a BGP session for multiple route types. For example, configure the session for both family **inet** routes and family **inet-vpn** routes. See Configuring IBGP Sessions Between PE Routers in VPNs and "Configuring Layer 3 VPNs to Carry IPv6 Traffic" on page 38.

Overview

Family qualifiers cause a route filter to match only one specific family. When you configure an IPv4 route filter without a family qualifier, as shown here, the route filter matches **inet** and **inet-vpn** routes.

```
route-filter ipv4-address/mask;
```

Likewise, when you configure an IPv6 route filter without a family qualifier, as shown here, the route filter matches **inet6** and **inet6-vpn** routes.

```
route-filter ipv6-address/mask;
```

Consider the case in which a BGP session has been configured for both family **inet** routes and family **inet-vpn** routes, and an import policy has been configured for this BGP session. This means that both family **inet** and family **inet-vpn** routes, when received, share the same import policy. The policy term might look as follows:

```
from {
  route-filter 0.0.0.0/0 exact;
}
then {
  next-hop self;
  accept;
}
```

This route-filter logic matches an **inet** route of 0.0.0.0 and an **inet-vpn** route whose IPv4 address portion is 0.0.0.0. The 8-byte route distinguisher portion of the **inet-vpn** route is not considered in the route-filter matching. This is a change in Junos OS behavior that was introduced in Junos OS Release 10.0.

If you do not want your policy to match both types of routes, add a family qualifier to your policy. To have the route-filter match only **inet** routes, add the family **inet** policy qualifier. To have the route-filter match only **inet-vpn** routes, add the family **inet-vpn** policy qualifier.

The family qualifier is evaluated before the route-filter is evaluated. Thus, the route-filter is not evaluated if the family match fails. The same logic applies to family **inet6** and family **inet6-vpn**. The route-filter used in the **inet6** example must use an IPv6 address. There is a potential efficiency gain in using a family qualifier because the family qualifier is tested before most other qualifiers, quickly eliminating routes from undesired families.

Configuration

CLI Quick Configuration

To quickly configure this example, copy the following commands, paste them into a text file, remove any line breaks, change any details necessary to match your network

configuration, and then copy and paste the commands into the CLI at the **[edit]** hierarchy level.

inet Example

```
set policy-options policy-statement specific-family from family inet
set policy-options policy-statement specific-family from route-filter 0.0.0.0/0 exact
set policy-options policy-statement specific-family then next-hop self
set policy-options policy-statement specific-family then accept
set protocols bgp import specific-family
```

Inet-vpn Example

```
set policy-options policy-statement specific-family from family inet-vpn
set policy-options policy-statement specific-family from route-filter 0.0.0.0/0 exact
set policy-options policy-statement specific-family then next-hop self
set policy-options policy-statement specific-family then accept
set protocols bgp import specific-family
```

inet6 Example

```
set policy-options policy-statement specific-family from family inet6
set policy-options policy-statement specific-family from route-filter 0::0/0 exact
set policy-options policy-statement specific-family then next-hop self
set policy-options policy-statement specific-family then accept
set protocols bgp import specific-family
```

Inet6-vpn Example

```
set policy-options policy-statement specific-family from family inet6-vpn
set policy-options policy-statement specific-family from route-filter 0::0/0 exact
set policy-options policy-statement specific-family then next-hop self
set policy-options policy-statement specific-family then accept
set protocols bgp import specific-family
```

Step-by-Step Procedure

The following example requires you to navigate various levels in the configuration hierarchy. For information about navigating the CLI, see the *Junos OS CLI User Guide*.

To configure a flow map:

1. Configure the family qualifier.

   ```
   [edit policy-options]
   user@host# set policy-statement specific-family from family inet
   ```

2. Configure the route filter.

   ```
   [edit policy-options]
   user@host# set policy-statement specific-family from route-filter 0.0.0.0/0 exact
   ```

3. Configure the policy actions.

   ```
   [edit policy-options]
   user@host# set policy-statement specific-family then next-hop self
   user@host# set policy-statement specific-family then accept
   ```

4. Apply the policy.

   ```
   [edit protocols bgp]
   user@host# set import specific-family
   ```

Results

Confirm your configuration by entering the **show policy-options** and **show protocols** commands.

```
user@host# show policy-options
policy-statement specific-family {
  from {
    family inet;
    route-filter 0.0.0.0/0 exact;
  }
  then {
    next-hop self;
    accept;
  }
}

user@host# show protocols
bgp {
  import specific-family;
}
```

If you are done configuring the router, enter **commit** from configuration mode.

Repeat the procedure for every protocol family for which you need a specific route-filter policy.

Verification

To verify the configuration, run the following commands:

- **show route advertising-protocol bgp *neighbor* detail**
- **show route instance *instance-name* detail**

Related Documentation

- *Junos OS Policy Framework Configuration Guide*

Example: Configuring Route Resolution

This example shows how to configure a routing table to accept routes from specific routing tables. It also shows how to configure a routing table to use specific import policies to produce a route resolution table to resolve routes.

Requirements

Before you begin, configure a Layer 3 VPN, as shown in one of the following examples:

- Example: Configuring Interprovider Layer 3 VPN Option A on page 233
- Example: Configuring Interprovider Layer 3 VPN Option B on page 253
- Example: Configuring and Verifying the Auto Export Feature

Overview

One method to achieve IPv4 route scaling is to modify how BGP routes are added to the forwarding tables. By default, the Routing Protocol Process (rpd) adds all the routes in inet.0 and inet.3 to the resolution tree. Normally, this includes the resolved IPv4 BGP routes, which can increase memory consumption. To achieve better scaling for IPv4 routes, this example shows how to configure the Junos OS so that resolved BGP routes are not added to the resolution tree. This is achieved by applying an import policy on the route resolution table, which ensures it does not accept any BGP routes from inet.0.

You would apply this configuration to all provider edge (PE) routers in the Layer 3 VPN.

TIP: Route resolution is useful in multiple scenarios, not just the one shown in this example.

One scenario for route resolution is when you have a label-switched path configured from a route reflector (RR) to a PE router, or when the PE routers only peer with the RR. This can result in routes being hidden. To resolve this issue, the following configuration changes the default resolution behavior by using inet.0 for next-hop resolution.

```
user@PE# set routing-options resolution rib bgp.l3vpn.0 resolution-ribs inet.0
```

Another scenario for route resolution is when you want to make sure that external BGP routes are not used to resolve indirect next-hops on other BGP routes. Suppose, for instance, that you do not want your default discard route to affect the next-hop feasibility algorithm. The following configuration enables you to avoid using a BGP prefix as the next-hop route.

```
user@PE# set policy-options policy-statement accept-igp-only term 1 from
  protocol [ ospf ospf3 ]
user@PE# set policy-options policy-statement accept-igp-only term 1 then
  accept
user@PE# set policy-options policy-statement accept-igp-only then reject
user@PE# set routing-options resolution rib inet.0 import accept-igp-only
```

Configuration

CLI Quick Configuration

To quickly configure this example, copy the following commands, paste them into a text file, remove any line breaks, change any details necessary to match your network configuration, and then copy and paste the commands into the CLI at the **[edit]** hierarchy level.

PE Router

```
set policy-options policy-statement protocol-bgp from protocol bgp
set policy-options policy-statement protocol-bgp then reject
set routing-options resolution rib inet.0 import protocol-bgp
```

Step-by-Step Procedure

The following example requires you to navigate various levels in the configuration hierarchy. For information about navigating the CLI, see Using the CLI Editor in Configuration Mode in the *Junos OS CLI User Guide*.

To configure route resolution:

1. Configure the routing policy.

```
[edit policy-options policy-statement protocol-bgp]
user@PE# set from protocol bgp
user@PE# set then reject
```

2. Apply the routing policy.

```
[edit routing-options resolution]
user@PE# set rib inet.0 import protocol-bgp
```

3. If you are done configuring the device, commit the configuration.

```
[edit]
user@PE# commit
```

Results

Confirm your configuration by issuing the **show policy-options** and **show routing-options** commands.

```
user@PE# show policy-options
resolution {
    rib inet.0 {
        import protocol-bgp;
    }
}
```

```
user@PE# show routing-options
policy-statement protocol-bgp {
    from protocol bgp;
    then reject;
}
```

Verification

Confirm that the configuration is working properly by running the following commands:

- show route
- show route forwarding-table
- show route resolution

Related Documentation

- Example: Configuring BGP Route Reflectors

CHAPTER 5

Layer 3 VPN Internet Access Examples

Non-VRF Internet Access

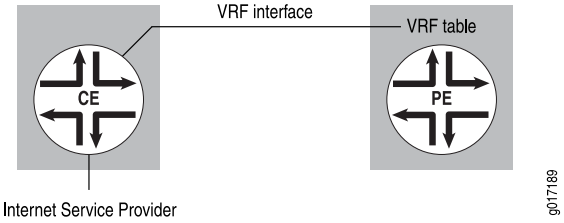
Junos OS supports Internet access from a Layer 3 virtual private network (VPN). This chapter provides examples that demonstrate how to configure a provider edge (PE) router to provide Internet access to customer edge (CE) routers in a VPN. The method you use depends on the needs and specifications of the individual network. To provide Internet access through a Layer 3 VPN, you need to configure policies on the PE router. You also need to configure the **next-table** statement at the **[edit routing-instances *routing-instance-name* routing-options static route]** hierarchy level. When configured, this statement can point a default route from the VPN table (routing instance) to the main routing table (default instance) **inet.0**. The main routing table stores all Internet routes and is where final route resolution occurs.

The following sections describe ways to provide Internet access to a CE router in a Layer 3 VPN without using the VPN routing and forwarding (VRF) interface. Because these methods effectively bypass the Layer 3 VPN, they are not discussed in detail.

CE Router Accesses Internet Independently of the PE Router

In this configuration, the PE router does not provide the Internet access. The CE router sends Internet traffic either to another service provider, or to the same service provider but a different router. The PE router handles Layer 3 VPN traffic only (see Figure 29 on page 198).

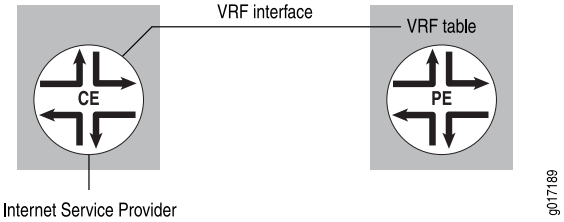
Figure 29: PE Router Does Not Provide Internet Access



PE Router Provides Layer 2 Internet Service

In this configuration, the PE router acts as a Layer 2 device, providing a Layer 2 connection (such as circuit cross-connect [CCC]) to another router that has a full set of Internet routes. The CE router can use just one physical interface and two logical interfaces to the PE router, or it can use multiple physical interfaces to the PE router (see Figure 30 on page 198).

Figure 30: PE Router Connects to a Router Connected to the Internet



Distributed Internet Access

In this scenario, the PE routers provide Internet access to the CE routers. In the examples that follow, it is assumed that the Internet routes (or defaults) are present in the **inet.0** table of the PE routers that provide Internet access to selected CE routers.

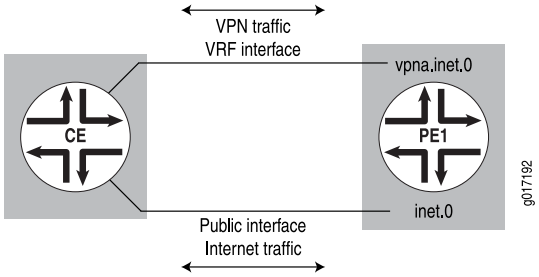
When accessing the Internet from a VPN, Network Address Translation (NAT) must be performed between the VPN's private addresses and the public addresses used on the Internet unless the VPN is using the public address space. This section includes several examples of how to provide Internet access for VPNs, most of which require that the CE routers perform the address translation. The "Routing Internet Traffic Through a Separate NAT Device" on page 214 example, however, requires that the service provider supply the NAT functionality using a NAT device connected to the PE router.

In all of the examples, the VPN's public IP address pool (whose entries correspond to the translated private addresses) must be added to the **inet.0** table and propagated to the Internet routers to receive reverse traffic from public destinations.

Routing VPN and Internet Traffic Through Different Interfaces

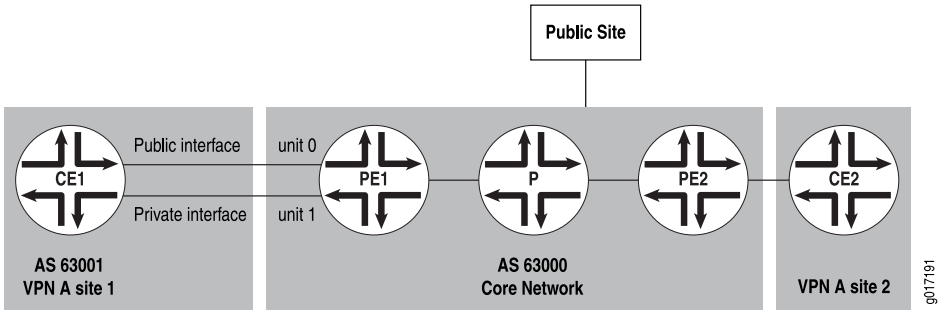
In this example, VPN and Internet traffic are routed through different interfaces. The CE router sends the VPN traffic through the VPN interface and sends the Internet traffic through a separate interface that is part of the main routing table on Router PE1 (the CE router can use either one physical interface with two logical units or two physical interfaces). NAT also occurs on the CE router (see Figure 31 on page 199).

Figure 31: Routing VPN and Internet Traffic Through Different Interfaces



The PE router is configured to install and advertise the public IP address pool for the VPN to other core routers (for return traffic). The VPN traffic is routed normally. Figure 32 on page 199 illustrates the PE router's VPN configuration.

Figure 32: Example of Internet Traffic Routed Through Separate Interfaces



The configuration in this example has the following features:

- Router PE1 uses two logical interfaces to connect to Router CE1 using Frame Relay encapsulation.
- The routing protocol between Router PE1 and Router CE1 is the EBGP.
- Router CE1's public IP address pool is **10.12.1.1** through **10.12.1.254** (**10.12.1.0/24**).
- The **next-hop-self** setting is derived from the **fix-nh policy** statement on Router PE1. PE routers are forced to use **next-hop-self** so that next-hop resolution is done only for the PE router's loopback address for non-VPN routes (by default, VPN–Internet Protocol version 4 [IPv4] routes are sent by means of **next-hop-self**).

You can configure Router CE1 with a static default route pointing to its public interface for everything else.

The following sections show how to route VPN and Internet traffic through different interfaces:

Configuring Interfaces on Router PE1

Configure an interface to handle VPN traffic and an interface to handle Internet traffic:

```
[edit]
interfaces {
    t3-0/2/0 {
        dce;
        encapsulation frame-relay;
        unit 0 {
            description "to CE1 VPN interface";
            dlci 10;
            family inet {
                address 192.168.197.13/30;
            }
        }
        unit 1 {
            description "to CE1 public interface";
            dlci 20;
            family inet {
                address 192.168.198.201/30;
            }
        }
    }
}
```

Configuring Routing Options on Router PE1

Configure a static route on Router PE1 to install a route to the CE router's public IP address pool in **inet.0**:

```
[edit]
routing-options {
    static {
        route 10.12.1.0/24 next-hop 192.168.198.202;
    }
}
```

Configuring BGP, IS-IS, and LDP Protocols on Router PE1

Configure BGP on Router PE1 to allow non-VPN and VPN peering and to advertise the VPN's public IP address pool:

```
[edit]
```

```
protocols {
  bgp {
    group pe-pe {
      type internal;
      local-address 10.255.14.171;
      family inet {
        any;
      }
      family inet-vpn {
        any;
      }
      export [fix-nh redist-static];
      neighbor 10.255.14.177;
      neighbor 10.255.14.179;
    }
  }
}
```

Configure IS-IS on Router PE1 to allow access to internal routes:

```
[edit protocols]
isis {
  level 1 disable;
  interface so-0/0/0.0;
  interface lo0.0;
}
```

Configure LDP on Router PE1 to tunnel VPN routes:

```
[edit protocols]
ldp {
  interface so-0/0/0.0;
}
```

Configuring a Routing Instance on Router PE1

Configure a routing instance on Router PE1:

```
[edit]
routing-instances {
  vpna {
    instance-type vrf;
    interface t3-0/2/0.0;
    route-distinguisher 10.255.14.171:100;
    vrf-import vpna-import;
    vrf-export vpna-export;
    protocols {
      bgp {
        group to-CE1 {
          peer-as 63001;
          neighbor 192.168.197.14;
        }
      }
    }
  }
}
```

Configuring Policy Options on Router PE1

You need to configure policy options on Router PE1. The **fix-nh** policy statement sets **next-hop-self** for all non-VPN routes:

```
[edit]
policy-options {
  policy-statement fix-nh {
    then {
      next-hop self;
    }
  }
}
```

The **redist-static** policy statement advertises the VPN's public IP address pool:

```
[edit policy-options]
policy-statement redist-static {
  term a {
    from {
      protocol static;
      route-filter 10.12.1.0/24 exact;
    }
    then accept;
  }
  term b {
    then reject;
  }
}
```

Configure import and export policies for **vpna**:

```
[edit policy-options]
policy-statement vpna-import {
  term a {
    from {
      protocol bgp;
      community vpna-comm;
    }
    then accept;
  }
  term b {
    then reject;
  }
}
policy-statement vpna-export {
  term a {
    from protocol bgp;
    then {
      community add vpna-comm;
      accept;
    }
  }
  term b {
    then reject;
  }
```

```
}
community vpna-comm members target:63000:100;
```

Traffic Routed by Different Interfaces: Configuration Summarized by Router

Router PE1

Interfaces

```
interfaces {
  t3-0/2/0 {
    dce;
    encapsulation frame-relay;
    unit 0 {
      description "to CE1 VPN interface";
      dlci 10;
      family inet {
        address 192.168.197.13/30;
      }
    }
    unit 1 {
      description "to CE1 public interface";
      dlci 20;
      family inet {
        address 192.168.198.201/30;
      }
    }
  }
}
```

Routing Options

```
routing-options {
  static {
    route 10.12.1.0/24 next-hop 192.168.198.202;
  }
}
```

BGP Protocol

```
protocols {
  bgp {
    group pe-pe {
      type internal;
      local-address 10.255.14.171;
      family inet {
        any;
      }
      family inet-vpn {
        any;
      }
      export [ fix-nh redist-static];
      neighbor 10.255.14.177;
      neighbor 10.255.14.179;
    }
  }
}
```

IS-IS Protocol

```
isis {
  level 1 disable;
  interface so-0/0/0.0;
```

```
    interface lo0.0;
}
```

LDP Protocol

```
ldp {
    interface so-0/0/0.0;
}
```

Routing Instance

```
routing-instances {
    vpna {
        instance-type vrf;
        interface t3-0/2/0.0;
        route-distinguisher 10.255.14.171:100;
        vrf-import vpna-import;
        vrf-export vpna-export;
        protocols {
            bgp {
                group to-CE1 {
                    peer-as 63001;
                    neighbor 192.168.197.14;
                }
            }
        }
    }
}
```

Policy Options/Policy Statements

```
policy-options {
    policy-statement fix-nh {
        then {
            next-hop self;
        }
    }
    policy-statement redist-static {
        term a {
            from {
                protocol static;
                route-filter 10.12.1.0/24 exact;
            }
            then accept;
        }
        term b {
            then reject;
        }
    }
}
```

Import and Export Policies

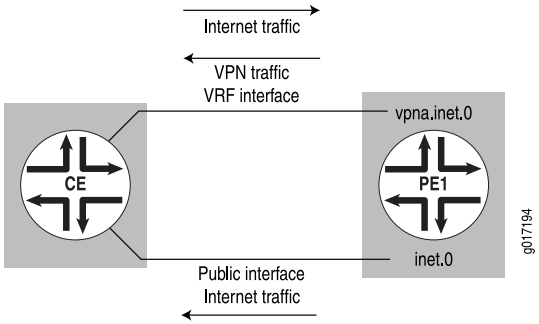
```
policy-statement vpna-import {
    term a {
        from {
            protocol bgp;
            community vpna-comm;
        }
        then accept;
    }
    term b {
        then reject;
```

```
    }
  }
  policy-statement vpna-export {
    term a {
      from protocol bgp;
      then {
        community add vpna-comm;
        accept;
      }
    }
    term b {
      then reject;
    }
  }
  community vpna-comm members target:63000:100;
```

Routing VPN and Outgoing Internet Traffic Through the Same Interface and Routing Return Internet Traffic Through a Different Interface

In this example, the CE router sends VPN and Internet traffic through the same interface but receives return Internet traffic through a different interface. The PE router has a default route in the VRF table pointing to the main routing table **inet.0**. It routes the VPN public IP address pool (return Internet traffic) through a different interface in **inet.0** (see Figure 33 on page 205). The CE router still performs NAT functions.

Figure 33: VPN and Outgoing Internet Traffic Routed Through the Same Interface and Return Internet Traffic Routed Through a Different Interface



The following section shows how to route VPN and outgoing Internet traffic through the same interface and routing return Internet traffic through a different interface:

- Configuration for Router PE1 on page 205

Configuration for Router PE1

This example has the same configuration as Router PE1 in "Routing VPN and Internet Traffic Through Different Interfaces" on page 199. It uses the topology shown in Figure 32 on page 199. The default route to the VPN routing table is configured differently. At the **[edit routing-instances *routing-instance-name* routing-options]** hierarchy level, you configure a default static route that is installed in **vpna.inet.0** and points to **inet.0** for resolution:

```
[edit]
```

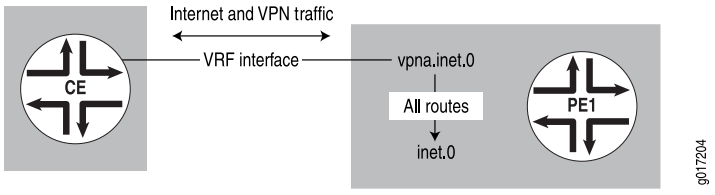
```
routing-instances {
   vpna {
      instance-type vrf;
      interface t3-0/2/0.0;
      route-distinguisher 10.255.14.171:100;
      vrf-import vpna-import;
      vrf-export vpna-export;
      routing-options {
         static {
            route 0.0.0.0/0 next-table inet.0;
         }
      }
      protocols {
         bgp {
            group to-CE1 {
               peer-as 63001;
               neighbor 192.168.197.14;
            }
         }
      }
   }
}
```

You also need to change the configuration of Router CE1 (from the configuration that works with the configuration for Router PE1 described in "Routing VPN and Internet Traffic Through Different Interfaces" on page 199) to account for the differences in the configuration of the PE routers.

Routing VPN and Internet Traffic Through the Same Interface Bidirectionally (VPN Has Public Addresses)

This section shows how to configure a single logical interface to handle VPN and Internet traffic traveling both to and from the Internet and the CE router. This interface can handle both VPN and Internet traffic as long as there are no private addresses in the VPN. The VPN routes received from the CE router are added to the main routing table **inet.0** by means of routing table groups. This allows the PE router to attract the return traffic from the Internet (see Figure 34 on page 206).

Figure 34: Interface Configured to Carry Both Internet and VPN Traffic



In this example, the CE router does not need to perform NAT, because all the VPN routes are public. The CE router has a single interface to the PE router, to which it advertises VPN routes. The PE router has a default route in the VRF table pointing to the main routing table **inet.0**. The PE router also imports VPN routes received from the CE router into **inet.0** by means of routing table groups.

The following configuration for Router PE1 uses the same topology as in "Routing VPN and Internet Traffic Through Different Interfaces" on page 199. This configuration uses a single logical interface (instead of two) between Router PE1 and Router CE1.

The following sections show how to route VPN and Internet traffic through the same interface bidirectionally (VPN has public addresses):

Configuring Routing Options on Router PE1

Configure a routing table group definition for installing VPN routes in routing table groups **vpna.inet.0** and **inet.0**:

```
[edit]
routing-options {
   rib-groups {
      vpna-to-inet0 {
         import-rib [ vpna.inet.0 inet.0 ];
      }
   }
}
```

Configuring Routing Protocols on Router PE1

Configure MPLS, BGP, IS-IS, and LDP protocols on Router PE1. This configuration does not include the **policy redist-static** statement at the **[edit protocols bgp group pe-pe]** hierarchy level. The VPN routes are sent directly to IBGP.

Configure BGP on Router PE1 to allow non-VPN and VPN peering, and to advertise the VPN's public IP address pool:

```
[edit]
protocols {
   mpls {
      interface t3-0/2/0.0;
   }
   bgp {
      group pe-pe {
         type internal;
         local-address 10.255.14.171;
         family inet {
            any;
         }
         family inet-vpn {
            any;
         }
         export fix-nh;
         neighbor 10.255.14.177;
         neighbor 10.255.14.173;
```

```
        }
    }
    isis {
        level 1 disable;
        interface so-0/0/0.0;
        interface lo0.0;
    }
    ldp {
        interface so-0/0/0.0;
    }
}
```

Configuring the Routing Instance on Router PE1

This section describes how to configure the routing instance on Router PE1. The static route defined in the **routing-options** statement directs Internet traffic from the CE router to the **inet.0** routing table. The routing table group defined by the **rib-group vpna-to-inet0** statement adds the VPN routes to **inet.0**.

Configure the routing instance on Router PE1:

```
[edit]
routing-instances {
    vpna {
        instance-type vrf;
        interface t3-0/2/0.0;
        route-distinguisher 10.255.14.171:100;
        vrf-import vpna-import;
        vrf-export vpna-export;
        routing-options {
            static {
                route 0.0.0.0/0 next-table inet.0;
            }
        }
        protocols {
            bgp {
                group to-CE1 {
                    family inet {
                        unicast {
                            rib-group vpna-to-inet0;
                        }
                    }
                    peer-as 63001;
                    neighbor 192.168.197.14;
                }
            }
        }
    }
}
```

You must configure Router CE1 to forward all traffic to Router PE1 using a default route. Alternatively, the default route can be advertised from Router PE1 to Router CE1 with EBGP.

Traffic Routed Through the Same Interface Bidirectionally: Configuration Summarized by Router

Router PE1

This example uses the same configuration as in "Routing VPN and Internet Traffic Through Different Interfaces" on page 199. This configuration uses a single logical interface (instead of two) between Router PE1 and Router CE1.

Routing Options

```
routing-options {
  rib-groups {
    vpna-to-inet0 {
      import-rib [ vpna.inet.0 inet.0 ];
    }
  }
}
```

Routing Protocols

```
protocols {
  mpls {
    interface t3-0/2/0.0;
  }
  bgp {
    group pe-pe {
      type internal;
      local-address 10.255.14.171;
      family inet {
        any;
      }
      family inet-vpn {
        any;
      }
      export fix-nh;
      neighbor 10.255.14.177;
      neighbor 10.255.14.173;
    }
  }
  isis {
    level 1 disable;
    interface so-0/0/0.0;
    interface lo0.0;
  }
  ldp {
    interface so-0/0/0.0;
  }
}
```

Routing Instance

```
routing-instances {
  vpna {
    instance-type vrf;
    interface t3-0/2/0.0;
    route-distinguisher 10.255.14.171:100;
    vrf-import vpna-import;
    vrf-export vpna-export;
    routing-options {
      static {
        route 0.0.0.0/0 next-table inet.0;
```

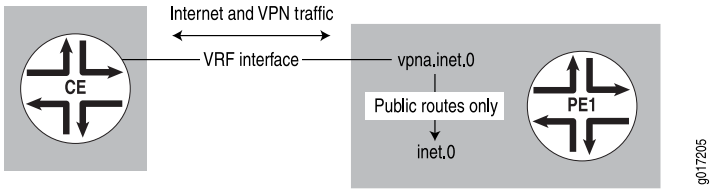
```
        }
      }
      protocols {
        bgp {
          group to-CE1 {
            family inet {
              unicast {
                rib-group vpna-to-inet0;
              }
            }
            peer-as 63001;
            neighbor 192.168.197.14;
          }
        }
      }
    }
  }
```

Routing VPN and Internet Traffic Through the Same Interface Bidirectionally (VPN Has Private Addresses)

The example in this section shows how to route VPN and Internet traffic through the same interface in both directions (from the CE router to the Internet and from the Internet to the CE router). The VPN in this example has private addresses. If you can configure EBGP on the CE router, you can configure a PE router using the configuration outlined in "Routing VPN and Internet Traffic Through the Same Interface Bidirectionally (VPN Has Public Addresses)" on page 206, even if the VPN has private addresses.

In the example described in this section, the CE router uses separate communities to advertise its VPN routes and public routes. The PE router selectively imports only the public routes into the **inet.0** routing table. This configuration ensures that return traffic from the Internet uses the same interface between the PE and CE routers as that used by VPN traffic going out to public Internet addresses (see Figure 35 on page 210).

Figure 35: VPN and Internet Traffic Routed Through the Same Interface



In this example, the CE router has one interface and a BGP session with the PE router, and it tags VPN routes and Internet routes with different communities. The PE router has one interface, selectively imports routes for the VPN's public IP address pool into **inet.0**, and has a default route in the VRF routing table pointing to **inet.0**.

The following sections show how to route VPN and Internet traffic through the same interface bidirectionally (VPN has private addresses):

- Configuring Routing Options for Router PE1 on page 211
- Configuring a Routing Instance for Router PE1 on page 211

Configuring Routing Options for Router PE1

On Router PE1, configure a routing table group to install VPN routes in the **vpna.inet.0** and **inet.0** routing tables:

```
[edit]
routing-options {
  rib-groups {
    vpna-to-inet0 {
      import-rib [ vpna.inet.0 inet.0 ];
    }
  }
}
```

Configuring a Routing Instance for Router PE1

On Router PE1, configure a routing instance. As part of the configuration for the routing instance, configure a static route that is installed in **vpna.inet.0** and is pointed at **inet.0** for resolution.

```
[edit]
routing-instances {
  vpna {
    instance-type vrf;
    interface t3-0/2/0.0;
    route-distinguisher 10.255.14.171:100;
    vrf-import vpna-import;
    vrf-export vpna-export;
    routing-options {
      static {
        route 0.0.0.0/0 next-table inet.0;
      }
    }
  }
}
```

At the **[edit routing-instances vpna protocols bgp]** hierarchy level, configure a policy (**import-public-addr-to-inet0**) to import public routes into **inet.0** and a routing table group (**vpna-to-inet0**) to allow BGP to install routes into multiple routing tables (**vpna.inet.0** and **inet.0**):

```
[edit routing-instances vpna]
protocols {
  bgp {
    group to-CE1 {
      import import-public-addr-to-inet0;
      family inet {
        unicast {
          rib-group vpna-to-inet0;
        }
      }
```

```
        peer-as 63001;
        neighbor 192.168.197.14;
      }
    }
}
```

Configuring Policy Options for Router PE1

Configure the policy options for Router PE1 to accept all routes initially (**term a**) and then to install routes with a **public-comm** community into routing table **inet.0** (**term b**):

```
[edit]
policy-options {
  policy-statement import-public-addr-to-inet0 {
    term a {
      from {
        protocol bgp;
        rib vpna.inet.0;
        community [ public-comm private-comm ];
      }
      then accept;
    }
    term b {
      from {
        protocol bgp;
        community public-comm;
      }
      to rib inet.0;
      then accept;
    }
    term c {
      then reject;
    }
  }
  community private-comm members target:1:333;
  community public-comm members target:1:111;
  community vpna-comm members target:63000:100;
}
```

Traffic Routed by the Same Interface Bidirectionally (VPN Has Private Addresses): Configuration Summarized by Router

Router PE1

Routing Options

```
[edit]
routing-options {
  rib-groups {
    vpna-to-inet0 {
      import-policy import-public-addr-to-inet0;
      import-rib [ vpna.inet.0 inet.0 ];
    }
  }
}
```

Routing Instances

```
[edit]
routing-instances {
```

```
  vpna {
    instance-type vrf;
    interface t3-0/2/0.0;
    route-distinguisher 10.255.14.171:100;
    vrf-import vpna-import;
    vrf-export vpna-export;
    routing-options {
      static {
        route 0.0.0.0/0 next-table inet.0;
      }
    }
  }
}
```

Routing Instances Protocols BGP

```
[edit routing-instances vpna]
protocols {
  bgp {
    group to-CE1 {
      family inet {
        unicast {
          rib-group vpna-to-inet0;
        }
      }
      peer-as 63001;
      neighbor 192.168.197.14;
    }
  }
}
```

Policy Options

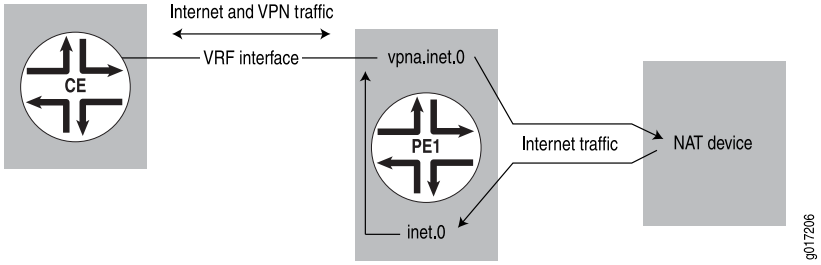
```
[edit]
policy-options {
  policy-statement import-public-addr-to-inet0 {
    term a {
      from {
        protocol bgp;
        rib vpna.inet.0;
        community [ public-comm private-comm ];
      }
      then accept;
    }
    term b {
      from {
        protocol bgp;
        community public-comm;
      }
      to rib inet.0;
      then accept;
    }
    term c {
      then reject;
    }
  }
  community private-comm members target:1:333;
  community public-comm members target:1:111;
  community vpna-comm members target:63000:100;
```

}

Routing Internet Traffic Through a Separate NAT Device

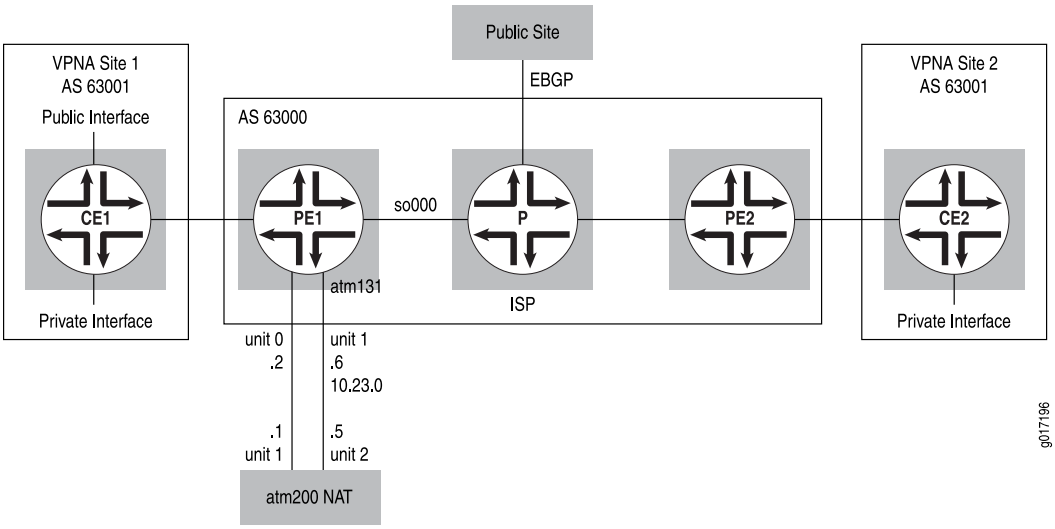
In this example, the CE router does not perform NAT. It sends both VPN and Internet traffic over the same interface to the PE router. The PE router is connected to an NAT device by means of two interfaces. One interface is configured in the PE router's VRF table and points to a VPN interface on the NAT device, which can route Internet traffic for the VPN. The other interface is in a default instance; for example, part of public routing table **inet.0**. There can be a single physical connection between the PE router and the NAT device and multiple logical connections—one for each VRF table and another interface—as part of the global routing table (see Figure 36 on page 214).

Figure 36: Internet Traffic Routed Through a Separate NAT Device



This example's topology expands upon that illustrated in Figure 32 on page 199. The CE router sends both VPN and Internet traffic to Router PE1. VPN traffic is routed based on the VPN routes received by Router PE1. Traffic for everything else is sent to the NAT device using Router PE1's private interface to the NAT device, which then translates the private addresses and sends the traffic back to Router PE1 using that router's public interface (see Figure 37 on page 214).

Figure 37: Internet Traffic Routed Through a NAT Example Topology



The following sections show how to route Internet traffic through a separate NAT device:

Configuring Interfaces on Router PE1

Configure an interface for VPN traffic to and from Router CE1, an interface for VPN traffic to and from the NAT device, and an interface for Internet traffic to and from the NAT device:

```
[edit]
interfaces {
    t3-0/2/0 {
        dce;
        encapsulation frame-relay;
        unit 0 {
            description "to CE1 VPN interface";
            dlci 10;
            family inet {
                address 192.168.197.13/30;
            }
        }
    }
    at-1/3/1 {
        atm-options {
            vpi 1 maximum-vcs 255;
        }
        unit 0 {
            description "to NAT VPN interface";
            vci 1.100;
            family inet {
                address 10.23.0.2/32 {
                    destination 10.23.0.1;
                }
            }
        }
        unit 1 {
            description "to NAT public interface";
            vci 1.101;
            family inet {
                address 10.23.0.6/32 {
                    destination 10.23.0.5;
                }
            }
        }
    }
}
```

Configuring Routing Options for Router PE1

Configure a static route on Router PE1 to direct Internet traffic to the CE router through the NAT device. Router PE1 distributes this route to the Internet.

```
[edit]
routing-options {
  static {
    route 10.12.1.0/24 next-hop 10.23.0.5;
  }
}
```

Configuring Routing Protocols on Router PE1

Configure MPLS, BGP, IS-IS, and LDP on Router PE1. For the MPLS configuration, include the NAT device's VPN interface in the VRF table. As part of the BGP configuration, include a policy to advertise the public IP address pool:

```
[edit]
protocols {
  mpls {
    interface t3-0/2/0.0;
    interface at-1/3/1.0;
  }
  bgp {
    group pe-pe {
      type internal;
      local-address 10.255.14.171;
      family inet {
        any;
      }
      family inet-vpn {
        any;
      }
      export [ fix-nh redist-static ];
      neighbor 10.255.14.177;
      neighbor 10.255.14.173;
    }
  }
  isis {
    level 1 disable;
    interface so-0/0/0.0;
    interface lo0.0;
  }
  ldp {
    interface so-0/0/0.0;
  }
}
```

Configuring a Routing Instance for Router PE1

Configure a routing instance on Router PE1. As part of the routing instance configuration, under **routing-options**, configure a static default route in **vpna.inet.0** pointing to the NAT device's VPN interface (this directs all non-VPN traffic to the NAT device):

```
[edit]
routing-instances {
    vpna {
        instance-type vrf;
        interface t3-0/2/0.0;
        interface at-1/3/1.0;
        route-distinguisher 10.255.14.171:100;
        vrf-import vpna-import;
        vrf-export vpna-export;
        routing-options {
            static {
                route 0.0.0.0/0 next-hop 10.23.0.1;
            }
        }
        protocols {
            bgp {
                group to-CE1 {
                    peer-as 63001;
                    neighbor 192.168.197.14;
                }
            }
        }
    }
}
policy-options {
    policy-statement fix-nh {
        then {
            next-hop self;
        }
    }
    policy-statement redist-static {
        term a {
            from {
                protocol static;
                route-filter 10.12.1.0/24 exact;
            }
            then accept;
        }
        term b {
            from protocol bgp;
            then accept;
        }
        term c {
            then accept;
        }
    }
    policy-statement vpna-import {
        term a {
            from {
                protocol bgp;
                community vpna-comm;
            }
            then accept;
        }
        term b {
            then reject;
```

```
    }
  }
  policy-statement vpna-export {
    term a {
      from protocol bgp;
      then {
        community add vpna-comm;
        accept;
      }
    }
    term b {
      then reject;
    }
  }
  community vpna-comm members target:63000:100;
}
```

Traffic Routed by Separate NAT Device: Configuration Summarized by Router

Router PE1

Interfaces

```
interfaces {
  t3-0/2/0 {
    dce;
    encapsulation frame-relay;
    unit 0 {
      description "to CE1 VPN interface";
      dlci 10;
      family inet {
        address 192.168.197.13/30;
      }
    }
  }
  at-1/3/1 {
    atm-options {
      vpi 1 maximum-vcs 255;
    }
    unit 0 {
      description "to NAT VPN interface";
      vci 1.100;
      family inet {
        address 10.23.0.2/32 {
          destination 10.23.0.1;
        }
      }
    }
    unit 1 {
      description "to NAT public interface";
      vci 1.101;
      family inet {
        address 10.23.0.6/32 {
          destination 10.23.0.5;
        }
      }
    }
  }
```

```
}
```

Routing Options

```
routing-options {
  static {
    route 10.12.1.0/24 next-hop 10.23.0.5;
  }
}
```

Routing Protocols

```
protocols {
  mpls {
    interface t3-0/2/0.0;
    interface at-1/3/1.0;
  }
  bgp {
    group pe-pe {
      type internal;
      local-address 10.255.14.171;
      family inet {
        any;
      }
      family inet-vpn {
        any;
      }
      export [ fix-nh redist-static ];
      neighbor 10.255.14.177;
      neighbor 10.255.14.173;
    }
  }
  isis {
    level 1 disable;
    interface so-0/0/0.0;
    interface lo0.0;
  }
  ldp {
    interface so-0/0/0.0;
  }
}
```

Routing Instance

```
routing-instances {
  vpna {
    instance-type vrf;
    interface t3-0/2/0.0;
    interface at-1/3/1.0;
    route-distinguisher 10.255.14.171:100;
    vrf-import vpna-import;
    vrf-export vpna-export;
    routing-options {
      static {
        route 0.0.0.0/0 next-hop 10.23.0.1;
      }
    }
    protocols {
      bgp {
        group to-CE1 {
          peer-as 63001;
```

```
            neighbor 192.168.197.14;
          }
        }
      }
    }
}
```

Policy Options

```
policy-options {
  policy-statement fix-nh {
    then {
      next-hop self;
    }
  }
  policy-statement redist-static {
    term a {
      from {
        protocol static;
        route-filter 10.12.1.0/24 exact;
      }
      then accept;
    }
    term b {
      from protocol bgp;
      then accept;
    }
    term c {
      then accept;
    }
  }
  policy-statement vpna-import {
    term a {
      from {
        protocol bgp;
        community vpna-comm;
      }
      then accept;
    }
    term b {
      then reject;
    }
  }
  policy-statement vpna-export {
    term a {
      from protocol bgp;
      then {
        community add vpna-comm;
        accept;
      }
    }
    term b {
      then reject;
    }
  }
  community vpna-comm members target:63000:100;
}
```

Centralized Internet Access

This section describes several ways to configure a CE router to act as a central site for Internet access. Internet traffic from other sites (CE routers) is routed to the hub CE router (which also performs NAT) using that router's VPN interface. The hub CE router then forwards the traffic to a PE router connected to the Internet through another interface identified in the **inet.0** table. The hub CE router can advertise a default route to the spoke CE routers. The disadvantage of this type of configuration is that all traffic has to go through the central CE router before going to the Internet, causing network delays if this router receives too much traffic. However, in a corporate network, traffic might have to be routed to a central site because most corporate networks separate the VPN from the Internet by means of a single firewall.

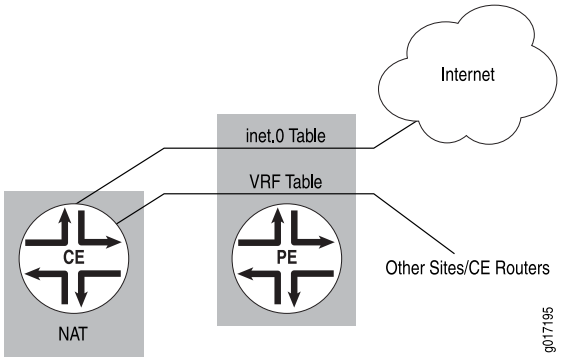
This section includes the following examples:

- Routing Internet Traffic Through a Hub CE Router on page 221
- Routing Internet Traffic Through Multiple CE Routers on page 225

Routing Internet Traffic Through a Hub CE Router

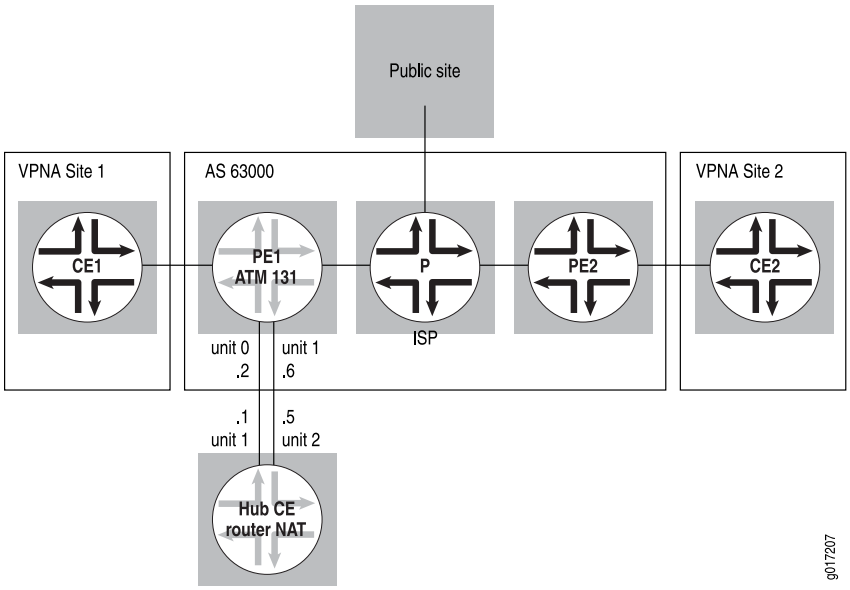
In this example, Internet traffic is routed through a hub CE router. The hub CE router has two interfaces to the hub PE router: a VPN interface and a public interface. It performs NAT on traffic forwarded from the hub PE router through the VPN interface and forwards that traffic from its public interface back to the hub PE router. The hub PE router has a static default route in its VRF table pointing to the hub CE router's VPN interface. It announces this default route to the rest of the VPN, attracting all non-VPN traffic to the hub CE route. The hub PE router also installs and distributes the VPN's public IP address space (see Figure 38 on page 221).

Figure 38: Internet Access Through a Hub CE Router Performing NAT



The configuration for this example is almost identical to that described in "Routing Internet Traffic Through a Separate NAT Device" on page 214. The difference is that Router PE1 is configured to announce a static default route to the other CE routers (see Figure 39 on page 222).

Figure 39: Internet Access Provided Through a Hub CE Router



The following sections show how to configure centralized Internet access by routing Internet traffic through a hub CE router:

- Configuring a Routing Instance on Router PE1 on page 222
- Configuring Policy Options on Router PE1 on page 223
- Internet Traffic Routed by a Hub CE Router: Configuration Summarized by Router on page 224

Configuring a Routing Instance on Router PE1

Configure a routing instance for Router PE1. As part of this configuration, under **routing-options**, configure a default static route (**route 0.0.0.0/0**) to be installed in **vpna.inet.0**, and point the route to the hub CE router's VPN interface (**10.23.0.1**). Also, configure BGP under the routing instance to export the default route to the local CE router:

```
[edit]
routing-instances {
  vpna {
    instance-type vrf;
    interface t3-0/2/0.0;
    interface at-1/3/1.0;
    route-distinguisher 10.255.14.171:100;
    vrf-import vpna-import;
    vrf-export vpna-export;
    routing-options {
      static {
        route 0.0.0.0/0 next-hop 10.23.0.1;
      }
    }
    protocols {
      bgp {
```

```
        group to-CE1 {
          export export-default;
          peer-as 63001;
          neighbor 192.168.197.14;
        }
      }
    }
  }
}
```

Configuring Policy Options on Router PE1

Configure policy options on Router PE1. As part of this configuration, Router PE1 should export the static default route to all the remote PE routers in **vpna** (configured in the **policy-statement vpna-export** statement under **term b**):

```
[edit]
policy-options {
  policy-statement vpna-export {
    term a {
      from protocol bgp;
      then {
        community add vpna-comm;
        accept;
      }
    }
    term b {
      from {
        protocol static;
        route-filter 0.0.0.0/0 exact;
      }
      then {
        community add vpna-comm;
        accept;
      }
    }
    term c {
      then reject;
    }
  }
  policy-statement export-default {
    term a {
      from {
        protocol static;
        route-filter 0.0.0.0/0 exact;
      }
      then accept;
    }
    term b {
      from protocol bgp;
      then accept;
    }
    term c {
      then reject;
    }
  }
```

```
}
```

Internet Traffic Routed by a Hub CE Router: Configuration Summarized by Router

Router PE1

The configuration for Router PE1 is almost identical to that for the example in "Routing Internet Traffic Through a Separate NAT Device" on page 214. The difference is that Router PE1 is configured to announce a static default route to the other CE routers.

Routing Instance

```
routing-instances {
  vpna {
    instance-type vrf;
    interface t3-0/2/0.0;
    interface at-1/3/1.0;
    route-distinguisher 10.255.14.171:100;
    vrf-import vpna-import;
    vrf-export vpna-export;
    routing-options {
      static {
        route 0.0.0.0/0 next-hop 10.23.0.1;
      }
    }
    protocols {
      bgp {
        group to-CE1 {
          export export-default;
          peer-as 63001;
          neighbor 192.168.197.14;
        }
      }
    }
  }
}
```

Policy Options

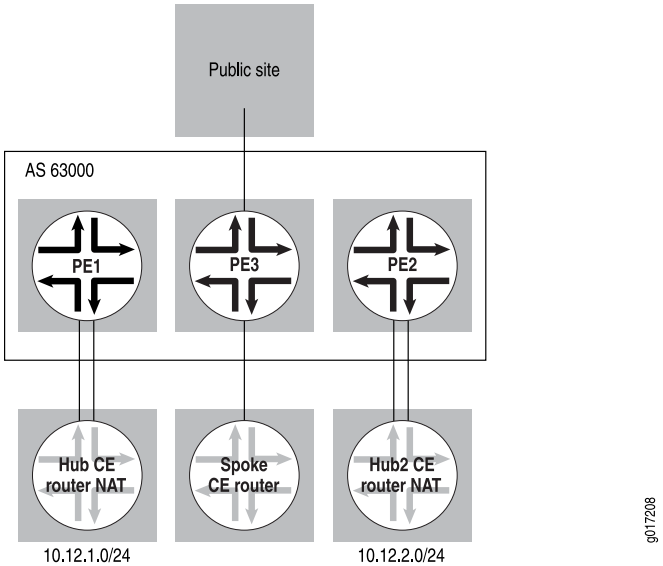
```
policy-options {
  policy-statement vpna-export {
    term a {
      from protocol bgp;
      then {
        community add vpna-comm;
        accept;
      }
    }
    term b {
      from {
        protocol static;
        route-filter 0.0.0.0/0 exact;
      }
      then {
        community add vpna-comm;
        accept;
      }
    }
    term c {
      then reject;
```

```
    }
  }
  policy-statement export-default {
    term a {
      from {
        protocol static;
        route-filter 0.0.0.0/0 exact;
      }
      then accept;
    }
    term b {
      from protocol bgp;
      then accept;
    }
    term c {
      then reject;
    }
  }
}
```

Routing Internet Traffic Through Multiple CE Routers

The example in this section is an extension of that described in "Centralized Internet Access" on page 221. This example provides different exit points for different sites by means of multiple hub CE routers that perform similar functions. Each hub CE router tags the default route with a different route target and allows the spoke CE routers to select the hub site that should be used for Internet access (see Figure 40 on page 225).

Figure 40: Two Hub CE Routers Handling Internet Traffic and NAT



This example uses two hub CE routers that handle NAT and Internet traffic:

- Hub1 CE router tags **0/0** with community **public-comm1** (target: **1:111**)
- Hub2 CE router tags **0/0** with community **public-comm2** (target: **1:112**)

The spoke CE router in this example is configured to have a bias toward Hub2 for Internet access.

The following sections describe how configure two hub CE routers to handle internet traffic and NAT:

Configuring a Routing Instance on Router PE1

Configure a routing instance on Router PE1:

```
[edit]
routing-instances {
  vpna {
    instance-type vrf;
    interface t3-0/2/0.0;
    interface at-1/3/1.0;
    route-distinguisher 10.255.14.171:100;
    vrf-import vpna-import;
    vrf-export vpna-export;
    routing-options {
      static {
        route 0.0.0.0/0 next-hop 10.23.0.1;
      }
    }
    protocols {
      bgp {
        group to-CE1 {
          export export-default;
          peer-as 63001;
          neighbor 192.168.197.14;
        }
      }
    }
  }
}
```

Configuring Policy Options on Router PE1

The policy options for Router PE1 are the same as in "Routing Internet Traffic Through a Hub CE Router" on page 221, but the configuration in this example includes an additional community, **public-comm1**, in the **export** statement:

```
[edit]
policy-options {
  policy-statement vpna-import {
    term a {
      from {
```

```
            protocol bgp;
            community vpna-comm;
          }
          then accept;
        }
        term b {
          then reject;
        }
      }
      policy-statement vpna-export {
        term a {
          from {
            protocol static;
            route-filter 0.0.0.0/0 exact;
          }
          then {
            community add public-comm1;
            community add vpna-comm;
            accept;
          }
        }
        term b {
          from protocol bgp;
          then {
            community add vpna-comm;
            accept;
          }
        }
        term c {
          then reject;
        }
      }
      community public-comm1 members target:1:111;
      community public-comm2 members target:1:112;
      community vpna-comm members target:63000:100;
    }
```

The configuration of Router PE2 is identical to that of Router PE1 except that Router PE2 exports the default route through community **public-comm2**.

Configuring a Routing Instance on Router PE3

Configure routing instance **vpna** on Router PE3:

```
[edit]
routing-instances {
  vpna {
    instance-type vrf;
    interface t1-0/2/0.0;
    route-distinguisher 10.255.14.173:100;
    vrf-import vpna-import;
    vrf-export vpna-export;
    protocols {
      rip {
        group to-vpn12 {
          export export-CE;
```

```
            neighbor t1-0/2/0.0;
          }
        }
      }
    }
  }
```

Configuring Policy Options on Router PE3

Configure the **vrf-import** policy for Router PE3 to select the Internet exit point based on the additional communities specified in "Configuring Policy Options on Router PE1" on page 226:

```
[edit]
policy-options {
  policy-statement vpna-export {
    term a {
      from protocol rip;
      then {
        community add vpna-comm;
        accept;
      }
    }
    term b {
      then reject;
    }
  }
  policy-statement vpna-import {
    term a {
      from {
        protocol bgp;
        community public-comm1;
        route-filter 0.0.0.0/0 exact;
      }
      then reject;
    }
    term b {
      from {
        protocol bgp;
        community vpna-comm;
      }
      then accept;
    }
    term c {
      then reject;
    }
  }
  policy-statement export-CE {
    from protocol bgp;
    then accept;
  }
  community vpna-comm members target:69:100;
  community public-comm1 members target:1:111;
  community public-comm2 members target:1:112;
}
```

Routing Internet Traffic Through Multiple CE Routers: Configuration Summarized by Router

Router PE1

This configuration is an extension of the example in "Routing Internet Traffic Through a Hub CE Router" on page 221. It provides different exit points for various sites by using multiple hub CE routers that perform similar functions.

Routing Instances

```
routing-instances {
  vpna {
    instance-type vrf;
    interface t3-0/2/0.0;
    interface at-1/3/1.0;
    route-distinguisher 10.255.14.171:100;
    vrf-import vpna-import;
    vrf-export vpna-export;
    routing-options {
      static {
        route 0.0.0.0/0 next-hop 10.23.0.1;
      }
    }
    protocols {
      bgp {
        group to-CE1 {
          export export-default;
          peer-as 63001;
          neighbor 192.168.197.14;
        }
      }
    }
  }
}
```

Policy Options

```
policy-options {
  policy-statement vpna-import {
    term a {
      from {
        protocol bgp;
        community vpna-comm;
      }
      then accept;
    }
    term b {
      then reject;
    }
  }
  policy-statement vpna-export {
    term a {
      from {
        protocol static;
        route-filter 0.0.0.0/0 exact;
      }
      then {
        community add public-comm1;
```

```
            community add vpna-comm;
            accept;
          }
        }
        term b {
          from protocol bgp;
          then {
            community add vpna-comm;
            accept;
          }
        }
        term c {
          then reject;
        }
      }
      community public-comm1 members target:1:111;
      community public-comm2 members target:1:112;
      community vpna-comm members target:63000:100;
    }
```

Router PE2

The configuration of Router PE2 is identical to that of Router PE1, except that Router PE2 exports the default route through community **public-comm2** (see "Policy Options" on page 229).

Router PE3

Routing Instances

```
routing-instances {
  vpna {
    instance-type vrf;
    interface t1-0/2/0.0;
    route-distinguisher 10.255.14.173:100;
    vrf-import vpna-import;
    vrf-export vpna-export;
    protocols {
      rip {
        group to-vpn12 {
          export export-CE;
          neighbor t1-0/2/0.0;
        }
      }
    }
  }
}
```

Policy Options

```
policy-options {
  policy-statement vpna-export {
    term a {
      from protocol rip;
      then {
        community add vpna-comm;
        accept;
      }
    }
    term b {
```

```
        then reject;
      }
    }
    policy-statement vpna-import {
      term a {
        from {
          protocol bgp;
          community public-comm1;
          route-filter 0.0.0.0/0 exact;
        }
        then reject;
      }
      term b {
        from {
          protocol bgp;
          community vpna-comm;
        }
        then accept;
      }
      term c {
        then reject;
      }
    }
    policy-statement export-CE {
      from protocol bgp;
      then accept;
    }
    community vpna-comm members target:69:100;
    community public-comm1 members target:1:111;
    community public-comm2 members target:1:112;
}
```


CHAPTER 6

Additional Examples

Example: Configuring Interprovider Layer 3 VPN Option A

This example provides a step-by-step procedure to configure interprovider Layer 3 VPN option A, which is one of the recommended implementations of MPLS VPN when that service is required by a customer that has more than one AS and but not all of the customer's ASs can be serviced by the same service provider. It is organized in the following sections:

Requirements

This example uses the following hardware and software components:

- Junos OS Release 9.5 or later.
- Eight M Series, T Series, TX Series, or MX Series Juniper Networks routers.

Configuration Overview and Topology

This is the simplest and least scalable interprovider VPN solution to the problem of providing VPN services to a customer that has different sites, not all of which can use the same service provider (SP).

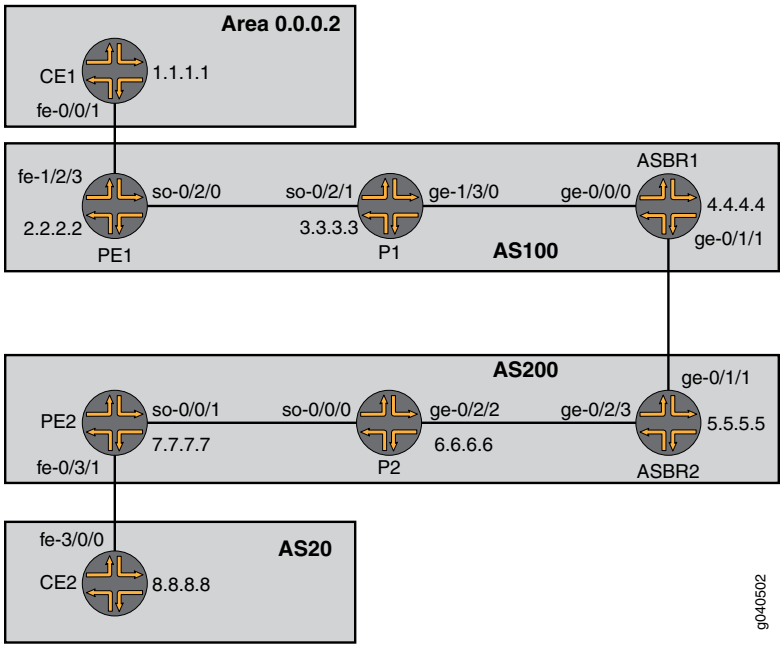
RFC 4364, section 10, refers to this method as Interprovider VRF-to-VRF connections at the AS border routers.

In this configuration:

- The VPN routing and forwarding (VRF) table in the ASBR of one AS is linked to the VRF table in the ASBR in the other AS. Each ASBR must contain a VRF instance for every VPN configured in both service provider networks. Then an IGP or BGP must be configured between the ASBRs. This has the disadvantage of limiting scalability.
- In this configuration, the autonomous system boundary routers (ASBRs) at both SPs are configured as regular PE routers, and provide MPLS L3 VPN service to the neighbor SP.
- Each PE router treats the other as if it were a customer edge (CE) router. ASBRs play the role of regular CE routers for the ASBR of the remote SP. ASBRs see each other as CE devices.
- A provider edge (PE) router in one autonomous system (AS) attaches directly to a PE router in another AS.
- The two PE routers are attached by multiple sub-interfaces, at least one for each of the VPNs whose routes need to be passed from AS to AS.
- The PE routers associate each sub-interface with a VPN routing and forwarding (VRF) table, and use EBGP to distribute unlabeled IPv4 addresses to each other.
- In this solution, all common VPNs defined at both PEs must also be defined at one or more ASBRs between the two SPs. This is not a very scalable methodology, especially when a transit SP is used by two regional SPs for interconnection.
- This is a procedure that is simple to configure and it does not require MPLS at the border between ASs. Additionally, it does not scale as well as other recommended procedures.

The topology of the network is shown in Figure 41 on page 235.

Figure 41: Physical Topology of Interprovider Layer 3 VPN Option A



Configuration

NOTE: The procedure presented here is written with the assumption that the reader is already familiar with MPLS MVPN configuration. This example focuses on explaining the unique configuration required for carrier-of-carriers solutions for VPN services to different sites.

To configure interprovider layer 3 VPN option A, perform the following tasks:

Configuring Router CE1

Step-by-Step Procedure

1. On Router CE1, configure the IP address and protocol family on the Fast Ethernet interface for the link between Router CE1 and Router PE1. Specify the **inet** address family type.

```
[edit interfaces fe-0/0/1.0]
family inet {
  address 18.18.18.1/30;
}
```

2. On Router CE1, configure the IP address and protocol family on the loopback interface. Specify the **inet** address family type.

```
[edit interfaces lo0]
unit 0 {
  family inet {
    address 1.1.1.1/32;
  }
}
```

3. On Router CE1, configure an IGP. The IGP can be a static route, RIP, OSPF, ISIS, or EBGP. In this example we configure OSPF. Include the Fast Ethernet interface for the link between Router CE1 and Router PE1 and the logical loopback interface of Router CE1.

```
[edit protocols]
ospf {
  area 0.0.0.2 {
    interface fe-0/0/1.0;
    interface lo0.0;
  }
}
```

Configuring Router PE1

Step-by-Step Procedure

1. On Router PE1, configure IPv4 addresses on the SONET, Fast Ethernet, and logical loopback interfaces. Specify the **inet** address family on all of the interfaces. Specify the **mpls** address family on the SONET and Fast Ethernet interfaces.

```
[edit interfaces]
so-0/2/0 {
  unit 0 {
    family inet {
      address 19.19.19.1/30;
    }
    family mpls;
  }
}
fe-1/2/3 {
  unit 0 {
    family inet {
      address 18.18.18.2/30;
    }
    family mpls;
  }
}
lo0 {
  unit 0 {
    family inet {
      address 2.2.2.2/32;
    }
```

```
    }
  }
```

2. On Router PE1, configure the routing instance for VPN2. Specify the **vrf** instance type and specify the customer-facing Fast Ethernet interface. Configure a route distinguisher to create a unique VPN-IPv4 address prefix. Apply the VRF import and export policies to enable the sending and receiving of route targets. Configure the OSPF protocol within the VRF. Specify the customer-facing Fast Ethernet interface and specify the export policy to export BGP routes into OSPF.

```
[edit routing-instances]
vpn2CE1 {
  instance-type vrf;
  interface fe-1/2/3.0;
  route-distinguisher 1:100;
  vrf-import vpnimport;
  vrf-export vpnexport;
  protocols {
    ospf {
      export bgp-to-ospf;
      area 0.0.0.2 {
        interface fe-1/2/3.0;
      }
    }
  }
}
```

3. On Router PE1, configure the RSVP and MPLS protocols to support the label-switched path (LSP). Configure the LSP to Router ASBR1 and specify the IP address of the logical loopback interface on Router ASBR1. Configure a BGP group. Specify the group type as **internal**. Specify the local address as the logical loopback interface on Router PE1. Specify the neighbor address as the logical loopback interface on Router ASBR1. Specify the **inet-vpn** address family and **unicast** traffic type to enable BGP to carry IPv4 network layer reachability information (NLRI) for VPN routes. Configure the OSPF protocol. Specify the core-facing SONET interface and specify the logical loopback interface on Router PE1.

```
[edit protocols]
rsvp {
  interface so-0/2/0.0;
  interface lo0.0;
}
mpls {
  label-switched-path To-ASBR1 {
    to 4.4.4.4;
  }
  interface so-0/2/0.0;
  interface lo0.0;
}
bgp {
  group To_ASBR1 {
    type internal;
    local-address 2.2.2.2;
    neighbor 4.4.4.4 {
      family inet-vpn {
```

```
            unicast;
          }
        }
      }
    }
    ospf {
      traffic-engineering;
      area 0.0.0.0 {
        interface so-0/2/0.0;
        interface lo0.0;
      }
    }
```

4. On Router PE1, configure the BGP local autonomous system number.

```
[edit routing-options]
autonomous-system 100;
```

5. On Router PE1, configure a policy to export the BGP routes into OSPF.

```
[edit policy-options]
policy-statement bgp-to-ospf {
  term 1 {
    from protocol bgp;
    then accept;
  }
  term 2 {
    then reject;
  }
}
```

6. On Router PE1, configure a policy to add the VRF route target to the routes being advertised for this VPN.

```
[edit policy-options]
policy-statement vpnexport {
  term 1 {
    from protocol ospf;
    then {
      community add test_comm;
      accept;
    }
  }
  term 2 {
    then reject;
  }
}
```

7. On Router PE1, configure a policy to import routes from BGP that have the **test_comm** community attached.

```
[edit policy-options]
policy-statement vpnimport {
  term 1 {
    from {
      protocol bgp;
      community test_comm;
    }
```

```
        then accept;
    }
    term 2 {
        then reject;
    }
}
```

8. On Router PE1, define the **test_comm** BGP community with a route target.

```
[edit policy-options]
community test_comm members target:1:100;
```

Configuring Router P1

Step-by-Step Procedure

1. On Router P1, configure IP addresses for the SONET and Gigabit Ethernet interfaces. Enable the interfaces to process the **inet** and **mpls** address families. Configure the IP address for the **lo0.0** loopback interface and enable the interface to process the **inet** address family.

```
[edit interfaces]
so-0/2/1 {
    unit 0 {
        family inet {
            address 19.19.19.2/30;
        }
        family mpls;
    }
}
ge-1/3/0 {
    unit 0 {
        family inet {
            address 20.20.20.1/30;
        }
        family mpls;
    }
}
lo0 {
    unit 0 {
        family inet {
            address 3.3.3.3/32;
        }
    }
}
```

2. On Router P1, configure the RSVP and MPLS protocols to support the LSP. Specify the SONET and Gigabit Ethernet interfaces.

   Configure the OSPF protocol. Specify the SONET and Gigabit Ethernet interfaces and specify the logical loopback interface. Enable OSPF to support traffic engineering extensions.

```
[edit protocols]
rsvp {
    interface so-0/2/1.0;
    interface ge-1/3/0.0;
    interface lo0.0;
```

```
}
mpls {
  interface lo0.0;
  interface ge-1/3/0.0;
  interface so-0/2/1.0;
}
ospf {
  traffic-engineering;
  area 0.0.0.0 {
    interface ge-1/3/0.0;
    interface so-0/2/1.0;
    interface lo0.0;
  }
}
```

Configuring Router ASBR1

Step-by-Step Procedure

1. On Router ASBR1, configure IP addresses for the Gigabit Ethernet interfaces. Enable the interfaces to process the **inet** and **mpls** addresses families. Configure the IP addresses for the **lo0.0** loopback interface and enable the interface to process the **inet** address family.

```
[edit interfaces]
ge-0/0/0 {
  unit 0 {
    family inet {
      address 20.20.20.2/30;
    }
    family mpls;
  }
}
ge-0/1/1 {
  unit 0 {
    family inet {
      address 21.21.21.1/30;
    }
    family mpls;
  }
}
lo0 {
  unit 0 {
    family inet {
      address 4.4.4.4/32;
    }
  }
}
```

2. On Router ASBR1, configure the **To_ASBR2** routing instance. Specify the **vrf** instance type and specify the core-facing Gigabit Ethernet interface. Configure a route distinguisher to create a unique VPN-IPv4 address prefix. Configure a route target for the VPN. Configure the BGP peer group within the VRF. Specify AS 200 as the peer AS and specify the IP address of the Gigabit Ethernet interface on Router ASBR2 as the neighbor address.

```
[edit routing instances]
```

```
To_ASBR2{
  instance-type vrf;
  interface ge-0/1/1.0;
  route-distinguisher 1:100;
  vrf-target target:1:100;
  protocols {
    bgp {
      group To_ASBR2 {
        type external;
        neighbor 21.21.21.2 {
          peer-as 200;
        }
      }
    }
  }
}
```

3. On Router ASBR1, configure the RSVP and MPLS protocols to support the LSP. Specify the Gigabit Ethernet interfaces.

   Configure the OSPF protocol. Specify the SONET and Gigabit Ethernet interfaces and specify the logical loopback interface. Enable OSPF to support traffic engineering extensions.

```
[edit protocols]
rsvp {
  interface ge-0/0/0.0;
  interface lo0.0;
}
mpls {
  label-switched-path To_PE1 {
    to 2.2.2.2;
  }
  interface lo0.0;
  interface ge-0/0/0.0;
}
ospf {
  traffic-engineering;
  area 0.0.0.0 {
    interface ge-0/0/0.0;
    interface lo0.0;
  }
}
```

4. On Router ASBR1, create the **To-PE1** internal BGP peer group. Specify the local IP peer address as the local **lo0.0** address. Specify the neighbor IP peer address as the **lo0.0** interface address of Router PE1.

```
[edit protocols]
bgp {
  group To-PE1 {
    type internal;
    local-address 4.4.4.4;
    neighbor 2.2.2.2 {
      family inet-vpn {
        unicast;
      }
```

```
    }
  }
}
```

5. On Router ASBR1, configure the BGP local autonomous system number.

```
[edit routing-options]
autonomous-system 100;
```

Configuring Router ASBR2

Step-by-Step Procedure

1. On Router ASBR2, configure IP addresses for the Gigabit Ethernet interfaces. Enable the interfaces to process the **inet** and **mpls** address families. Configure the IP address for the **lo0.0** loopback interface and enable the interface to process the **inet** address family.

```
[edit interfaces]
ge-0/1/1 {
  unit 0 {
    family inet {
      address 21.21.21.2/30;
    }
    family mpls;
  }
}
ge-0/2/3 {
  unit 0 {
    family inet {
      address 22.22.22.1/30;
    }
    family mpls;
  }
}
lo0 {
  unit 0 {
    family inet {
      address 5.5.5.5/32;
    }
  }
}
```

2. On Router ASBR2, configure the **To_ASBR1** routing instance. Specify the **vrf** instance type and specify the core-facing Gigabit Ethernet interface. Configure a route distinguisher to create a unique VPN-IPv4 address prefix. Configure a route target for the VPN. Configure the BGP peer group within the VRF. Specify AS 100 as the peer AS and specify the IP address of the Gigabit Ethernet interface on Router ASBR1 as the neighbor address.

```
[edit routing-instances]
To_ASBR1 {
  instance-type vrf;
  interface ge-0/1/1.0;
  route-distinguisher 1:100;
  vrf-target target:1:100;
  protocols {
    bgp {
```

```
        group To_ASBR1 {
          type external;
          neighbor 21.21.21.1 {
            peer-as 100;
          }
        }
      }
    }
  }
```

3. On Router ASBR2, configure the RSVP and MPLS protocols to support the LSP. Specify the Gigabit Ethernet interfaces.

   Configure the OSPF protocol. Specify the SONET and Gigabit Ethernet interfaces and specify the logical loopback interface. Enable OSPF to support traffic engineering extensions.

```
[edit protocols]
rsvp {
  interface ge-0/2/3.0;
  interface lo0.0;
}
mpls {
  label-switched-path To_PE2 {
    to 7.7.7.7;
  }
  interface lo0.0;
  interface ge-0/2/3.0;
}
ospf {
  traffic-engineering;
  area 0.0.0.0 {
    interface ge-0/2/3.0;
    interface lo0.0;
  }
}
```

4. On Router ASBR2, create the **To-PE2** internal BGP peer group. Specify the local IP peer address as the local **lo0.0** address. Specify the neighbor IP peer address as the **lo0.0** interface address of Router PE2.

```
[edit protocols]
bgp {
  group To-PE2 {
    type internal;
    local-address 5.5.5.5;
    neighbor 7.7.7.7 {
      family inet-vpn {
        unicast;
      }
    }
  }
```

5. On Router ASBR2, configure the BGP local autonomous system number.

```
[edit routing-options]
autonomous-system 200;
```

Configuring Router P2

Step-by-Step Procedure

1. On Router P2, configure IP addresses for the SONET and Gigabit Ethernet interfaces. Enable the interfaces to process the **inet** and **mpls** address families. Configure the IP address for the **lo0.0** loopback interface and enable the interface to process the **inet** address family.

```
[edit interfaces]
so-0/0/0 {
  unit 0 {
    family inet {
      address 23.23.23.1/30;
    }
    family mpls;
  }
}
ge-0/2/2 {
  unit 0 {
    family inet {
      address 22.22.22.2/30;
    }
    family mpls;
  }
}
lo0 {
  unit 0 {
    family inet {
      address 6.6.6.6/32;
    }
  }
}
```

2. On Router P2, configure the RSVP and MPLS protocols to support the LSP. Specify the SONET and Gigabit Ethernet interfaces.

   Configure the OSPF protocol. Specify the SONET and Gigabit Ethernet interfaces and specify the logical loopback interface. Enable OSPF to support traffic engineering extensions.

```
[edit protocols]
rsvp {
  interface so-0/0/0.0;
  interface ge-0/2/2.0;
  interface lo0.0;
}
mpls {
  interface lo0.0;
  interface ge-0/2/2.0;
  interface so-0/0/0.0;
}
ospf {
  traffic-engineering;
  area 0.0.0.0 {
    interface ge-0/2/2.0;
    interface so-0/0/0.0;
    interface lo0.0;
```

```
    }
  }
```

Configuring Router PE2

Step-by-Step Procedure

1. On Router PE2, configure IPv4 addresses on the SONET, Fast Ethernet, and logical loopback interfaces. Specify the **inet** address family on all of the interfaces. Specify the **mpls** address family on the SONET and Fast Ethernet interfaces.

```
[edit interfaces]
so-0/0/1 {
  unit 0 {
    family inet {
      address 23.23.23.2/30;
    }
    family mpls;
  }
}
fe-0/3/1 {
  unit 0 {
    family inet {
      address 24.24.24.1/30;
    }
    family mpls;
  }
lo0 {
  unit 0 {
    family inet {
      address 7.7.7.7/32;
    }
  }
}
```

2. On Router PE2, configure the routing instance for VPN2. Specify the **vrf** instance type and specify the customer-facing Fast Ethernet interface. Configure a route distinguisher to create a unique VPN-IPv4 address prefix. Apply the VRF import and export policies to enable the sending and receiving of route targets. Configure the BGP peer group within the VRF. Specify AS **20** as the peer AS and specify the IP address of the Fast Ethernet interface on Router CE2 as the neighbor address.

```
[edit routing-instances]
vpn2CE2 {
  instance-type vrf;
  interface fe-0/3/1.0;
  route-distinguisher 1:100;
  vrf-import vpnimport;
  vrf-export vpnexport;
  protocols {
    bgp {
      group To_CE2 {
        peer-as 20;
        neighbor 24.24.24.2;
      }
    }
  }
```

```
}
```

3. On Router PE2, configure the RSVP and MPLS protocols to support the LSP. Configure the LSP to ASBR2 and specify the IP address of the logical loopback interface on Router ASBR2. Configure a BGP group. Specify the group type as **internal**. Specify the local address as the logical loopback interface on Router PE2. Specify the neighbor address as the logical loopback interface on the Router ASBR2. Specify the **inet-vpn** address family and **unicast** traffic type to enable BGP to carry IPv4 NLRI for VPN routes. Configure the OSPF protocol. Specify the core-facing SONET interface and specify the logical loopback interface on Router PE2.

```
[edit protocols]
rsvp {
   interface so-0/0/1.0;
   interface lo0.0;
}
mpls {
   label-switched-path To-ASBR2 {
      to 5.5.5.5;
   }
   interface so-0/0/1.0;
   interface lo0.0;
}
bgp {
   group To_ASBR2 {
      type internal;
      local-address 7.7.7.7;
      neighbor 5.5.5.5 {
         family inet-vpn {
            unicast;
         }
      }
   }
}
ospf {
   traffic-engineering;
   area 0.0.0.0 {
      interface so-0/0/1.0;
      interface lo0.0;
   }
}
```

4. On Router PE2, configure the BGP local autonomous system number.

```
[edit routing-options]
autonomous-system 200;
```

5. On Router PE2, configure a policy to add the VRF route target to the routes being advertised for this VPN.

```
[edit policy-options]
policy-statement vpnexport {
   term 1 {
      from protocol bgp;
      then {
         community add test_comm;
         accept;
```

```
    }
  }
  term 2 {
    then reject;
  }
}
```

6. On Router PE2, configure a policy to import routes from BGP that have the **test_comm** community attached.

```
[edit policy-options]
  policy-statement vpnimport {
    term 1 {
      from {
        protocol bgp;
        community test_comm;
      }
      then accept;
    }
    term 2 {
      then reject;
    }
  }
```

7. On Router PE2, define the **test_comm** BGP community with a route target.

```
[edit policy-options]
community test_comm members target:1:100;
```

Configuring Router CE2

Step-by-Step Procedure

1. On Router CE2, configure the IP address and protocol family on the Fast Ethernet interface for the link between Router CE2 and Router PE2. Specify the **inet** address family type.

```
[edit interfaces]
fe-3/0/0 {
  unit 0 {
    family inet {
      address 24.24.24.2/30;
    }
  }
}
```

2. On Router CE2, configure the IP address and protocol family on the loopback interface. Specify the **inet** address family type.

```
[edit interfaces lo0]
lo0 {
  unit 0 {
    family inet {
      address 8.8.8.8/32;
    }
  }
}
```

3. On Router CE2, configure an IGP. The IGP can be a static route, RIP, OSPF, ISIS, or EBGP. In this example, we configure EBGP. Specify AS **200** as the peer AS and specify the BGP neighbor IP address as the Fast Ethernet interface of Router PE2.

```
[edit protocols]
bgp {
  group To_PE2 {
    neighbor 24.24.24.1 {
      export myroutes;
      peer-as 200;
    }
  }
}
```

Verifying the VPN Operation

Step-by-Step Procedure

1. Commit the configuration on each router.

NOTE: The MPLS labels shown in this example will be different than the labels used in your configuration.

2. On Router PE1, display the routes for the **vpn2CE1** routing instance using the **show ospf route** command. Verify that the **1.1.1.1** route is learned from OSPF.

```
user@PE1> show ospf route instance vpn2CE1

Topology default Route Table:

Prefix             Path   Route       NH   Metric  NextHop       Nexthop
                  Type   Type        Type        Interface     addr/label
1.1.1.1          Intra  Router      IP        1  fe-1/2/3.0    18.18.18.1
1.1.1.1/32       Intra  Network     IP        1  fe-1/2/3.0    18.18.18.1
18.18.18.0/30    Intra  Network     IP        1  fe-1/2/3.0    18.18.18.1
```

3. On Router PE1, use the **show route advertising-protocol** command to verify that Router PE1 advertises the **1.1.1.1** route to Router ASBR1 using MP-BGP with the VPN MPLS label.

```
user@PE1> show route advertising-protocol bgp 4.4.4.4 extensive

vpn2CE1.inet.0: 6 destinations, 6 routes (6 active, 0 holddown, 0 hidden)
* 1.1.1.1/32 (1 entry, 1 announced)
 BGP group To_PE1 type Internal
     Route Distinguisher: 1:100
     VPN Label: 299856
     Nexthop: Self
     Flags: Nexthop Change
     MED: 1
     Localpref: 100
     AS path: [100] I
     Communities: target:1:100 rte-type:0.0.0.2:1:0
```

4. On Router ASBR1, use the **show route receive-protocol** command to verify that the router receives and accepts the **1.1.1.1** route and places it in the **To_ASBR2.inet.0** routing table.

```
user@ASBR1> show route receive-protocol bgp 2.2.2.2 extensive

inet.0: 7 destinations, 7 routes (7 active, 0 holddown, 0 hidden)

inet.3: 1 destinations, 1 routes (1 active, 0 holddown, 0 hidden)

To_ASBR2.inet.0: 4 destinations, 4 routes (4 active, 0 holddown, 0 hidden)
* 1.1.1.1/32 (1 entry, 1 announced)
     Route Distinguisher: 1:100
     VPN Label: 299856
     Nexthop: 2.2.2.2
     MED: 1
     Localpref: 100
     AS path: I
     Communities: target:1:100 rte-type:0.0.0.2:1:0

MPLS.0: 4 destinations, 4 routes (4 active, 0 holddown, 0 hidden)

BGP.13VPN.0: 1 destinations, 1 routes (1 active, 0 holddown, 0 hidden)

* 1:100:1.1.1.1/32 (1 entry, 0 announced)
     Route Distinguisher: 1:100
     VPN Label: 299856
     Nexthop: 2.2.2.2
     MED: 1
     Localpref: 100
     AS path: I
     Communities: target:1:100 rte-type:0.0.0.2:1:0
```

5. On Router ASBR1, use the **show route advertising-protocol** command to verify that Router ASBR1 advertises the **1.1.1.1** route to Router ASBR2.

```
user@ASBR1> show route advertising-protocol bgp 21.21.21.2 extensive

To_ASBR2.inet.0: 4 destinations, 4 routes (4 active, 0 holddown, 0 hidden)
* 1.1.1.1/32 (1 entry, 1 announced)
 BGP group To_ASBR2.inet.0 type External
     Nexthop: Self
     AS path: [100] I
     Communities: target:1:100 rte-type:0.0.0.2:1:0
```

6. On Router ASBR2, use the **show route receive-protocol** command to verify that the router receives and accepts the **1.1.1.1** route and places it in the **To_ASBR1.inet.0** routing table.

```
user@ASBR2> show route receive-protocol bgp 21.21.21.1 extensive

inet.0: 7 destinations, 7 routes (7 active, 0 holddown, 0 hidden)

inet.3: 1 destinations, 1 routes (1 active, 0 holddown, 0 hidden)

To_ASBR1.inet.0: 4 destinations, 4 routes (4 active, 0 holddown, 0 hidden)
* 1.1.1.1/32 (1 entry, 1 announced)
     Accepted
     Nexthop: 21.21.21.1
     AS path: 100 I
     Communities: target:1:100 rte-type:0.0.0.2:1:0
```

```
MPLS.0: 4 destinations, 4 routes (4 active, 0 holddown, 0 hidden)

BGP.l3VPN.0: 1 destinations, 1 routes (1 active, 0 holddown, 0 hidden)
```

7. On Router ASBR2, use the **show route advertising-protocol** command to verify that Router ASBR2 advertises the **1.1.1.1** route to Router PE2 in the **To-PE2** routing instance.

```
user@ASBR2> show route advertising-protocol bgp 7.7.7.7 extensive

To_ASBR1.inet.0: 4 destinations, 4 routes (4 active, 0 holddown, 0 hidden)
* 1.1.1.1/32 (1 entry, 1 announced)
 BGP group To-PE2 type Internal
     Route Distinguisher: 1:100
     VPN Label: 299936
     Nexthop: Self
     Flags: Nexthop Change
     Localpref: 100
     AS path: [200] 100 I
     Communities: target:1:100 rte-type:0.0.0.2:1:0
```

8. On Router PE2, use the **show route receive-protocol** command to verify that the router receives and accepts the **1.1.1.1** route and places it in the **To_CE2.inet.0** routing table.

```
user@PE2> show route receive-protocol bgp 5.5.5.5 extensive

inet.0: 12 destinations, 13 routes (12 active, 0 holddown, 0 hidden)

inet.3: 1 destinations, 1 routes (1 active, 0 holddown, 0 hidden)

__juniper_private1__.inet.0: 14 destinations, 14 routes (8 active, 0
holddown, 6 hidden)

__juniper_private2__.inet.0: 1 destinations, 1 routes (0 active, 0
holddown, 1 hidden)

To_CE2.inet.0: 5 destinations, 6 routes (5 active, 0 holddown, 0 hidden)
* 1.1.1.1/32 (1 entry, 1 announced)
     Accepted
     Route Distinguisher: 1:100
     VPN Label: 299936
     Nexthop: 5.5.5.5
     Localpref: 100
     AS path: 100 I
     AS path: Recorded
     Communities: target:1:100 rte-type:0.0.0.2:1:0
```

9. On Router PE2, use the **show route advertising-protocol** command to verify that Router PE2 advertises the **1.1.1.1** route to Router CE2 through the **To_CE2** peer group.

```
user@PE2> show route advertising-protocol bgp 24.24.24.2 extensive

To_CE2.0: 5 destinations, 6 routes (5 active, 0 holddown, 0 hidden)
* 1.1.1.1/32 (1 entry, 1 announced)
 BGP group To_CE2 type External
     Nexthop: Self
     AS path: [200] 100 I
     Communities: target:1:100 rte-type:0.0.0.2:1:0
```


10. On Router CE2, use the **show route** command to verify that Router CE2 receives the **1.1.1.1** route from Router PE2.

```
user@CE2> show route 1.1.1.1

inet.0: 6 destinations, 6 routes (6 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

1.1.1.1/32         *[BGP/170] 00:25:36, localpref 100
                      AS path: 200 100 I
                    > to 24.24.24.1 via fe-3/0/0.0
```

11. On Router CE2, use the **ping** command and specify **8.8.8.8** as the source of the ping packets to verify connectivity with Router CE1.

```
user@CE2> ping 1.1.1.1 source 8.8.8.8

PING 1.1.1.1 (1.1.1.1): 56 data bytes
64 bytes from 1.1.1.1: icmp_seq=0 ttl=58 time=4.672 ms
64 bytes from 1.1.1.1: icmp_seq=1 ttl=58 time=10.480 ms
64 bytes from 1.1.1.1: icmp_seq=2 ttl=58 time=10.560 ms
```

12. On Router PE2, use the **show route** command to verify that the traffic is sent with an inner label of **299936** and a top label of **299776**.

```
user@PE2> show route 1.1.1.1 detail


To_CE2.inet.0: 5 destinations, 6 routes (5 active, 0 holddown, 0 hidden)
1.1.1.1/32 (1 entry, 1 announced)
        *BGP    Preference: 170/-101
                Route Distinguisher: 1:100
                Next hop type: Indirect
                Next-hop reference count: 6
                Source: 5.5.5.5
                Next hop type: Router, Next hop index: 648
                Next hop: via so-0/0/1.0 weight 0x1, selected
                Label-switched-path To-ASBR2
                Label operation: Push 299936, Push 299776(top)
                Protocol next hop: 5.5.5.5
                Push 299984
                Indirect next hop: 8c6109c 262143
                State: <Secondary Active Int Ext>
                Local AS:   200 Peer AS:   200
                Age: 3:37       Metric2: 2
                Task: BGP_200.5.5.5.5+179
                Announcement bits (3): 0-RT 1-KRT 2-BGP RT Background
                AS path: 100 I
                AS path: Recorded
                Communities: target:1:100 rte-type:0.0.0.2:1:0
                Accepted
                VPN Label: 299984
                Localpref: 100
                Router ID: 5.5.5.5
                Primary Routing Table BGP.l3VPN.0
```

13. On Router ASBR2, use the **show route table** command to verify that Router ASBR2 receives the traffic.

```
lab@ASBR2# show route table mpls.0 detail
```

```
299936 (1 entry, 1 announced)
        *VPN    Preference: 170
                Next hop type: Router, Next hop index: 649
                Next-hop reference count: 2
                Source: 21.21.21.1
                Next hop: 21.21.21.1 via ge-0/1/1.0, selected
                Label operation: Pop
                State: <Active Int Ext>
                Local AS:   200
                Age: 9:54
                Task: BGP RT Background
                Announcement bits (1): 0-KRT
                AS path: 100 I
                Ref Cnt: 1
                Communities: target:1:100 rte-type:0.0.0.2:1:0
```

14. On Router ASBR2, use the **show route table** command to verify that Router ASBR2 receives the traffic.

```
lab@ASBR2# show route 1.1.1.1 detail

To_ASBR1.inet.0: 4 destinations, 4 routes (4 active, 0 holddown, 0 hidden)
1.1.1.1/32 (1 entry, 1 announced)
        *BGP    Preference: 170/-101
                Next hop type: Router, Next hop index: 576
                Next-hop reference count: 3
                Source: 21.21.21.1
                Next hop: 21.21.21.1 via ge-0/1/1.0, selected
                State: <Active Ext>
                Peer AS:   100
                Age: 13:07
                Task: BGP_100.21.21.21.1+53372
                Announcement bits (2): 0-KRT 1-BGP RT Background
                AS path: 100 I
                Communities: target:1:100 rte-type:0.0.0.2:1:0
                Accepted
                Localpref: 100
                Router ID: 21.21.21.1
```

15. On Router ASBR1, use the **show route** command to verify that ASBR1 sends traffic toward PE1 with the top label **299792** and VPN label **299856**.

```
lab@ASBR1# show route 1.1.1.1 detail

To_ASBR2.inet.0: 4 destinations, 4 routes (4 active, 0 holddown, 0 hidden)
1.1.1.1/32 (1 entry, 1 announced)
        *BGP    Preference: 170/-101
                Route Distinguisher: 1:100
                Next hop type: Indirect
                Next-hop reference count: 3
                Source: 2.2.2.2
                Next hop type: Router, Next hop index: 669
                Next hop: 20.20.20.1 via ge-0/0/0.0 weight 0x1, selected
                Label-switched-path To_PE1
                Label operation: Push 299856, Push 299792(top)
                Protocol next hop: 2.2.2.2               Push 299856
                Indirect next hop: 8af70a0 262143
                State: <Secondary Active Int Ext>
                Local AS:   100 Peer AS:   100
                Age: 12:15      Metric: 1       Metric2: 2
                Task: BGP_100.2.2.2.2+58065
```



```
                    Announcement bits (2): 0-KRT 1-BGP RT Background
                    AS path: I
                    Communities: target:1:100 rte-type:0.0.0.2:1:0
                    VPN Label: 299856
                    Localpref: 100
                    Router ID: 2.2.2.2
                    Primary Routing Table BGP.l3VPN.0
```

16. On Router PE1, use the **show route table** command to verify that Router PE1 receives the traffic with label **299856**, pops the label,l and the traffic is sent toward Router CE1 through interface **fe-1/2/3.0**.

```
lab@PE1# show route table mpls.0 detail

299856 (1 entry, 1 announced)
        *VPN    Preference: 170
                Next hop type: Router, Next hop index: 666
                Next-hop reference count: 2
                Next hop: 18.18.18.1 via fe-1/2/3.0, selected
                Label operation: Pop
                State: <Active Int Ext>
                Local AS:   100
                Age: 17:38
                Task: BGP RT Background
                Announcement bits (1): 0-KRT
                AS path: I
                Ref Cnt: 1
                Communities: rte-type:0.0.0.2:1:0
```

17. On Router PE1, use the **show route** command to verify that PE1 receives the traffic after the top label is popped by Router P and the traffic is sent toward Router CE1 through interface **fe-1/2/3.0**.

```
lab@PE1# show route 1.1.1.1 detail


 vpn2CE1.inet.0: 6 destinations, 6 routes (6 active, 0 holddown, 0 hidden)
1.1.1.1/32 (1 entry, 1 announced)
        *OSPF   Preference: 10
                Next hop type: Router, Next hop index: 634
                Next-hop reference count: 3
                Next hop: 18.18.18.1 via fe-1/2/3.0, selected
                State: <Active Int>
                Age: 18:42      Metric: 1
                Area: 0.0.0.2
                Task: VPN2alice-OSPFv2
                Announcement bits (2): 2-KRT 3-BGP RT Background
                AS path: I
                Communities: rte-type:0.0.0.2:1:0
```

Related Documentation

- Interprovider Layer 3 VPN Option A Overview

Example: Configuring Interprovider Layer 3 VPN Option B

This example provides a step-by-step procedure to configure interprovider layer 3 VPN option B, which is one of the recommended implementations of an MPLS VPN when that

service is required by a customer that has more than one AS, but not all of the customer's ASs can be serviced by the same service provider. It is organized in the following sections:

Requirements

This example uses the following hardware and software components:

- Junos OS Release 9.5 or later.
- Eight M Series, T Series, TX Series, or MX Series Juniper Networks routers.

Configuration Overview and Topology

Interprovider layer 3 VPN option B is a somewhat scalable solution to the problem of providing VPN services to a customer that has different sites, not all of which can use the same service provider. *RFC 4364*, section 10, refers to this method as interprovider EBGP redistribution of labeled VPN-IPv4 routes from AS to neighboring AS.

In the topology shown in Figure 1, the following events occur:

- The PE routers use IBGP to redistribute labeled VPN-IPv4 routes either to an ASBR, or to a route reflector of which an ASBR is a client.
- The ASBR then uses EBGP to redistribute those labeled VPN-IPv4 routes to an ASBR in another AS, which distributes them to the PE routers in that AS, or to another ASBR for distribution.
- Labeled VPN-IPv4 routes are distributed between ASBR routers on each site. There is no need to define a separate VPN routing and forwarding instance (VRF) for each common VPN that resides on two different SPs.
- Router PE2 distributes VPN-IPv4 routes to Router ASBR2 using MP-IBGP.
- Router ASBR2 distributes these labeled VPN-IPv4 routes to Router ASBR1, using the MP-EBGP session between them.
- Router ASBR1 redistributes those routes to Router PE1, using MP-IBGP. Each time a label is advertised, routers change the next-hop information and labels.
- An MPLS path is established between Router PE1 and Router PE2. This path enables changing of the next-hop attribute for the routes that are learned from the neighbor SP router and map the incoming label for the given routes to the outgoing label advertised to PE routers in the internal network.
- The ingress PE router inserts two labels onto the IP packet coming from the end customer. The inner label is for the VPN-IPv4 routes learned from internal ASBRs and the outer label is for the route to the internal ASBR, obtained through resource reservation protocol (RSVP) or label distribution protocol (LDP).
- When a packet arrives at the ASBR, it removes the outer label (when explicit-null signaling is used; otherwise, penultimate hop-popping (PHP) pops the label) and

swaps the inner label with the label obtained from the neighbor ASBR through MP-EBGP label and prefix advertisements.

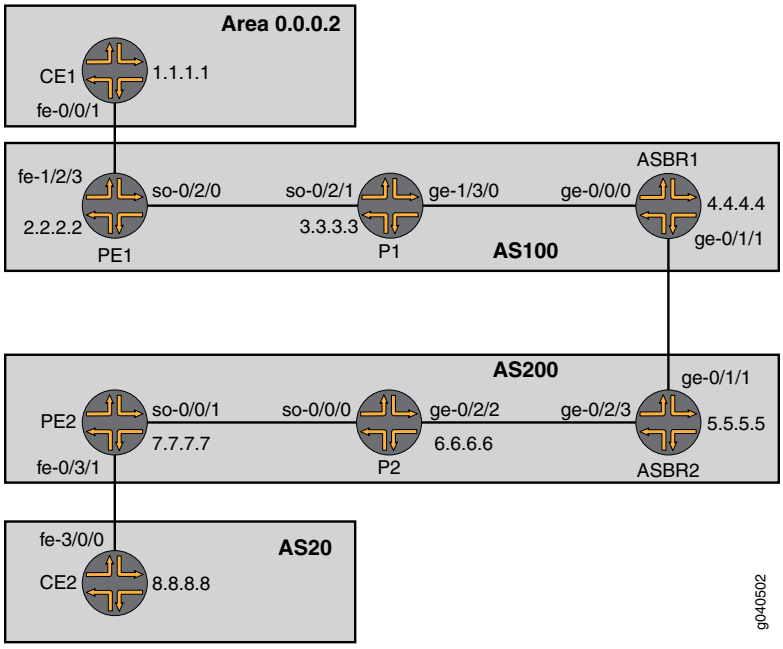
- The second ASBR swaps the VPN-IPv4 label and pushes another label to reach the PE router in its own AS.
- The remaining process is the same as for a regular VPN.

NOTE: In this solution, ASBR routers keep all VPN-IPv4 routes in the routing information base (RIB), and the labels associated with the prefixes are kept in the forwarding information base (FIB). Because the RIB and FIB tables can take occupy much of the respective allocated memory, this solution is not very scalable for an interprovider VPN.

If a transit SP is used between SP1 and SP2, the transit SP also has to keep all VPN-IPv4 routes in the RIB and the corresponding labels in the FIB. The ASBRs at the transit SP have the same functionality as ASBRs in the SP1 or SP2 networks in this solution.

The topology of the network is shown in Figure 42 on page 255.

Figure 42: Physical Topology of Interprovider Layer 3 VPN Option B



Configuration

NOTE: The procedure presented here is written with the assumption that the reader is already familiar with MPLS MVPN configuration. This example focuses on explaining the unique configuration required for carrier-of-carriers solutions for VPN services to different sites.

To configure layer 3 VPN option B, perform the following tasks:

Configuring Router CE1

Step-by-Step Procedure

1. On Router CE1, configure the IP address and protocol family on the Fast Ethernet interface for the link between Router CE1 and Router PE1. Specify the **inet** address family type.

```
[edit interfaces fe-0/0/1.0]
family inet {
  address 18.18.18.1/30;
}
```

2. On Router CE1, configure the IP address and protocol family on the loopback interface. Specify the **inet** address family type.

```
[edit interfaces lo0]
unit 0 {
  family inet {
    address 1.1.1.1/32;
  }
}
```

3. On Router CE1, configure an IGP. Include the logical interface for the link between Router CE1 and Router PE1 and the logical loopback interface of Router CE1. The IGP can be a static route, RIP, OSPF, ISIS, or EBGP. In this example we configure OSPF.

```
[edit protocols]
ospf {
  area 0.0.0.2 {
    interface fe-0/0/1.0;
    interface lo0.0;
```

```
    }
  }
```

Configuring Router PE1

Step-by-Step Procedure

1. On Router PE1, configure IPv4 addresses on the SONET, Fast Ethernet, and logical loopback interfaces. Specify the **inet** address family on all of the interfaces. Specify the **mpls** address family on the SONET and Fast Ethernet interfaces.

```
[edit interfaces]
so-0/2/0 {
  unit 0 {
    family inet {
      address 19.19.19.1/30;
    }
    family mpls;
  }
}
fe-1/2/3 {
  unit 0 {
    family inet {
      address 18.18.18.2/30;
    }
    family mpls;
  }
}
lo0 {
  unit 0 {
    family inet {
      address 2.2.2.2/32;
    }
  }
}
```

2. On Router PE1, configure the routing instance for VPN2. Specify the **vrf** instance type and specify the customer-facing Fast Ethernet interface. Configure a route distinguisher to create a unique VPN-IPv4 address prefix. Apply the VRF import and export policies to enable the sending and receiving of route targets. Configure the OSPF protocol within the VRF. Specify the customer-facing Fast Ethernet interface and specify the export policy to export BGP routes into OSPF.

```
[edit routing-instances]
vpn2CE1 {
  instance-type vrf;
  interface fe-1/2/3.0;
  route-distinguisher 1:100;
  vrf-import vpnimport;
  vrf-export vpnexport;
  protocols {
    ospf {
      export bgp-to-ospf;
      area 0.0.0.2 {
        interface fe-1/2/3.0;
      }
    }
```

```
    }
  }
```

3. On Router PE1, configure the RSVP and MPLS protocols to support the label-switched path (LSP). Configure the LSP to Router ASBR1 and specify the IP address of the logical loopback interface on Router ASBR1. Configure a BGP group. Specify the group type as **internal**. Specify the local address as the logical loopback interface on Router PE1. Specify the neighbor address as the logical loopback interface on Router ASBR1. Specify the **inet-vpn** address family and **unicast** traffic type to enable BGP to carry IPv4 network layer reachability information (NLRI) for VPN routes. Configure the OSPF protocol. Specify the core-facing SONET interface and specify the logical loopback interface on Router PE1.

```
[edit protocols]
rsvp {
  interface so-0/2/0.0;
  interface lo0.0;
}
mpls {
  label-switched-path To-ASBR1 {
    to 4.4.4.4;
  }
  interface so-0/2/0.0;
  interface lo0.0;
}
bgp {
  group To_ASBR1 {
    type internal;
    local-address 2.2.2.2;
    neighbor 4.4.4.4 {
      family inet-vpn {
        unicast;
      }
    }
  }
}
ospf {
  traffic-engineering;
  area 0.0.0.0 {
    interface so-0/2/0.0;
    interface lo0.0;
  }
}
```

4. On Router PE1, configure the BGP local autonomous system number.

```
[edit routing-options]
autonomous-system 100;
```

5. On Router PE1, configure a policy to export the BGP routes into OSPF.

```
[edit policy-options]
policy-statement bgp-to-ospf {
  term 1 {
    from protocol bgp;
    then accept;
  }
```

```
    term 2 {
      then reject;
    }
  }
```

6. On Router PE1, configure a policy to add the VRF route target to the routes being advertised for this VPN.

```
[edit policy-options]
  policy-statement vpnexport {
    term 1 {
      from protocol ospf;
      then {
        community add test_comm;
        accept;
      }
    }
    term 2 {
      then reject;
    }
  }
```

7. On Router PE1, configure a policy to import routes from BGP that have the **test_comm** community attached.

```
[edit policy-options]
  policy-statement vpnimport {
    term 1 {
      from {
        protocol bgp;
        community test_comm;
      }
      then accept;
    }
    term 2 {
      then reject;
    }
  }
```

8. On Router PE1, define the **test_comm** BGP community with a route target.

```
[edit policy-options]
community test_comm members target:1:100;
```

Configuring Router P1

Step-by-Step Procedure

1. On Router P1, configure IP addresses for the SONET and Gigabit Ethernet interfaces. Enable the interfaces to process the **inet** and **mpls** address families. Configure the IP addresses for the **lo0.0** loopback interface and enable the interface to process the **inet** address family.

```
[edit interfaces]
so-0/2/1 {
  unit 0 {
    family inet {
      address 19.19.19.2/30;
    }
```

```
      family mpls;
    }
  }
  ge-1/3/0 {
    unit 0 {
      family inet {
        address 20.20.20.1/30;
      }
      family mpls;
    }
  }
  lo0 {
    unit 0 {
      family inet {
        address 3.3.3.3/32;
      }
    }
  }
```

2. On Router P1, configure the RSVP and MPLS protocols to support the LSP. Specify the SONET and Gigabit Ethernet interfaces.

   Configure the OSPF protocol. Specify the SONET and Gigabit Ethernet interfaces and specify the logical loopback interface. Enable OSPF to support traffic engineering extensions.

```
[edit protocols]
rsvp {
  interface so-0/2/1.0;
  interface ge-1/3/0.0;
  interface lo0.0;
}
mpls {
  interface lo0.0;
  interface ge-1/3/0.0;
  interface so-0/2/1.0;
}
ospf {
  traffic-engineering;
  area 0.0.0.0 {
    interface ge-1/3/0.0;
    interface so-0/2/1.0;
    interface lo0.0;
  }
}
```

Configuring Router ASBR1

Step-by-Step Procedure

1. On Router ASBR1, configure IP addresses for the Gigabit Ethernet interfaces. Enable the interfaces to process the **inet** and **mpls** addresses families. Configure the IP addresses for the **lo0.0** loopback interface and enable the interface to process the **inet** address family.

```
[edit interfaces]
ge-0/0/0 {
  unit 0 {
```

```
      family inet {
        address 20.20.20.2/30;
      }
      family mpls;
    }
  }
  ge-0/1/1 {
    unit 0 {
      family inet {
        address 21.21.21.1/30;
      }
      family mpls;
    }
  }
  lo0 {
    unit 0 {
      family inet {
        address 4.4.4.4/32;
      }
    }
  }
```

2. On Router ASBR1, configure the RSVP and MPLS protocols to support the LSP. Specify the Gigabit Ethernet interfaces and the **lo0.0** logical loopback interface.

   Configure the OSPF protocol. Specify the SONET and Gigabit Ethernet interfaces and specify the logical loopback interface. Enable OSPF to support traffic engineering extensions.

   ```
   [edit protocols]
   rsvp {
     interface ge-0/0/0.0;
     interface lo0.0;
   }
   mpls {
     label-switched-path To_PE1 {
       to 2.2.2.2;
     }
     interface lo0.0;
     interface ge-0/0/0.0;
   }
   ospf {
     traffic-engineering;
     area 0.0.0.0 {
       interface ge-0/0/0.0;
       interface lo0.0;
     }
   }
   ```

3. On Router ASBR1, create the **To-PE1** internal BGP peer group. Specify the local IP peer address as the local **lo0.0** address. Specify the neighbor IP peer address as the **lo0.0** interface address of Router PE1.

   ```
   [edit protocols]
   bgp {
     group To-PE1 {
       type internal;
   ```

```
    local-address 4.4.4.4;
    neighbor 2.2.2.2 {
      family inet-vpn {
        unicast;
      }
    }
  }
}
```

4. On Router ASBR1, create the **To-ASBR2** external BGP peer group. Enable the router to use BGP to advertise NLRI for unicast routes. Specify the neighbor IP peer address as the Gigabit Ethernet interface address of Router ASBR2.

```
[edit protocols]
bgp {
  group To-ASBR2 {
    type external;
    family inet-vpn {
      unicast;
    }
    neighbor 21.21.21.2 {
      peer-as 200;
    }
  }
}
```

Configuring Router ASBR2

Step-by-Step Procedure

1. On Router ASBR2, configure IP addresses for the Gigabit Ethernet interfaces. Enable the interfaces to process the **inet** and **mpls** address families. Configure the IP address for the **lo0.0** loopback interface and enable the interface to process the **inet** address family.

```
[edit interfaces]
ge-0/1/1 {
  unit 0 {
    family inet {
      address 21.21.21.2/30;
    }
    family mpls;
  }
}
ge-0/2/3 {
  unit 0 {
    family inet {
      address 22.22.22.1/30;
    }
    family mpls;
  }
}
lo0 {
  unit 0 {
    family inet {
      address 5.5.5.5/32;
    }
```

```
    }
}
```

2. On Router ASBR2, configure the RSVP and MPLS protocols to support the LSP. Specify the Gigabit Ethernet interfaces.

   Configure the OSPF protocol. Specify the SONET and Gigabit Ethernet interfaces and specify the logical loopback interface. Enable OSPF to support traffic engineering extensions.

```
[edit protocols]
rsvp {
    interface ge-0/2/3.0;
    interface lo0.0;
}
mpls {
    label-switched-path To_PE2 {
        to 7.7.7.7;
    }
    interface lo0.0;
    interface ge-0/2/3.0;
}
ospf {
    traffic-engineering;
    area 0.0.0.0 {
        interface ge-0/2/3.0;
        interface lo0.0;
    }
}
```

3. On Router ASBR2, create the **To-PE2** internal BGP peer group. Specify the local IP peer address as the local **lo0.0** address. Specify the neighbor IP peer address as the **lo0.0** interface address of Router PE2.

```
[edit protocols]
bgp {
    group To-PE2 {
        type internal;
        local-address 5.5.5.5;
        neighbor 7.7.7.7 {
            family inet-vpn {
                unicast;
            }
        }
    }
```

4. On Router ASBR2, create the **To-ASBR1** external BGP peer group. Enable the router to use BGP to advertise NLRI for unicast routes. Specify the neighbor IP peer address as the Gigabit Ethernet interface on Router ASBR1.

```
[edit protocols]
bgp {
    group To-ASBR1 {
        type external;
        family inet-vpn {
            unicast;
        }
```

```
    neighbor 21.21.21.1 {
      peer-as 100;
    }
  }
}
```

Configuring Router P2

Step-by-Step Procedure

1. On Router P2, configure IP addresses for the SONET and Gigabit Ethernet interfaces. Enable the interfaces to process the **inet** and **mpls** addresses families. Configure the IP addresses for the **lo0.0** loopback interface and enable the interface to process the **inet** address family.

```
[edit interfaces]
so-0/0/0 {
  unit 0 {
    family inet {
      address 23.23.23.1/30;
    }
    family mpls;
  }
}
ge-0/2/2 {
  unit 0 {
    family inet {
      address 22.22.22.2/30;
    }
    family mpls;
  }
}
lo0 {
  unit 0 {
    family inet {
      address 6.6.6.6/32;
    }
  }
}
```

2. On Router P2, configure the RSVP and MPLS protocols to support the LSP. Specify the SONET and Gigabit Ethernet interfaces.

   Configure the OSPF protocol. Specify the SONET and Gigabit Ethernet interfaces and specify the logical loopback interface. Enable OSPF to support traffic engineering extensions.

```
[edit protocols]
rsvp {
  interface so-0/0/0.0;
  interface ge-0/2/2.0;
  interface lo0.0;
}
mpls {
  interface lo0.0;
  interface ge-0/2/2.0;
  interface so-0/0/0.0;
}
```

```
ospf {
  traffic-engineering;
  area 0.0.0.0 {
    interface ge-0/2/2.0;
    interface so-0/0/0.0;
    interface lo0.0;
  }
}
```

Configuring Router PE2

Step-by-Step Procedure

1. On Router PE2, configure IPv4 addresses on the SONET, Fast Ethernet, and logical loopback interfaces. Specify the **inet** address family on all of the interfaces. Specify the **mpls** address family on the SONET and Fast Ethernet interfaces.

```
[edit interfaces]
so-0/0/1 {
  unit 0 {
    family inet {
      address 23.23.23.2/30;
    }
    family mpls;
  }
}
fe-0/3/1 {
  unit 0 {
    family inet {
      address 24.24.24.1/30;
    }
    family mpls;
  }
}
lo0 {
  unit 0 {
    family inet {
      address 7.7.7.7/32;
    }
  }
}
```

2. On Router PE2, configure the routing instance for VPN2. Specify the **vrf** instance type and specify the customer-facing Fast Ethernet interface. Configure a route distinguisher to create a unique VPN-IPv4 address prefix. Apply the VRF import and export policies to enable the sending and receiving of route targets. Configure the BGP peer group within the VRF. Specify AS **20** as the peer AS and specify the IP address of the Fast Ethernet interface on Router CE1 as the neighbor address.

```
[edit routing-instances]
vpn2CE2 {
  instance-type vrf;
  interface fe-0/3/1.0;
  route-distinguisher 1:100;
  vrf-import vpnimport;
  vrf-export vpnexport;
  protocols {
```

```
    bgp {
      group To_CE2 {
        peer-as 20;
        neighbor 24.24.24.2;
      }
    }
  }
}
```

3. On Router PE2, configure the RSVP and MPLS protocols to support the LSP. Configure the LSP to ASBR2 and specify the IP address of the logical loopback interface on Router ASBR2. Configure a BGP group. Specify the group type as **internal**. Specify the local address as the logical loopback interface on Router PE2. Specify the neighbor address as the logical loopback interface on the Router ASBR2. Specify the **inet-vpn** address family and **unicast** traffic type to enable BGP to carry IPv4 NLRI for VPN routes. Configure the OSPF protocol. Specify the core-facing SONET interface and the logical loopback interface on Router PE2.

```
[edit protocols]
rsvp {
  interface so-0/0/1.0;
  interface lo0.0;
}
mpls {
  label-switched-path To-ASBR2 {
    to 5.5.5.5;
  }
  interface so-0/0/1.0;
  interface lo0.0;
}
bgp {
  group To_ASBR2 {
    type internal;
    local-address 7.7.7.7;
    neighbor 5.5.5.5 {
      family inet-vpn {
        unicast;
      }
    }
  }
}
ospf {
  traffic-engineering;
  area 0.0.0.0 {
    interface so-0/0/1.0;
    interface lo0.0;
  }
}
```

4. On Router PE2, configure the BGP local autonomous system number.

```
[edit routing-options]
autonomous-system 200;
```

5. On Router PE2, configure a policy to add the VRF route target to the routes being advertised for this VPN.

```
[edit policy-options]
policy-statement vpnexport {
  term 1 {
    from protocol bgp;
    then {
      community add test_comm;
      accept;
    }
  }
  term 2 {
    then reject;
  }
}
```

6. On Router PE2, configure a policy to import routes from BGP that have the **test_comm** community attached.

```
[edit policy-options]
policy-statement vpnimport {
  term 1 {
    from {
      protocol bgp;
      community test_comm;
    }
    then accept;
  }
  term 2 {
    then reject;
  }
}
```

7. On Router PE1, define the **test_comm** BGP community with a route target.

```
[edit policy-options]
community test_comm members target:1:100;
```

Configuring Router CE2

Step-by-Step Procedure

1. On Router CE2, configure the IP address and protocol family on the Fast Ethernet interface for the link between Router CE2 and Router PE2. Specify the **inet** address family type.

```
[edit interfaces]
fe-3/0/0 {
  unit 0 {
    family inet {
      address 24.24.24.2/30;
    }
  }
}
```

2. On Router CE2, configure the IP address and protocol family on the loopback interface. Specify the **inet** address family type.

```
[edit interfaces]
lo0 {
  unit 0 {
```

```
    family inet {
      address 8.8.8.8/32;
    }
  }
}
```

3. On Router CE2, configure an IGP. The IGP can be a static route, RIP, OSPF, ISIS, or EBGP. In this example, we configure EBGP. Specify AS **200** as the peer AS and specify the BGP neighbor IP address as the Fast Ethernet interface of Router PE2. Include the **export** statement.

```
[edit protocols]
bgp {
  group To_PE2 {
    neighbor 24.24.24.1 {
      export myroutes;
      peer-as 200;
    }
  }
}
```

Verifying the VPN Operation

Step-by-Step Procedure

1. Commit the configuration on each router.

NOTE: The MPLS labels shown in this example will be different than the labels used in your configuration.

2. On Router PE1, display the routes for the **vpn2CE1** routing instance using the **show ospf route** command. Verify that the **1.1.1.1** route is learned from OSPF.

```
user@PE1> show ospf route instance vpn2CE1

Topology default Route Table:

Prefix             Path   Route       NH    Metric  NextHop       Nexthop
                   Type   Type        Type          Interface     addr/label
1.1.1.1           Intra  Router      IP         1  fe-1/2/3.0    18.18.18.1
1.1.1.1/32          Intra  Network     IP         1  fe-1/2/3.0    18.18.18.1
```

3. On Router PE1, use the **show route advertising-protocol** command to verify that Router PE1 advertises the **1.1.1.1** route to Router ASBR1 using MP-BGP with the VPN MPLS label.

```
user@PE1> show route advertising-protocol bgp 4.4.4.4 extensive

vpn2CE1.inet.0: 5 destinations, 5 routes (5 active, 0 holddown, 0 hidden)
* 1.1.1.1/32 (1 entry, 1 announced)
 BGP group To_ASBR1 type Internal
     Route Distinguisher: 1:100
     VPN Label: 299952
     Nexthop: Self
     Flags: Nexthop Change
     MED: 1
```

```
     Localpref: 100
     AS path: [100] I
     Communities: target:1:100 rte-type:0.0.0.2:1:0
```

4. On Router ASBR1, use the **show route receive-protocol** command to verify that the router receives and accepts the **1.1.1.1** route and places it in the **bgp.l3vpn.0** routing table.

```
user@ASBR1> show route receive-protocol bgp 2.2.2.2 extensive

inet.0: 9 destinations, 9 routes (9 active, 0 holddown, 0 hidden)

inet.3: 1 destinations, 1 routes (1 active, 0 holddown, 0 hidden)

mpls.0: 5 destinations, 5 routes (5 active, 0 holddown, 0 hidden)

bgp.l3vpn.0: 2 destinations, 2 routes (2 active, 0 holddown, 0 hidden)
* 1:100:1.1.1.1/32 (1 entry, 1 announced)
     Route Distinguisher: 1:100
     VPN Label: 299952
     Nexthop: 2.2.2.2
     MED: 1
     Localpref: 100
     AS path: I
     Communities: target:1:100 rte-type:0.0.0.2:1:0
```

5. On Router ASBR1, use the **show route advertising-protocol** command to verify that Router ASBR1 advertises the **1.1.1.1** route to Router ASBR2.

```
user@ASBR1> show route advertising-protocol bgp 21.21.21.2 extensive

bgp.l3vpn.0: 2 destinations, 2 routes (2 active, 0 holddown, 0 hidden)
* 1:100:1.1.1.1/32 (1 entry, 1 announced)
 BGP group To-ASBR2 type External
     Route Distinguisher: 1:100
     VPN Label: 299984
     Nexthop: Self
     Flags: Nexthop Change
     AS path: [100] I
     Communities: target:1:100 rte-type:0.0.0.2:1:0
```

6. On Router ASBR2, use the **show route receive-protocol** command to verify that the router receives and accepts the **1.1.1.1** route and places it in the **bgp.l3vpn.0** routing table.

```
user@ASBR2> show route receive-protocol bgp 21.21.21.1 extensive

inet.0: 9 destinations, 9 routes (9 active, 0 holddown, 0 hidden)

inet.3: 1 destinations, 1 routes (1 active, 0 holddown, 0 hidden)

mpls.0: 5 destinations, 5 routes (5 active, 0 holddown, 0 hidden)

bgp.l3vpn.0: 2 destinations, 2 routes (2 active, 0 holddown, 0 hidden)
* 1:100:1.1.1.1/32 (1 entry, 1 announced)
     Accepted
     Route Distinguisher: 1:100
     VPN Label: 299984
     Nexthop: 21.21.21.1
     AS path: 100 I
     Communities: target:1:100 rte-type:0.0.0.2:1:0
```

7. On Router ASBR2, use the **show route advertising-protocol** command to verify that Router ASBR2 advertises the **1.1.1.1** route to Router PE2 in the **To-PE2** routing instance.

```
user@ASBR2> show route advertising-protocol bgp 7.7.7.7 extensive

bgp.l3vpn.0: 2 destinations, 2 routes (2 active, 0 holddown, 0 hidden)
* 1:100:1.1.1.1/32 (1 entry, 1 announced)
 BGP group To-PE2 type Internal
     Route Distinguisher: 1:100
     VPN Label: 300048
     Nexthop: Self
     Flags: Nexthop Change
     Localpref: 100
     AS path: [200] 100 I
     Communities: target:1:100 rte-type:0.0.0.2:1:0
```

8. On Router PE2, use the **show route receive-protocol** command to verify that the router receives and accepts the **1.1.1.1** route and places it in the **To_CE2.inet.0** routing table.

```
user@PE2> show route receive-protocol bgp 5.5.5.5 extensive

inet.0: 12 destinations, 13 routes (12 active, 0 holddown, 0 hidden)

inet.3: 1 destinations, 1 routes (1 active, 0 holddown, 0 hidden)

__juniper_private1__.inet.0: 14 destinations, 14 routes (8 active, 0
holddown, 6 hidden)

__juniper_private2__.inet.0: 1 destinations, 1 routes (0 active, 0
holddown, 1 hidden)

To_CE2.inet.0: 4 destinations, 5 routes (4 active, 0 holddown, 0 hidden)
* 1.1.1.1/32 (1 entry, 1 announced)
     Accepted
     Route Distinguisher: 1:100
     VPN Label: 300048
     Nexthop: 5.5.5.5
     Localpref: 100
     AS path: 100 I
     AS path: Recorded
     Communities: target:1:100 rte-type:0.0.0.2:1:0

iso.0: 1 destinations, 1 routes (1 active, 0 holddown, 0 hidden)

mpls.0: 4 destinations, 4 routes (4 active, 0 holddown, 0 hidden)

bgp.l3vpn.0: 1 destinations, 1 routes (1 active, 0 holddown, 0 hidden)

* 1:100:1.1.1.1/32 (1 entry, 0 announced)
     Accepted
     Route Distinguisher: 1:100
     VPN Label: 300048
     Nexthop: 5.5.5.5
     Localpref: 100
     AS path: 100 I
     AS path: Recorded
     Communities: target:1:100 rte-type:0.0.0.2:1:0

__juniper_private1__.inet6.0: 4 destinations, 4 routes (4 active, 0
holddown, 0 hidden)
```

9. On Router PE2, use the **show route advertising-protocol** command to verify that Router PE2 advertises the **1.1.1.1** route to Router CE2 through the **To_CE2** peer group.

```
user@PE2> show route advertising-protocol bgp 24.24.24.2 extensive

To_CE2.inet.0: 4 destinations, 5 routes (4 active, 0 holddown, 0 hidden)
* 1.1.1.1/32 (1 entry, 1 announced)
 BGP group To_CE2 type External
     Nexthop: Self
     AS path: [200] 100 I
     Communities: target:1:100 rte-type:0.0.0.2:1:0
```

10. On Router CE2, use the **show route** command to verify that Router CE2 receives the **1.1.1.1** route from Router PE2.

```
user@CE2> show route 1.1.1.1

inet.0: 6 destinations, 6 routes (6 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

1.1.1.1/32          *[BGP/170] 00:25:36, localpref 100
                       AS path: 200 100 I
                     > to 24.24.24.1 via fe-3/0/0.0
```

11. On Router CE2, use the **ping** command and specify **8.8.8.8** as the source of the ping packets to verify connectivity with Router CE1.

```
user@CE2> ping 1.1.1.1 source 8.8.8.8

PING 1.1.1.1 (1.1.1.1): 56 data bytes
64 bytes from 1.1.1.1: icmp_seq=0 ttl=58 time=4.786 ms
64 bytes from 1.1.1.1: icmp_seq=1 ttl=58 time=10.210 ms
64 bytes from 1.1.1.1: icmp_seq=2 ttl=58 time=10.588 ms
```

12. On Router PE2, use the **show route** command to verify that the traffic is sent with an inner label of **300048** and a top label of **299776**.

```
user@PE2> show route 1.1.1.1 detail

To_CE2.inet.0: 4 destinations, 5 routes (4 active, 0 holddown, 0 hidden)
1.1.1.1/32 (1 entry, 1 announced)
        *BGP    Preference: 170/-101
                Route Distinguisher: 1:100
                Next hop type: Indirect
                Next-hop reference count: 3
                Source: 5.5.5.5
                Next hop type: Router, Next hop index: 653
                Next hop: via so-0/0/1.0 weight 0x1, selected
                Label-switched-path To-PE2
                Label operation: Push 300048, Push 299776(top)
                Protocol next hop: 5.5.5.5
                Push 300048
                Indirect next hop: 8c61138 262143
                State: <Secondary Active Int Ext>
                Local AS:   200 Peer AS:   200
                Age: 27:48      Metric2: 2
                Task: bgp_200.5.5.5.5+60185
                Announcement bits (3): 0-RT 1-KRT 2-BGP RT Background
                AS path: 100 I
                AS path: Recorded
                Communities: target:1:100 rte-type:0.0.0.2:1:0
                Accepted
```

```
                VPN Label: 300048
                Localpref: 100
                Router ID: 5.5.5.5
                Primary Routing Table bgp.l3vpn.0
```

13. On Router ASBR2, use the **show route table** command to verify that Router ASBR2 receives the traffic after the top label is popped by Router P2, that label **300048** is swapped with label **299984**, and that the packet is sent toward Router ASBR1 through interface **ge-0/1/1.0**.

```
user@ASBR2> show route table mpls.0 detail

300048 (1 entry, 1 announced)
        *VPN    Preference: 170
                Next hop type: Router, Next hop index: 648
                Next-hop reference count: 2
                Source: 21.21.21.1
                Next hop: 21.21.21.1 via ge-0/1/1.0, selected
                Label operation: Swap 299984
                State: <Active Int Ext>
                Local AS:   200
                Age: 30:39
                Task: BGP RT Background
                Announcement bits (1): 0-KRT
                AS path: 100 I
                Ref Cnt: 1
                Communities: target:1:100 rte-type:0.0.0.2:1:0
```

14. On Router ASBR1, use the **show route table** command to verify that Router ASBR1 receives the traffic with label **299984**, swaps the label with **299952**, and pushes a new top label of **299792**.

```
user@ASBR1> show route table mpls.0 detail

299984 (1 entry, 1 announced)
        *VPN    Preference: 170
                Next hop type: Indirect
                Next-hop reference count: 2
                Source: 2.2.2.2
                Next hop type: Router, Next hop index: 538
                Next hop: 20.20.20.1 via ge-0/0/0.0 weight 0x1, selected
                Label-switched-path To_PE1
                Label operation: Swap 299952, Push 299792(top)
                Protocol next hop: 2.2.2.2
                Swap 299952
                Indirect next hop: 8af70a0 262142
                State: <Active Int Ext>
                Local AS:   100
                Age: 34:09      Metric2: 2
                Task: BGP RT Background
                Announcement bits (1): 0-KRT
                AS path: I
                Ref Cnt: 1
                Communities: target:1:100 rte-type:0.0.0.2:1:0
```

15. On Router PE1, use the **show route table** command to verify that Router PE1 receives the traffic with label **299952**, and then pops the inner label.

```
user@PE1> show route route table mpls.0 detail
```

```
299952 (1 entry, 1 announced)
        *VPN    Preference: 170
                Next hop type: Router, Next hop index: 536
                Next-hop reference count: 2
                Next hop: 18.18.18.1 via fe-1/2/3.0, selected
                Label operation: Pop
                State: Active Int Ext
                Local AS:   100
                Age: 40:26
                Task: BGP RT Background
                Announcement bits (1): 0-KRT
                AS path: I
                Ref Cnt: 1
                Communities: rte-type:0.0.0.2:1:0
```

Related Documentation

- Interprovider Layer 3 VPN Option B Overview

Example: Configuring Interprovider Layer 3 VPN Option C

This example provides a step-by-step procedure to configure interprovider layer 3 VPN option C, which is one of the recommended implementations of MPLS VPN when that service is required by a customer that has more than one AS but not all of the customer's ASs can be serviced by the same service provider (SP). It is organized in the following sections:

Requirements

This example requires the following hardware and software components:

- Junos OS Release 9.5 or later.
- Eight Juniper Networks M Series Multiservice Edge Routers, T Series Core Routers, TX Matrix Routers, or MX Series 3D Universal Edge Routers.

Configuration Overview and Topology

Interprovider layer 3 VPN option C is a very scalable interprovider VPN solution to the problem of providing VPN services to a customer that has different sites, not all of which can use the same SP.

RFC 4364 section 10, refers to this method as multihop EBGP redistribution of labeled VPN-IPv4 routes between source and destination ASs, with EBGP redistribution of labeled IPv4 routes from AS to neighboring AS.

This solution is similar to the solution described in *Implementing Interprovider Layer 3 VPN Option B*, except internal IPv4 unicast routes are advertised instead of external VPN-IPv4-unicast routes, using EBGP. Internal routes are internal to leaf SPs (SP1 and SP2 in this example), and external routes are those learned from the end customer requesting VPN services.

In this configuration:

- After the loopback address of Router PE2 is learned by Router PE1 and the loopback address of Router PE1 is learned by Router PE2, the end PE routers establish an MP-EBGP session for exchanging VPN-IPv4 routes.
- Since VPN-IPv4 routes are exchanged among end PE routers, any other router on the path from Router PE1 and Router PE2 does not need to keep or install VPN-IPv4 routes in their routing information base (RIB) or forwarding information base (FIB) tables.
- An MPLS path needs to be established between Router PE1 and Router PE2.

RFC 4364 describes only one solution that uses a BGP labeled-unicast approach. In this approach, the ASBR routers advertise the loopback addresses of the PE routers and associate each prefix with a label according to *RFC 3107*. Service providers may use RSVP or LDP to establish an LSP between ASBR routers and PE routers in their internal network.

In this network, ASBR2 receives label information associated with the loopback IP address of Router PE1 and advertises another label to Router ASBR1 using MP-EBGP labeled-unicast. Meanwhile, the ASBRs build their own MPLS forwarding table according to the received and advertised routes and labels. Router ASBR1 uses its own IP address as the next-hop information.

Router ASBR2 receives this prefix associated with a label, assigns another label, changes the next-hop address to its own address, and advertises it to Router PE1. Router PE1 now has an update with the label information and next-hop to Router ASBR1. Also, Router PE1 already has a label associated with the IP address of Router ASBR1. If Router PE1 sends an IP packet to Router PE2, it pushes two labels: one for the IP address of Router PE2 (obtained using MP-IBGP labeled-unicast advertisement) and one for the IP address of Router ASBR1 (obtained using LDP or RSVP).

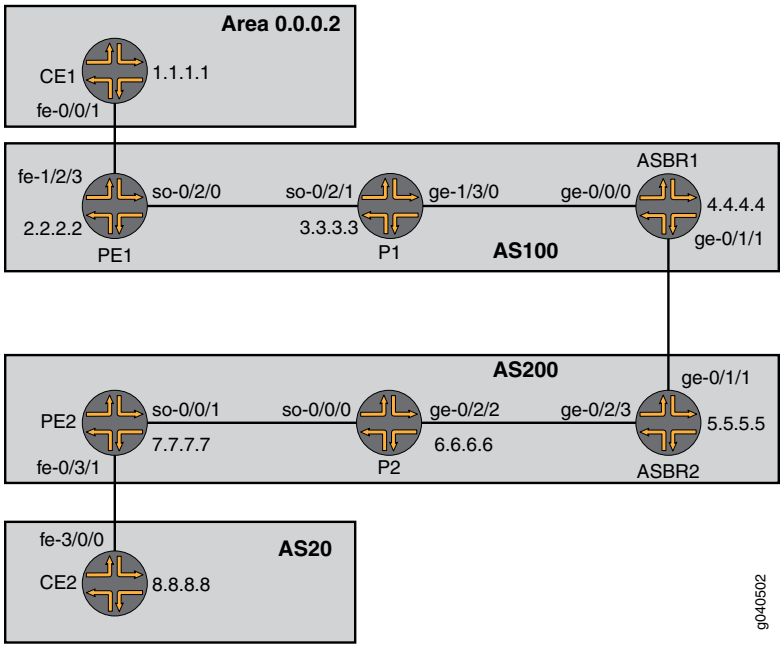
Router ASBR1 then pops the outer label and swaps the inner label with the label learned from a neighbor ASBR for its neighboring PE router. Router ASBR2 performs a similar function and swaps the incoming label (only one) and pushes another label that is associated with the address of Router PE2. Router PE2 pops both labels and passes the remaining IP packet to its own CPU. After the end-to-end connection among the PE

routers is created, the PE routers establish an MP-EBGP session to exchange VPN-IPv4 routes.

In this solution, PE routers push three labels onto the IP packet coming from the VPN end user. The inner-most label, obtained using MP-EBGP, determines the correct VPN routing and forwarding (VRF) routing instance at the remote PE. The middle label is associated with the IP address of the remote PE and is obtained from an ASBR using MP-IBGP labeled-unicast. The outer label is associated with the IP addresses of the ASBRs and is obtained using LDP or RSVP.

The physical topology of the network is shown in Figure 43 on page 275.

Figure 43: Physical Topology of Interprovider Layer 3 VPN Option C



Configuration

NOTE: The procedure presented here is written with the assumption that the reader is already familiar with MPLS MVPN configuration. This example focuses on explaining the unique configuration required for carrier-of-carriers solutions for VPN services to different sites.

To configure interprovider layer 3 VPN option C, perform the following tasks:

- Configuring Router CE1 on page 276
- Configuring Router PE1 on page 276
- Configuring Router P1 on page 279
- Configuring Router ASBR1 on page 280
- Configuring Router ASBR2 on page 283

Configuring Router CE1

Step-by-Step Procedure

1. On Router CE1, configure the IP address and protocol family on the Fast Ethernet interface for the link between Router CE1 and Router PE1. Specify the **inet** address family type.

```
[edit interfaces fe-0/0/1.0]
family inet {
   address 18.18.18.1/30;
}
```

2. On Router CE1, configure the IP address and protocol family on the loopback interface. Specify the **inet** address family type.

```
[edit interfaces lo0]
unit 0 {
   family inet {
      address 1.1.1.1/32;
   }
}
```

3. On Router CE1, configure an IGP. The IGP can be a static route, RIP, OSPF, ISIS, or EBGP. In this example we configure OSPF. Include the logical interface for the link between Router CE1 and Router PE1 and the logical loopback interface of Router CE1.

```
[edit protocols]
ospf {
   area 0.0.0.2 {
      interface fe-0/0/1.0;
      interface lo0.0;
   }
}
```

Configuring Router PE1

Step-by-Step Procedure

1. On Router PE1, configure IPv4 addresses on the SONET, Fast Ethernet, and logical loopback interfaces. Specify the **inet** address family on all of the interfaces. Specify the **mpls** address family on the SONET interfaces.

```
[edit interfaces]
so-0/2/0 {
   unit 0 {
      family inet {
         address 19.19.19.1/30;
      }
      family mpls;
   }
}
fe-1/2/3 {
```

```
    unit 0 {
      family inet {
        address 18.18.18.2/30;
      }
    }
  }
  lo0 {
    unit 0 {
      family inet {
        address 2.2.2.2/32;
      }
    }
  }
```

2. On Router PE1, configure the routing instance for VPN2. Specify the **vrf** instance type and specify the customer-facing Fast Ethernet interface. Configure a route distinguisher to create a unique VPN-IPv4 address prefix. Apply the VRF import and export policies to enable the sending and receiving of route targets. Configure the OSPF protocol within the VRF. Specify the customer-facing Fast Ethernet interface and specify the export policy to export BGP routes into OSPF.

```
[edit routing-instances]
vpn2CE1 {
  instance-type vrf;
  interface fe-1/2/3.0;
  route-distinguisher 1:100;
  vrf-import vpnimport;
  vrf-export vpnexport;
  protocols {
    ospf {
      export bgp-to-ospf;
      area 0.0.0.2 {
        interface fe-1/2/3.0;
      }
    }
  }
}
```

3. On Router PE1, configure the RSVP and MPLS protocols to support the LSP. Configure the LSP to Router ASBR1 and specify the IP address of the logical loopback interface on Router ASBR1. Configure the OSPF protocol. Specify the core-facing SONET interface and specify the logical loopback interface on Router PE1.

```
[edit protocols]
rsvp {
  interface so-0/2/0.0;
  interface lo0.0;
}
mpls {
  label-switched-path To-ASBR1 {
    to 4.4.4.4;
  }
  interface so-0/2/0.0;
  interface lo0.0;
}
ospf {
```

```
    traffic-engineering;
    area 0.0.0.0 {
      interface so-0/2/0.0;
      interface lo0.0;
    }
  }
```

4. On Router PE1, configure the **To_ASBR1** peer BGP group. Specify the group type as **internal**. Specify the local address as the logical loopback interface on Router PE1. Specify the neighbor address as the logical loopback interface on Router ASBR1. Specify the **inet** address family. For a PE router to install a route in the VRF, the next hop must resolve to a route stored within the **inet.3** table. The **labeled-unicast resolve-vpn** statements allow labeled routes to be placed in the **inet.3** routing table for route resolution, which are then resolved for PE router connections where the remote PE is located across another AS.

```
[edit protocols]
bgp {
  group To_ASBR1 {
    type internal;
    local-address 2.2.2.2;
    neighbor 4.4.4.4 {
      family inet {
        labeled-unicast {
          resolve-vpn;
        }
      }
    }
  }
}
```

5. On Router PE1, configure multihop EBGP toward PE2. Specify the **inet-vpn** family.

```
[edit protocols]
bgp {
  group To_PE2 {
    multihop {
      ttl 20;
    }
    local-address 2.2.2.2;
    family inet-VPN {
      unicast;
    }
    neighbor 7.7.7.7 {
      peer-as 200;
    }
  }
}
```

6. On Router PE1, configure the BGP local autonomous system number.

```
[edit routing-options]
autonomous-system 100;
```

7. On Router PE1, configure a policy to export the BGP routes into OSPF.

```
[edit policy-options]
policy-statement bgp-to-ospf {
```

```
    term 1 {
      from protocol bgp;
      then accept;
    }
    term 2 {
      then reject;
    }
  }
```

8. On Router PE1, configure a policy to add the VRF route target to the routes being advertised for this VPN.

```
[edit policy-options]
  policy-statement vpnexport {
    term 1 {
      from protocol ospf;
      then {
        community add test_comm;
        accept;
      }
    }
    term 2 {
      then reject;
    }
  }
```

9. On Router PE1, configure a policy to import routes from BGP that have the **test_comm** community attached.

```
[edit policy-options]
  policy-statement vpnimport {
    term 1 {
      from {
        protocol bgp;
        community test_comm;
      }
      then accept;
    }
    term 2 {
      then reject;
    }
  }
```

10. On Router PE1, define the **test_comm** BGP community with a route target.

```
[edit policy-options]
community test_comm members target:1:100;
```

Configuring Router P1

Step-by-Step Procedure

1. On Router P1, configure IP addresses for the SONET and Gigabit Ethernet interfaces. Enable the interfaces to process the **inet** and **mpls** address families. Configure the IP address for the **lo0.0** loopback interface and enable the interface to process the **inet** address family.

```
[edit interfaces]
so-0/2/1 {
```

```
    unit 0 {
      family inet {
        address 19.19.19.2/30;
      }
      family mpls;
    }
  }
  ge-1/3/0 {
    unit 0 {
      family inet {
        address 20.20.20.1/30;
      }
      family mpls;
    }
  }
  lo0 {
    unit 0 {
      family inet {
        address 3.3.3.3/32;
      }
    }
  }
```

2. On Router P1, configure the RSVP and MPLS protocols to support the LSP. Specify the SONET and Gigabit Ethernet interfaces.

   Configure the OSPF protocol. Specify the SONET and Gigabit Ethernet interfaces and specify the logical loopback interface. Enable OSPF to support traffic engineering extensions.

```
[edit protocols]
rsvp {
  interface so-0/2/1.0;
  interface ge-1/3/0.0;
  interface lo0.0;
}
mpls {
  interface lo0.0;
  interface ge-1/3/0.0;
  interface so-0/2/1.0;
}
ospf {
  traffic-engineering;
  area 0.0.0.0 {
    interface ge-1/3/0.0;
    interface so-0/2/1.0;
    interface lo0.0;
  }
}
```

Configuring Router ASBR1

Step-by-Step Procedure

1. On Router ASBR1, configure IP addresses for the Gigabit Ethernet interfaces. Enable the interfaces to process the **inet** and **mpls** addresses families. Configure the IP

addresses for the **lo0.0** loopback interface and enable the interface to process the **inet** address family.

```
[edit interfaces]
ge-0/0/0 {
    unit 0 {
        family inet {
            address 20.20.20.2/30;
        }
        family mpls;
    }
}
ge-0/1/1 {
    unit 0 {
        family inet {
            address 21.21.21.1/30;
        }
        family mpls;
    }
}
lo0 {
    unit 0 {
        family inet {
            address 4.4.4.4/32;
        }
    }
}
```

2. On Router ASBR1, configure the RSVP and MPLS protocols to support the LSP. Specify the Gigabit Ethernet interfaces and the logical loopback interface. Include the **traffic-engineering bgp-igp-both-ribs** statement at the **[edit protocols mpls]** hierarchy level.

   Configure the OSPF protocol. Specify the SONET and Gigabit Ethernet interfaces and specify the logical loopback interface. Enable OSPF to support traffic engineering extensions.

```
[edit protocols]
rsvp {
    interface ge-0/0/0.0;
    interface lo0.0;
}
mpls {
    traffic-engineering bgp-igp-both-ribs;
    label-switched-path To_PE1 {
        to 2.2.2.2;
    }
    interface lo0.0;
    interface ge-0/0/0.0;
    interface ge-0/1/1.0;
}
ospf {
    traffic-engineering;
    area 0.0.0.0 {
        interface ge-0/0/0.0;
        interface lo0.0;
```

```
    }
  }
```

3. On Router ASBR1, create the **To-PE1** internal BGP peer group. Specify the local IP peer address as the local **lo0.0** address. Specify the neighbor IP peer address as the Gigabit Ethernet interface address of Router PE1.

```
[edit protocols]
bgp {
  group To-PE1 {
    type internal;
    local-address 4.4.4.4;
    neighbor 2.2.2.2 {
      family inet {
        labeled-unicast;
      }
    }
  }
```

4. On Router ASBR1, create the **To-ASBR2** external BGP peer group. Enable the router to use BGP to advertise network layer reachability information (NLRI) for unicast routes. Specify the neighbor IP peer address as the Gigabit Ethernet interface address on Router ASBR2.

```
[edit protocols]
group To-ASBR2 {
  type external;
  family inet {
    labeled-unicast;
  }
  export To-ASBR2;
  neighbor 21.21.21.2 {
    peer-as 200;
  }
}
```

5. On Router ASBR1, configure the BGP local autonomous system number.

```
[edit routing-options]
autonomous-system 100;
```

6. On Router PE 1, configure a policy to import routes from BGP that have the **test_comm** community attached.

```
[edit policy-options]
policy-statement To-ASBR2 {
  term 1 {
    route-filter 2.2.2.2/32 exact;
    then accept;
  }
  term 2 {
    then reject;
  }
}
```


Configuring Router ASBR2

Step-by-Step Procedure

1. On Router ASBR2, configure IP addresses for the Gigabit Ethernet interfaces. Enable the interfaces to process the **inet** and **mpls** address families. Configure the IP address for the **lo0.0** loopback interface and enable the interface to process the **inet** address family.

```
[edit interfaces]
ge-0/1/1 {
    unit 0 {
        family inet {
            address 21.21.21.2/30;
        }
        family mpls;
    }
}
ge-0/2/3 {
    unit 0 {
        family inet {
            address 22.22.22.1/30;
        }
        family mpls;
    }
}
lo0 {
    unit 0 {
        family inet {
            address 5.5.5.5/32;
        }
    }
}
```

2. On Router ASBR2, configure the RSVP and MPLS protocols to support the LSP. Specify the Gigabit Ethernet interfaces. Include the **traffic-engineering bgp-igp-both-ribs** statement at the **[edit protocols mpls]** hierarchy level.

   Configure the OSPF protocol. Specify the SONET and Gigabit Ethernet interfaces and specify the logical loopback interface. Enable OSPF to support traffic engineering extensions.

```
[edit protocols]
rsvp {
    interface ge-0/2/3.0;
    interface lo0.0;
}
mpls {
    traffic-engineering bgp-igp-both-ribs;
    label-switched-path To_PE2 {
        to 7.7.7.7;
    }
    interface lo0.0
    interface ge-0/2/3.0;
    interface ge-0/1/1.0;
}
ospf {
```

```
    traffic-engineering;
    area 0.0.0.0 {
      interface ge-0/2/3.0;
      interface lo0.0;
    }
  }
```

3. On Router ASBR2, create the **To-PE2** internal BGP peer group. Specify the local IP peer address as the local **lo0.0** address. Specify the neighbor IP peer address as the **lo0.0** interface address of Router PE2.

```
[edit protocols]
bgp {
  group To-PE2 {
    type internal;
    local-address 5.5.5.5;
    neighbor 7.7.7.7 {
      family inet {
        labeled-unicast;
      }
    }
  }
}
```

4. On Router ASBR2, create the **To-ASBR1** external BGP peer group. Enable the router to use BGP to advertise NLRI for unicast routes. Specify the neighbor IP peer address as the Gigabit Ethernet interface address on Router ASBR1.

```
[edit protocols]
bgp {
  group To-ASBR1 {
    type external;
    family inet {
      labeled-unicast;
    }
    export To-ASBR1;
    neighbor 21.21.21.1 {
      peer-as 100;
    }
  }
}
```

5. On Router ASBR2 configure the BGP local autonomous system number.

```
[edit]
lab@ASBR2# show routing-options
autonomous-system 200;
```

6. On Router ASBR2, configure a policy to import routes from BGP that match the **4.4.4.4/32** route.

```
[edit policy-options]
policy-statement To-ASBR2 {
  term 1 {
    route-filter 4.4.4.4/32 exact;;
    then accept;
  }
  term 2 {
```

```
        then reject;
    }
}
```

Configuring Router P2

Step-by-Step Procedure

1. On Router P2, configure IP addresses for the SONET and Gigabit Ethernet interfaces. Enable the interfaces to process the **inet** and **mpls** addresses families. Configure the IP addresses for the **lo0.0** loopback interface and enable the interface to process the **inet** address family.

```
[edit interfaces]
so-0/0/0 {
    unit 0 {
        family inet {
            address 23.23.23.1/30;
        }
        family mpls;
    }
}
ge-0/2/2 {
    unit 0 {
        family inet {
            address 22.22.22.2/30;
        }
        family mpls;
    }
}
lo0 {
    unit 0 {
        family inet {
            address 6.6.6.6/32;
        }
    }
}
```

2. On Router P2, configure the RSVP and MPLS protocols to support the LSP. Specify the SONET and Gigabit Ethernet interfaces.

   Configure the OSPF protocol. Specify the SONET and Gigabit Ethernet interfaces and specify the logical loopback interface. Enable OSPF to support traffic engineering extensions.

```
[edit protocols]
rsvp {
    interface so-0/0/0.0;
    interface ge-0/2/2.0;
    interface lo0.0;
}
mpls {
    interface lo0.0;
    interface ge-0/2/2.0;
    interface so-0/0/0.0;
}
ospf {
    traffic-engineering;
```

```
    area 0.0.0.0 {
      interface ge-0/2/2.0;
      interface so-0/0/0.0;
      interface lo0.0;
    }
  }
```

Configuring Router PE2

Step-by-Step Procedure

1. On Router PE2, configure IPv4 addresses on the SONET, Fast Ethernet, and logical loopback interfaces. Specify the **inet** address family on all of the interfaces. Specify the **mpls** address family on the SONET interface.

```
[edit interfaces]
so-0/0/1 {
  unit 0 {
    family inet {
      address 23.23.23.2/30;
    }
    family mpls;
  }
}
fe-0/3/1 {
  unit 0 {
    family inet {
      address 24.24.24.1/30;
    }
  }
}
lo0 {
  unit 0 {
    family inet {
      address 7.7.7.7/32;
    }
  }
}
```

2. On Router PE2, configure the routing instance for VPN2. Specify the **vrf** instance type and specify the customer-facing Fast Ethernet interface. Configure a route distinguisher to create a unique VPN-IPv4 address prefix. Apply the VRF import and export policies to enable the sending and receiving of route targets. Configure the BGP peer group within the VRF. Specify AS **20** as the peer AS and specify the IP address of the Fast Ethernet interface on Router CE1 as the neighbor address.

```
[edit routing-instances]
vpn2CE2 {
  instance-type vrf;
  interface fe-0/3/1.0;
  route-distinguisher 1:100;
  vrf-import vpnimport;
  vrf-export vpnexport;
  protocols {
    bgp {
      group To_CE2 {
        peer-as 20;
```

```
            neighbor 24.24.24.2;
          }
        }
      }
    }
```

3. On Router PE2, configure the RSVP and MPLS protocols to support the LSP. Configure the LSP to ASBR2 and specify the IP address of the logical loopback interface on Router ASBR2. Configure the OSPF protocol. Specify the core-facing SONET interface and specify the logical loopback interface on Router PE2.

```
[edit protocols]
rsvp {
  interface so-0/0/1.0;
  interface lo0.0;
}
mpls {
  label-switched-path To-ASBR2 {
    to 5.5.5.5;
  }
  interface so-0/0/1.0;
  interface lo0.0;
}
ospf {
  traffic-engineering;
  area 0.0.0.0 {
    interface so-0/0/1.0;
    interface lo0.0;
  }
}
```

4. On Router PE2, configure the **To_ASBR2** BGP group. Specify the group type as **internal**. Specify the local address as the logical loopback interface on Router PE2. Specify the neighbor address as the logical loopback interface on the Router ASBR2.

```
[edit protocols]
bgp {
  group To_ASBR2 {
    type internal;
    local-address 7.7.7.7;
    neighbor 5.5.5.5 {
      family inet {
        labeled-unicast {
          resolve-vpn;
        }
      }
    }
  }
}
```

5. On Router PE2, configure multihop EBGP towards Router PE1 Specify the **inet-vpn** address family.

```
[edit protocols]
bgp {
  group To_PE1 {
    multihop {
```

```
        ttl 20;
      }
      family inet-vpn {
        unicast;
      }
      neighbor 2.2.2.2 {
        peer-as 100;
      }
    }
  }
```

6. On Router PE2, configure the BGP local autonomous system number.

```
[edit routing-options]
autonomous-system 200;
```

7. On Router PE2, configure a policy to add the VRF route target to the routes being advertised for this VPN.

```
[edit policy-options]
policy-statement vpnexport {
  term 1 {
    from protocol bgp;
    then {
      community add test_comm;
      accept;
    }
  }
  term 2 {
    then reject;
  }
}
```

8. On Router PE2, configure a policy to import routes from BGP that have the **test_comm** community attached.

```
[edit policy-options]
policy-statement vpnimport {
  term 1 {
    from {
      protocol bgp;
      community test_comm;
    }
    then accept;
  }
  term 2 {
    then reject;
  }
}
```

9. On Router PE1, define the **test_comm** BGP community with a route target.

```
[edit policy-options]
community test_comm members target:1:100;
```

Configuring Router CE2

Step-by-Step Procedure

1. On Router CE2, configure the IP address and protocol family on the Fast Ethernet interface for the link between Router CE2 and Router PE2. Specify the **inet** address family type.

```
[edit interfaces]
fe-3/0/0 {
  unit 0 {
    family inet {
      address 24.24.24.2/30;
    }
  }
}
```

2. On Router CE2, configure the IP address and protocol family on the loopback interface. Specify the **inet** address family type.

```
[edit interfaces lo0]
lo0 {
  unit 0 {
    family inet {
      address 8.8.8.8/32;
    }
  }
}
```

3. On Router CE2, configure an IGP. The IGP can be a static route, RIP, OSPF, ISIS, or EBGP. In this example, we configure EBGP. Specify the BGP neighbor IP address as the logical loopback interface of Router PE1.

```
[edit protocols]
bgp {
  group To_PE2 {
    neighbor 24.24.24.1 {
      export myroutes;
      peer-as 200;
    }
  }
}
```

Verifying the VPN Operation

Step-by-Step Procedure

1. Commit the configuration on each router.

NOTE: The MPLS labels shown in this example will be different than the labels used in your configuration.

2. On Router PE1, display the routes for the **vpn2CE1** routing instance using the **show ospf route** command. Verify that the **1.1.1.1** route is learned from OSPF.

```
user@PE1> show ospf route instance vpn2CE1
```

```
Topology default Route Table:

Prefix              Path   Route       NH   Metric  NextHop       Nexthop

                   Type   Type        Type         Interface     addr/label
1.1.1.1             Intra  Router      IP        1  fe-1/2/3.0    18.18.18.1
1.1.1.1/32         Intra  Network     IP        1  fe-1/2/3.0    18.18.18.1
```

3. On Router PE1, use the **show route advertising-protocol** command to verify that Router PE1 advertises the **1.1.1.1** route to Router PE2 using MP-BGP with the VPN MPLS label.

```
user@PE1> show route advertising-protocol bgp 7.7.7.7 extensive

bgp.l3vpn.0: 2 destinations, 2 routes (2 active, 0 holddown, 0 hidden)
* 1:100:1.1.1.1/32 (1 entry, 1 announced)
 BGP group To_PE2 type External
     Route Distinguisher: 1:100
     VPN Label: 300016
     Nexthop: Self
     Flags: Nexthop Change
     MED: 1
     AS path: [100] I
     Communities: target:1:100 rte-type:0.0.0.2:1:0
```

4. On Router ASBR1, use the **show route advertising-protocol** command to verify that Router ASBR1 advertises the **2.2.2.2** route to Router ASBR2.

```
user@ASBR1> show route advertising-protocol bgp 21.21.21.2 extensive

inet.0: 14 destinations, 16 routes (14 active, 0 holddown, 0 hidden)
* 2.2.2.2/32 (2 entries, 1 announced)
 BGP group To-PE2 type External
     Route Label: 300172
     Nexthop: Self
     Flags: Nexthop Change
     MED: 2
     AS path: [100] I
```

5. On Router ASBR2, use the **show route receive-protocol** command to verify that the router receives and accepts the **2.2.2.2** route and places it in the **To_ASBR2.inet.0** routing table.

```
user@ASBR2> show route receive-protocol bgp 21.21.21.1 extensive

inet.0: 10 destinations, 11 routes (10 active, 0 holddown, 0 hidden)
* 2.2.2.2/32 (1 entry, 1 announced)
     Accepted
     Route Label: 300172
     Nexthop: 21.21.21.1
     MED: 2
     AS path: 100 I
```

6. On Router ASBR2, use the **show route advertising-protocol** command to verify that Router ASBR2 advertises the **2.2.2.2** route to Router PE2 in the **To-PE2** routing instance.

```
user@ASBR2> show route advertising-protocol bgp 7.7.7.7 extensive

inet.0: 10 destinations, 11 routes (10 active, 0 holddown, 0 hidden)
* 2.2.2.2/32 (1 entry, 1 announced)
```

```
 BGP group To-PE2 type Internal
     Route Label: 300192
     Nexthop: Self
     Flags: Nexthop Change
     MED: 2
     Localpref: 100
     AS path: [200] 100 I
```

7. On Router PE2, use the **show route receive-protocol** command to verify that Router PE2 receives the route and puts it in the **inet.0.** routing table. Verify that Router PE2 also receives the update from Router PE1 and accepts the route.

```
user@PE2> show route receive-protocol bgp 5.5.5.5 extensive

inet.0: 13 destinations, 14 routes (13 active, 0 holddown, 0 hidden)
* 2.2.2.2/32 (1 entry, 1 announced)
     Accepted
     Route Label: 300192
     Nexthop: 5.5.5.5
     MED: 2
     Localpref: 100
     AS path: 100 I
     AS path: Recorded

inet.3: 2 destinations, 2 routes (2 active, 0 holddown, 0 hidden)

* 2.2.2.2/32 (1 entry, 1 announced)
     Accepted
     Route Label: 300192
     Nexthop: 5.5.5.5
     MED: 2
     Localpref: 100
     AS path: 100 I
     AS path: Recorded
```

8. On Router PE2, use the **show route receive-protocol** command to verify that Router PE2 puts the route in the routing table of the **To_CE2** routing instance and advertises the route to Router CE2 using EBGP.

```
user@PE2> show route receive-protocol bgp 2.2.2.2 detail

inet.0: 17 destinations, 18 routes (17 active, 0 holddown, 0 hidden)

inet.3: 6 destinations, 6 routes (6 active, 0 holddown, 0 hidden)

__juniper_private1__.inet.0: 14 destinations, 14 routes (8 active, 0
holddown, 6 hidden)

__juniper_private2__.inet.0: 1 destinations, 1 routes (0 active, 0
holddown, 1 hidden)

To_CE2.inet.0: 4 destinations, 5 routes (4 active, 0 holddown, 0 hidden)
* 1.1.1.1/32 (1 entry, 1 announced)
     Accepted
     Route Distinguisher: 1:100
     VPN Label: 300016
     Nexthop: 2.2.2.2
     MED: 1
     AS path: 100 I
     AS path: Recorded
     Communities: target:1:100 rte-type:0.0.0.2:1:0
```

```
iso.0: 1 destinations, 1 routes (1 active, 0 holddown, 0 hidden)

mpls.0: 4 destinations, 4 routes (4 active, 0 holddown, 0 hidden)

bgp.l3vpn.0: 2 destinations, 2 routes (2 active, 0 holddown, 0 hidden)

* 1:100:1.1.1.1/32 (1 entry, 0 announced)
     Accepted
     Route Distinguisher: 1:100
     VPN Label: 300016
     Nexthop: 2.2.2.2
     MED: 1
     AS path: 100 I
     AS path: Recorded
     Communities: target:1:100 rte-type:0.0.0.2:1:0

__juniper_private1__.inet6.0: 4 destinations, 4 routes (4 active, 0
holddown, 0 hidden)
```

9. On Router PE2, use the **show route advertising-protocol** command to verify that Router PE2 advertises the **1.1.1.1** route to Router CE2 through the **To_CE2** peer group.

```
user@PE2> show route advertising-protocol bgp 24.24.24.2 extensive

To_CE2.inet.0: 4 destinations, 5 routes (4 active, 0 holddown, 0 hidden)
* 1.1.1.1/32 (1 entry, 1 announced)
 BGP group To_CE2 type External
     Nexthop: Self
     AS path: [200] 100 I
     Communities: target:1:100 rte-type:0.0.0.2:1:0
```

10. On Router CE2, use the **show route** command to verify that Router CE2 receives the **1.1.1.1** route from Router PE2.

```
user@CE2> show route 1.1.1.1

inet.0: 6 destinations, 6 routes (6 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

1.1.1.1/32          *[BGP/170] 00:25:36, localpref 100
                       AS path: 200 100 I
                     > to 24.24.24.1 via fe-3/0/0.0
```

11. On Router CE2, use the **ping** command and specify **8.8.8.8** as the source of the ping packets to verify connectivity with Router CE1.

```
user@CE2> ping 1.1.1.1 source 8.8.8.8

PING 1.1.1.1 (1.1.1.1): 56 data bytes
64 bytes from 1.1.1.1: icmp_seq=0 ttl=58 time=4.786 ms
64 bytes from 1.1.1.1: icmp_seq=1 ttl=58 time=10.210 ms
64 bytes from 1.1.1.1: icmp_seq=2 ttl=58 time=10.588 ms
```

12. On Router PE2, use the **show route** command to verify that the traffic is sent with an inner label of **300016**, a middle label of **300192**, and a top label of **299776**.

```
user@PE2> show route 1.1.1.1 detail

To_CE2.inet.0: 4 destinations, 5 routes (4 active, 0 holddown, 0 hidden)
1.1.1.1/32 (1 entry, 1 announced)
        *BGP    Preference: 170/-101
```



```
                    Route Distinguisher: 1:100
                    Next hop type: Indirect
                    Next-hop reference count: 3
                    Source: 2.2.2.2
                    Next hop type: Router, Next hop index: 653
                    Next hop: via so-0/0/1.0 weight 0x1, selected
                    Label-switched-path To-PE2
                  Label operation: Push 300016, Push 300192, Push 299776(top)
                    Protocol next hop: 2.2.2.2
                    Push 300016
                    Indirect next hop: 8c61138 262142
                    State: <Secondary Active Ext>
                    Local AS:   200 Peer AS:   100
                    Age: 17:33      Metric: 1       Metric2: 2
                    Task: BGP_100.2.2.2.2+62319
                    Announcement bits (3): 0-RT 1-KRT 2-BGP RT Background
                    AS path: 100 I
                    AS path: Recorded
                    Communities: target:1:100 rte-type:0.0.0.2:1:0
                    Accepted
                    VPN Label: 300016
                    Localpref: 100
                    Router ID: 2.2.2.2
                    Primary Routing Table bgp.l3vpn.0
```

13. On Router ASBR2, use the **show route table** command to verify that Router ASBR2 receives the traffic after the top label is popped by Router P2. Verify that label **300192** is a swapped with label **300176** and the traffic is sent towards Router ASBR1 using interface ge-0/1/1.0. At this point, the bottom label **300016** is preserved.

```
lab@ASBR2# show route table mpls.0 detail

300192 (1 entry, 1 announced)
        *VPN    Preference: 170
                Next hop type: Router, Next hop index: 660
                Next-hop reference count: 2
                Source: 21.21.21.1
                Next hop: 21.21.21.1 via ge-0/1/1.0, selected
                Label operation: Swap 300176
                State: <Active Int Ext>
                Local AS:   200
                Age: 24:01
                Task: BGP RT Background
                Announcement bits (1): 0-KRT
                AS path: 100 I
                Ref Cnt: 1
```

14. On Router ASBR1, use the **show route table** command to verify that when Router ASBR1 receives traffic with label **300176**, it swaps the label with **299824** to reach Router PE1.

```
user@ASBR1> show route table mpls.0 detail

300176 (1 entry, 1 announced)
        *VPN    Preference: 170
                Next hop type: Router, Next hop index: 651
                Next-hop reference count: 2
                Next hop: 20.20.20.1 via ge-0/0/0.0 weight 0x1, selected
                Label operation: Swap 299824
                State: <Active Int Ext>
                Local AS:   100
```

```
                    Age: 25:53
                    Task: BGP RT Background
                    Announcement bits (1): 0-KRT
                    AS path: I
                    Ref Cnt: 1
```

15. On Router PE1, use the **show route table** command to verify that Router PE1 receives the traffic after the top label is popped by Router P1. Verify that label **300016** is popped and the traffic is sent towards Router CE1 using interface **fe-1/2/3.0**.

```
user@PE1> show route table mpls.0 detail

300016 (1 entry, 1 announced)
        *VPN    Preference: 170
                Next hop type: Router, Next hop index: 643
                Next-hop reference count: 2
                Next hop: 18.18.18.1 via fe-1/2/3.0, selected
                Label operation: Pop
                State:< Active Int Ext>
                Local AS:   100
                Age: 27:37
                Task: BGP RT Background
                Announcement bits (1): 0-KRT
                AS path: I
                Ref Cnt: 1
                Communities: rte-type:0.0.0.2:1:0
```

Related Documentation

- Interprovider Layer 3 VPN Option C Overview

Layer 3 VPN Overview

Layer 3 VPNs are based on RFC 2547bis, *BGP/MPLS IP VPNs*. RFC 2547bis defines a mechanism by which service providers can use their IP backbones to provide VPN services to their customers. A Layer 3 VPN is a set of sites that share common routing information and whose connectivity is controlled by a collection of policies. The sites that make up a Layer 3 VPN are connected over a provider's existing Internet backbone. RFC 2547bis VPNs are also known as BGP/MPLS VPNs because BGP is used to distribute VPN routing information across the provider's backbone, and MPLS is used to forward VPN traffic across the backbone to remote VPN sites.

Customer networks, because they are private, can use either public addresses or private addresses, as defined in RFC 1918, *Address Allocation for Private Internets*. When customer networks that use private addresses connect to the Internet infrastructure, the private addresses might overlap with the same private addresses used by other network users. MPLS/BGP VPNs solve this problem by adding a *route distinguisher*, a VPN identifier prefix to each address from a particular VPN site, thereby creating an address that is unique both within the VPN and within the Internet.

In addition, each VPN has its own VPN-specific routing table that contains the routing information for that VPN only. To separate VPN routes from routes in the Internet or those in other VPNs, the PE router creates a separate routing table for each VPN called a VPN routing and forwarding (VRF) table. The PE router creates one VRF table for each VPN that has a connection to a customer edge (CE) router. Any customer or site that

belongs to the VPN can access only the routes in the VRF tables for that VPN. Every VRF table has one or more extended community attributes associated with it that identify the route as belonging to a specific collection of routers. One of these, the *route target* attribute, identifies a collection of sites (VRF tables) to which a PE router distributes routes. The PE router uses the route target to constrain the import of remote routes into its VRF tables.

When an ingress PE router receives routes advertised from a directly connected CE router, it checks the received route against the VRF export policy for that VPN.

- If the route matches, the route is converted to VPN-IPv4 format—that is, the route distinguisher is added to the route. The PE router then announces the route in VPN-IPv4 format to the remote PE routers. It also attaches a route target to each route learned from the directly connected sites. The route target attached to each route is based on the configured export target policy of the VRF table. The routes are then distributed using IBGP sessions, which are configured in the provider's core network.
- If the route from the CE router does not match, it is not exported to other PE routers, but it can still be used locally for routing, for example, if two CE routers in the same VPN are directly connected to the same PE router.

When an egress PE router receives a route, it checks it against the import policy on the IBGP session between the PE routers. If it passes, the router places the route into its bgp.l3vpn.0 table. At the same time, the router checks the route against the VRF import policy for the VPN. If it matches, the route distinguisher is removed from the route and the route is placed into the VRF table (the *routing-instance-name*.inet.0 table) in IPv4 format.

Related Documentation

- Layer 2 Circuit Overview
- Layer 2 VPN Overview
- Interconnecting Layer 2 VPNs with Layer 3 VPNs Overview on page 296
- Applications for Interconnecting a Layer 2 Circuit with a Layer 3 VPN on page 296
- Example: Interconnecting a Layer 2 Circuit with a Layer 3 VPN on page 321
- Example: Interconnecting a Layer 2 VPN with a Layer 3 VPN on page 298

Applications for Interconnecting a Layer 2 Circuit with a Layer 3 VPN

MPLS-based Layer 2 services are growing in demand among enterprise and service providers. This creates new challenges related to interoperability between Layer 2 and Layer 3 services for service providers who want to provide end-to-end value-added services. There are various reasons to stitch different Layer 2 services to one another and to Layer 3 services. For example, to expand the service offerings and to expand geographically. The Junos OS has various features to address the needs of the service provider.

Interconnecting a Layer 2 Circuit with a Layer 3 VPN includes the following benefits:

- Interconnecting a Layer 2 Circuit with a Layer 3 VPN enables the sharing of a service provider's core network infrastructure between IP and Layer 2 circuit services, reducing the cost of providing those services. A Layer 2 MPLS circuit allows service providers to create a Layer 2 circuit service over an existing IP and MPLS backbone.
- Service providers do not have to invest in separate Layer 2 equipment to provide Layer 2 circuit service. A service provider can configure a provider edge router to run any Layer 3 protocol in addition to the Layer 2 protocols. Customers who prefer to maintain control over most of the administration of their own networks want Layer 2 circuit connections with their service provider instead of a Layer 3 VPN connection.

Related Documentation

- Layer 2 Circuit Overview
- Layer 3 VPN Overview on page 294
- Example: Interconnecting a Layer 2 Circuit with a Layer 3 VPN on page 321

Interconnecting Layer 2 VPNs with Layer 3 VPNs Overview

As MPLS-based Layer 2 services grow in demand, new challenges arise for service providers to be able to interoperate with Layer 2 and Layer 3 services and give their customers value-added services. Junos OS has various features to address the needs of service providers. One of these features is the use of a logical tunnel interface. This Junos OS functionality makes use of a tunnel PIC to loop packets out and back from the Packet Forwarding Engine to link the Layer 2 network with the Layer 3 network. The solution is limited by the logical tunnel bandwidth constraints imposed by the tunnel PIC.

Interconnecting Layer 2 VPNs with Layer 3 VPNs Applications

Interconnecting a Layer 2 VPN with a Layer 3 VPN provides the following benefits:

- A single access line to provide multiple services—Traditional VPNs over Layer 2 circuits require the provisioning and maintenance of separate networks for IP and for VPN services. In contrast, Layer 2 VPNs enable the sharing of a provider's core network infrastructure between IP and Layer 2 VPN services, thereby reducing the cost of providing those services.
- Flexibility—Many different types of networks can be accommodated by the service provider. If all sites in a VPN are owned by the same enterprise, this is an intranet. If various sites are owned by different enterprises, the VPN is an extranet. A site can be located in more than one VPN.
- Wide range of possible policies—You can give every site in a VPN a different route to every other site, or you can force traffic between certain pairs of sites routed via a third site and so pass certain traffic through a firewall.
- Scalable network—This design enhances the scalability because it eliminates the need for provider edge (PE) routers to maintain all of the service provider's VPN routes. Each PE router maintains a VRF table for each of its directly connected sites. Each customer connection (such as a Frame Relay PVC, an ATM PVC, or a VLAN) is mapped to a specific VRF table. Thus, it is a port on the PE router and not a site that is associated with a VRF table. Multiple ports on a PE router can be associated with a single VRF table. It is the ability of PE routers to maintain multiple forwarding tables that supports the per-VPN segregation of routing information.
- Use of route reflectors—Provider edge routers can maintain IBGP sessions to route reflectors as an alternative to a full mesh of IBGP sessions. Deploying multiple route reflectors enhances the scalability of the RFC 2547bis model because it eliminates the need for any single network component to maintain all VPN routes.
- Multiple VPNs are kept separate and distinct from each other—The customer edge routers do not peer with each other. Two sites have IP connectivity over the common backbone only, and only if there is a VPN which contains both sites. This feature keeps the VPNs separate and distinct from each other, even if two VPNs have an overlapping address space.
- Simple for customers to use—Customers can obtain IP backbone services from a service provider, and they do not need to maintain their own backbones.

Related Documentation

- Layer 2 VPN Overview
- Layer 3 VPN Overview on page 294
- Example: Interconnecting a Layer 2 VPN with a Layer 3 VPN on page 298

Example: Interconnecting a Layer 2 VPN with a Layer 3 VPN

This example provides a step-by-step procedure and commands for interconnecting and verifying a Layer 2 VPN with a Layer 3 VPN. It contains the following sections:

Requirements

This example uses the following hardware and software components:

- Junos OS Release 9.3 or later
- Five MX Series routers
- Three M Series routers
- Two T Series routers

Overview and Topology

A Layer 2 VPN is a type of virtual private network (VPN) that uses MPLS labels to transport data. The communication occurs between the provider edge (PE) routers.

Layer 2 VPNs use BGP as the signaling protocol and, consequently, have a simpler design and require less provisioning overhead than traditional VPNs over Layer 2 circuits. BGP signaling also enables autodiscovery of Layer 2 VPN peers. Layer 2 VPNs can have either a full-mesh or a hub-and-spoke topology. The tunneling mechanism in the core network is, typically, MPLS. However, Layer 2 VPNs can also use other tunneling protocols, such as GRE.

Layer 3 VPNs are based on RFC 2547bis, *BGP/MPLS IP VPNs*. RFC 2547bis defines a mechanism by which service providers can use their IP backbones to provide VPN services to their customers. A Layer 3 VPN is a set of sites that share common routing information and whose connectivity is controlled by a collection of policies. The sites that make up a Layer 3 VPN are connected over a provider's existing public Internet backbone. RFC 2547bis VPNs are also known as BGP/MPLS VPNs because BGP is used to distribute VPN routing information across the provider's backbone, and MPLS is used to forward VPN traffic across the backbone to remote VPN sites.

Customer networks, because they are private, can use either public addresses or private addresses, as defined in RFC 1918, *Address Allocation for Private Internets*. When customer networks that use private addresses connect to the public Internet infrastructure, the private addresses might overlap with the same private addresses used by other network users. MPLS/BGP VPNs solve this problem by adding a *distinguisher*, a VPN identifier prefix to each address from a particular VPN site, thereby creating an address that is unique both within the VPN and within the public Internet.

In addition, each VPN has its own VPN-specific routing table that contains the routing information for that VPN only. To separate a VPN's routes from routes in the public Internet or those in other VPNs, the PE router creates a separate routing table for each VPN called a VPN routing and forwarding (VRF) table. The PE router creates one VRF table for each VPN that has a connection to a customer edge (CE) router. Any customer or site that belongs to the VPN can access only the routes in the VRF tables for that VPN. Every VRF table has one or more extended community attributes associated with it that identify the route as belonging to a specific collection of routers. One of these, the *route target* attribute, identifies a collection of sites (VRF tables) to which a PE router distributes routes. The PE router uses the route target to constrain the import of remote routes into its VRF tables.

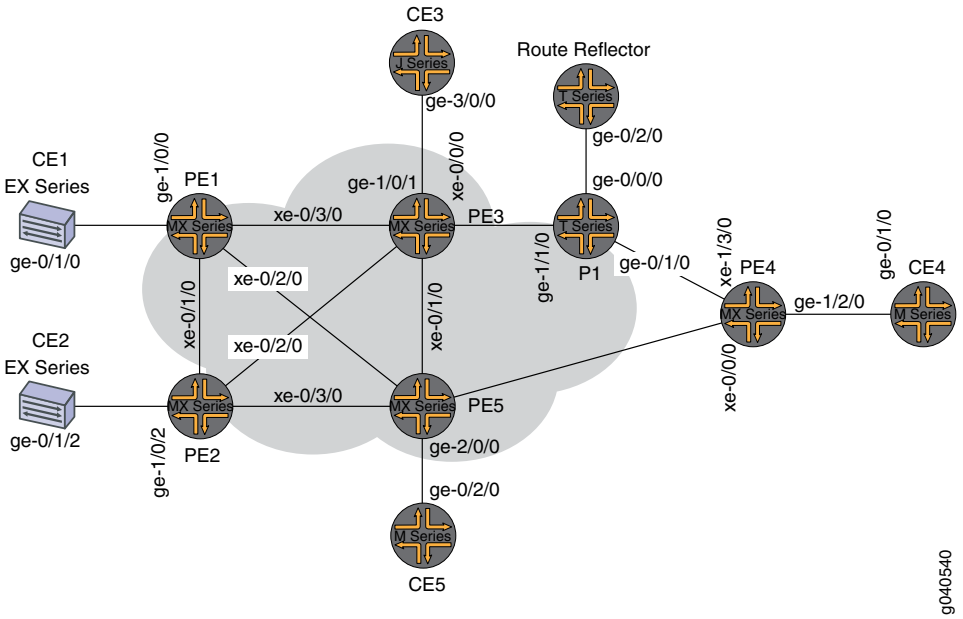
When an ingress PE router receives routes advertised from a directly connected CE router, it checks the received route against the VRF export policy for that VPN.

- If it matches, the route is converted to VPN-IPv4 format—that is, the route distinguisher is added to the route. The PE router then announces the route in VPN-IPv4 format to the remote PE routers. It also attaches a route target to each route learned from the directly connected sites. The route target attached to the route is based on the value of the VRF table's configured export target policy. The routes are then distributed using IBGP sessions, which are configured in the provider's core network.
- If the route from the CE router does not match, it is not exported to other PE routers, but it can still be used locally for routing, for example, if two CE routers in the same VPN are directly connected to the same PE router.

When an egress PE router receives a route, it checks it against the import policy on the IBGP session between the PE routers. If it passes, the router places the route into its bgp.l3vpn.0 table. At the same time, the router checks the route against the VRF import policy for the VPN. If it matches, the route distinguisher is removed from the route and the route is placed into the VRF table (the *routing-instance-name*.inet.0 table) in IPv4 format.

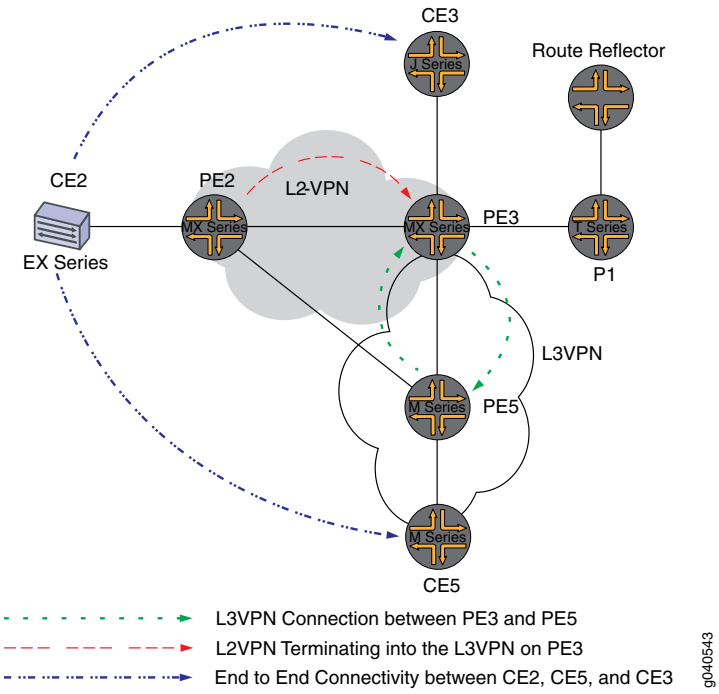
Figure 44 on page 300 shows the physical topology of a Layer 2 VPN-to-Layer 3 VPN interconnection.

Figure 44: Physical Topology of a Layer 2 VPN Terminating into a Layer 3 VPN



The logical topology of a Layer 2 VPN-to-Layer 3 VPN interconnection is shown in Figure 45 on page 300.

Figure 45: Logical Topology of a Layer 2 VPN Terminating into a Layer 3 VPN



The following definitions describe the meaning of the device abbreviations used in Figure 44 on page 300 and Figure 45 on page 300.

- Customer edge (CE) device—A device at the customer premises that provides access to the service provider's VPN over a data link to one or more provider edge (PE) routers.

  Typically the CE device is an IP router that establishes an adjacency with its directly connected PE routers. After the adjacency is established, the CE router advertises the site's local VPN routes to the PE router and learns remote VPN routes from the PE router.

- Provider edge (PE) device—A device, or set of devices, at the edge of the provider network that presents the provider's view of the customer site.

  PE routers exchange routing information with CE routers. PE routers are aware of the VPNs that connect through them, and PE routers maintain VPN state. A PE router is only required to maintain VPN routes for those VPNs to which it is directly attached. After learning local VPN routes from CE routers, a PE router exchanges VPN routing information with other PE routers using IBGP. Finally, when using MPLS to forward VPN data traffic across the provider's backbone, the ingress PE router functions as the ingress label-switching router (LSR) and the egress PE router functions as the egress LSR.

- Provider (P) device—A device that operates inside the provider's core network and does not directly interface to any CE.

  Although the P device is a key part of implementing VPNs for the service provider's customers and may provide routing for many provider-operated tunnels that belong to different VPNs, it is not itself VPN-aware and does not maintain VPN state. Its principal role is allowing the service provider to scale its VPN offerings, for example, by acting as an aggregation point for multiple PE routers.

  P routers function as MPLS transit LSRs when forwarding VPN data traffic between PE routers. P routers are required only to maintain routes to the provider's PE routers; they are not required to maintain specific VPN routing information for each customer site.

Configuration

To interconnect a Layer 2 VPN with a Layer 3 VPN, perform these tasks:

- Configuring the Base Protocols and Interfaces on page 301
- Configuring the VPN Interfaces on page 305

Configuring the Base Protocols and Interfaces

Step-by-Step Procedure

1. On each PE and P router, configure OSPF with traffic engineering extensions on all interfaces. Disable OSPF on the **fxp0.0** interface.

```
[edit protocols]
ospf {
    traffic-engineering;
    area 0.0.0.0 {
        interface all;
```

```
    interface fxp0.0 {
      disable;
    }
  }
}
```

2. On all the core routers, enable MPLS on all interfaces. Disable MPLS on the **fxp0.0** interface.

```
[edit protocols]
mpls {
  interface all;
  interface fxp0.0 {
    disable;
  }
}
```

3. On all the core routers, create an internal BGP peer group and specify the route reflector address (7.7.7.7) as the neighbor. Also enable BGP to carry Layer 2 VPLS network layer reachability information (NLRI) messages for this peer group by including the **signaling** statement at the **[edit protocols bgp group *group-name* family l2vpn]** hierarchy level.

```
[edit protocols]
bgp {
  group RR {
    type internal;
    local-address 2.2.2.2;
    family l2vpn {
      signaling;
    }
    neighbor 7.7.7.7;
  }
}
```

4. On Router PE3, create an internal BGP peer group and specify the route reflector IP address (7.7.7.7) as the neighbor. Enable BGP to carry Layer 2 VPLS NLRI messages for this peer group and enable the processing of VPN-IPv4 addresses by including the **unicast** statement at the **[edit protocols bgp group *group-name* family inet-vpn]** hierarchy level.

```
[edit protocols]
bgp {
  group RR {
    type internal;
    local-address 3.3.3.3;
    family inet-vpn {
      unicast;
    }
    family l2vpn {
      signaling;
    }
    neighbor 7.7.7.7;
  }
}
```

5. For the Layer 3 VPN domain on Router PE3 and Router PE5, enable RSVP on all interfaces. Disable RSVP on the **fxp0.0** interface.

```
[edit protocols]
rsvp {
    interface all;
    interface fxp0.0 {
        disable;
    }
}
```

6. On Router PE3 and Router PE5, create label-switched paths (LSPs) to the route reflector and the other PE routers. The following example shows the configuration on Router PE5.

```
[edit protocols]
mpls {
    label-switched-path to-RR {
        to 7.7.7.7;
    }
    label-switched-path to-PE2 {
        to 2.2.2.2;
    }
    label-switched-path to-PE3 {
        to 3.3.3.3;
    }
    label-switched-path to-PE4 {
        to 4.4.4.4;
    }
    label-switched-path to-PE1 {
        to 1.1.1.1;
    }
}
```

7. On Routers PE1, PE2, PE3, and PE5, configure the core interfaces with an IPv4 address and enable the MPLS address family. The following example shows the configuration of the **xe-0/1/0** interface on Router PE2.

```
[edit]
interfaces {
    xe-0/1/0 {
        unit 0 {
            family inet {
                address 10.10.2.2/30;
            }
            family mpls;
        }
    }
}
```

8. On Router PE2 and Router PE3, configure LDP for the Layer 2 VPN MPLS signaling protocol for all interfaces. Disable LDP on the **fxp0.0** interface. (RSVP can also be used.)

```
[edit protocols]
ldp {
  interface all;
  interface fxp0.0 {
    disable;
  }
}
```

9. On the route reflector, create an internal BGP peer group and specify the PE routers IP addresses as the neighbors.

```
[edit]
protocols {
  bgp {
    group RR {
      type internal;
      local-address 7.7.7.7;
      family inet {
        unicast;
      }
      family inet-vpn {
        unicast;
      }
      family l2vpn {
        signaling;
      }
      cluster 7.7.7.7;
      neighbor 1.1.1.1;
      neighbor 2.2.2.2;
      neighbor 4.4.4.4;
      neighbor 5.5.5.5;
      neighbor 3.3.3.3;
    }
  }
}
```

10. On the route reflector, configure MPLS LSPs towards Routers PE3 and PE5 to resolve the BGP next hops from inet.3 routing table.

```
[edit]
protocols {
  mpls {
    label-switched-path to-pe3 {
      to 3.3.3.3;
    }
    label-switched-path to-pe5 {
      to 5.5.5.5;
    }
    interface all;
  }
}
```

Configuring the VPN Interfaces

Step-by-Step Procedure

Router PE2 is one end of the Layer 2 VPN. Router PE3 is performing the Layer 2 VPN stitching between the Layer 2 VPN and the Layer 3 VPN. Router PE3 uses the logical tunnel interface (**lt** interface) configured with different logical interface units applied under two different Layer 2 VPN instances. The packet is looped though the **lt** interface configured on Router PE3. The configuration of Router PE5 contains the PE-CE interface.

1. On Router PE2, configure the **ge-1/0/2** interface encapsulation. Include the encapsulation statement and specify the **ethernet-ccc** option (**vlan-ccc** encapsulation is also supported) at the **[edit interfaces ge-1/0/2]** hierarchy level. The encapsulation should be the same in a whole Layer 2 VPN domain (Routers PE2 and PE3). Also, configure interface **lo0**.

```
[edit]
interfaces {
   ge-1/0/2 {
      encapsulation ethernet-ccc;
      unit 0;
   }
   lo0 {
      unit 0 {
         family inet {
            address 2.2.2.2/32;
         }
      }
   }
}
```

2. On Router PE2, configure the routing instance at the [**edit routing-instances]** hierarchy level. Also, configure the Layer 2 VPN protocol at the [**edit routing-instances *routing-instances-name* protocols]** hierarchy level. Configure the remote site ID as 3. Site ID 3 represents Router PE3 (Hub-PE). The Layer 2 VPN is using LDP as the signaling protocol. Be aware that in the following example, both the routing instance and the protocol are named **l2vpn**.

```
[edit]
routing-instances {
   l2vpn { # routing instance
      instance-type l2vpn;
      interface ge-1/0/2.0;
      route-distinguisher 65000:2;
      vrf-target target:65000:2;
      protocols {
         l2vpn { # protocol
            encapsulation-type ethernet;
            site CE2 {
               site-identifier 2;
               interface ge-1/0/2.0 {
                  remote-site-id 3;
               }
            }
         }
      }
```

```
    }
  }
```

3. On Router PE5, configure the Gigabit Ethernet interface for the PE-CE link **ge-2/0/0** and configure the **lo0** interface.

```
[edit interfaces]
ge-2/0/0 {
  unit 0 {
    family inet {
      address 80.80.80.1/24;
    }
  }
}
lo0 {
  unit 0 {
  }
}
```

4. On Router PE5, configure the Layer 3 VPN routing instance (**L3VPN**) at the **[edit routing-instances]** hierarchy level. Also configure BGP at the **[edit routing-instances L3VPN protocols]** hierarchy level.

```
[edit]
routing-instances {
  L3VPN {
    instance-type vrf;
    interface ge-2/0/0.0;
    route-distinguisher 65000:5;
    vrf-target target:65000:2;
    vrf-table-label;
    protocols {
      bgp {
        group ce5 {
          neighbor 80.80.80.2 {
            peer-as 200;
          }
        }
      }
    }
  }
}
```

5. In an MX Series router, such as Router PE3, you must create the tunnel services interface to be used for tunnel services. To create the tunnel service interface, include the **bandwidth** statement and specify the amount of bandwidth to reserve for tunnel services in gigabits per second at the **[edit chassis fpc *slot-number* pic *slot-number* tunnel-services]** hierarchy level.

```
[edit]
chassis {
  dump-on-panic;
  fpc 1 {
    pic 1 {
      tunnel-services {
        bandwidth 1g;
      }
```



```
    }
  }
}
```

6. On Router PE3, configure the Gigabit Ethernet interface.

   Include the **address** statement at the **[edit interfaces ge-1/0/1.0 family inet]** hierarchy level and specify **90.90.90.1/24** as the IP address.

```
[edit]
interfaces {
  ge-1/0/1 {
    unit 0 {
      family inet {
        address 90.90.90.1/24;
      }
    }
  }
}
```

7. On Router PE3, configure the **lt-1/1/10.0** logical tunnel interface at the **[edit interfaces lt-1/1/10 unit 0]** hierarchy level. Router PE3 is the router that is *stitching* the Layer 2 VPN to the Layer 3 VPN using the logical tunnel interface. The configuration of the peer unit interfaces is what makes the interconnection.

   To configure the interface, include the **encapsulation** statement and specify the **ethernet-ccc** option. Include the **peer-unit** statement and specify the logical interface unit **1** as the peer tunnel interface. Include the **family** statement and specify the **ccc** option.

```
[edit]
interfaces {
  lt-1/1/10 {
    unit 0 {
      encapsulation ethernet-ccc;
      peer-unit 1;
      family ccc;
    }
  }
}
```

8. On Router PE3, configure the **lt-1/1/10.1** logical tunnel interface at the **[edit interfaces lt-1/1/10 unit 1]** hierarchy level.

   To configure the interface, include the **encapsulation** statement and specify the **ethernet** option. Include the **peer-unit** statement and specify the logical interface unit **0** as the peer tunnel interface. Include the **family** statement and specify the **inet** option. Include the **address** statement at the **[edit interfaces lt-1/1/10 unit 0]** hierarchy level and specify **70.70.70.1/24** as the IPv4 address.

```
[edit]
interfaces {
  lt-1/1/10 {
    unit 1 {
      encapsulation ethernet;
      peer-unit 0;
      family inet {
```

```
            address 70.70.70.1/24;
          }
        }
      }
    }
```

9. On Router PE3, add the **lt** interface unit 1 to the routing instance at the **[edit routing-instances L3VPN]** hierarchy level. Configure the instance type as **vrf** with **lt** peer-unit 1 as a PE-CE interface to terminate the Layer 2 VPN on Router PE2 into the Layer 3 VPN on Router PE3.

```
[edit]
routing-instances {
  L3VPN {
    instance-type vrf;
    interface ge-1/0/1.0;
    interface lt-1/1/10.1;
    route-distinguisher 65000:33;
    vrf-target target:65000:2;
    vrf-table-label;
    protocols {
      bgp {
        export direct;
        group ce3 {
          neighbor 90.90.90.2 {
            peer-as 100;
          }
        }
      }
    }
  }
}
```

10. On Router PE3, add the **lt** interface unit 0 to the routing instance at the **[edit routing-instances protocols l2vpn]** hierarchy level. Also configure the same vrf target for the Layer 2 VPN and Layer 3 VPN routing instances, so that the routes can be leaked between the instances. The example configuration in the previous step shows the vrf target for the **L3VPN** routing instance. The following example shows the vrf target for the **l2vpn** routing instance.

```
[edit]
routing-instances {
  l2vpn {
    instance-type l2vpn;
    interface lt-1/1/10.0;
    route-distinguisher 65000:3;
    vrf-target target:65000:2;
    protocols {
      l2vpn {
        encapsulation-type ethernet;
        site CE3 {
          site-identifier 3;
          interface lt-1/1/10.0 {
            remote-site-id 2;
          }
        }
```

```
        }
      }
    }
  }
```

11. On Router PE3, configure the **policy-statement** statement to export the routes learned from the directly connected **lt** interface unit 1 to all the CE routers for connectivity, if needed.

```
[edit]
policy-options {
  policy-statement direct {
    term 1 {
      from protocol direct;
      then accept;
    }
  }
}
```

Results The following output shows the full configuration of Router PE2:

Router PE2

```
interfaces {
  xe-0/1/0 {
    unit 0 {
      family inet {
        address 10.10.2.2/30;
      }
      family mpls;
    }
  }
  xe-0/2/0 {
    unit 0 {
      family inet {
        address 10.10.5.1/30;
      }
      family mpls;
    }
  }
  xe-0/3/0 {
    unit 0 {
      family inet {
        address 10.10.4.1/30;
      }
      family mpls;
    }
  }
  ge-1/0/2 {
    encapsulation ethernet-ccc;
    unit 0;
  }
  fxp0 {
    apply-groups [ re0 re1 ];
  }
  lo0 {
    unit 0 {
      family inet {
```

```
            address 2.2.2.2/32;
          }
        }
      }
    }
    routing-options {
      static {
        route 172.0.0.0/8 next-hop 172.19.59.1;
      }
      autonomous-system 65000;
    }
    protocols {
      mpls {
        interface all;
        interface fxp0.0 {
          disable;
        }
      }
      bgp {
        group RR {
          type internal;
          local-address 2.2.2.2;
          family l2vpn {
            signaling;
          }
          neighbor 7.7.7.7;
        }
      }
      ospf {
        traffic-engineering;
        area 0.0.0.0 {
          interface all;
          interface fxp0.0 {
            disable;
          }
        }
      }
      ldp {
        interface all;
        interface fxp0.0 {
          disable;
        }
      }
    }
    routing-instances {
      l2vpn {
        instance-type l2vpn;
        interface ge-1/0/2.0;
        route-distinguisher 65000:2;
        vrf-target target:65000:2;
        protocols {
          l2vpn {
            encapsulation-type ethernet;
            site CE2 {
              site-identifier 2;
              interface ge-1/0/2.0 {
```

```
            remote-site-id 3;
          }
        }
      }
    }
  }
}
```

The following output shows the final configuration of Router PE5:

Router PE5

```
interfaces {
  ge-0/0/0 {
    unit 0 {
      family inet {
        address 10.10.4.2/30;
      }
      family mpls;
    }
  }
  xe-0/1/0 {
    unit 0 {
      family inet {
        address 10.10.6.2/30;
      }
      family mpls;
    }
  }
  ge-1/0/0 {
    unit 0 {
      family inet {
        address 10.10.9.1/30;
      }
      family mpls;
    }
  }
  xe-1/1/0 {
    unit 0 {
      family inet {
        address 10.10.3.2/30;
      }
      family mpls;
    }
  }
  ge-2/0/0 {
    unit 0 {
      family inet {
        address 80.80.80.1/24;
      }
    }
  }
  lo0 {
    unit 0 {
      family inet {
        address 5.5.5.5/32;
      }
    }
```

```
    }
  }
  routing-options {
    static {
      route 172.0.0.0/8 next-hop 172.19.59.1;
    }
    autonomous-system 65000;
  }
  protocols {
    rsvp {
      interface all {
        link-protection;
      }
      interface fxp0.0 {
        disable;
      }
    }
    mpls {
      label-switched-path to-RR {
        to 7.7.7.7;
      }
      label-switched-path to-PE2 {
        to 2.2.2.2;
      }
      label-switched-path to-PE3 {
        to 3.3.3.3;
      }
      label-switched-path to-PE4 {
        to 4.4.4.4;
      }
      label-switched-path to-PE1 {
        to 1.1.1.1;
      }
      interface all;
      interface fxp0.0 {
        disable;
      }
    }
    bgp {
      group to-rr {
        type internal;
        local-address 5.5.5.5;
        family inet-vpn {
          unicast;
        }
        family l2vpn {
          signaling;
        }
        neighbor 7.7.7.7;
      }
    }
    ospf {
      traffic-engineering;
      area 0.0.0.0 {
        interface all;
        interface fxp0.0 {
```

```
            disable;
          }
        }
      }
      ldp {
        interface all;
        interface fxp0.0 {
          disable;
        }
      }
    }
    routing-instances {
      L3VPN {
        instance-type vrf;
        interface ge-2/0/0.0;
        route-distinguisher 65000:5;
        vrf-target target:65000:2;
        vrf-table-label;
        protocols {
          bgp {
            group ce5 {
              neighbor 80.80.80.2 {
                peer-as 200;
              }
            }
          }
        }
      }
    }
```

The following output shows the final configuration of Router PE3:

Router PE3

```
chassis {
  dump-on-panic;
  fpc 1 {
    pic 1 {
      tunnel-services {
        bandwidth 1g;
      }
    }
  }
  network-services ip;
}
interfaces {
  ge-1/0/1 {
    unit 0 {
      family inet {
        address 90.90.90.1/24;
      }
    }
  }
  lt-1/1/10 {
    unit 0 {
      encapsulation ethernet-ccc;
      peer-unit 1;
      family ccc;
```

```
        }
        unit 1 {
            encapsulation ethernet;
            peer-unit 0;
            family inet {
                address 70.70.70.1/24;
            }
        }
    }
    xe-2/0/0 {
        unit 0 {
            family inet {
                address 10.10.20.2/30;
            }
            family mpls;
        }
    }
    xe-2/1/0 {
        unit 0 {
            family inet {
                address 10.10.6.1/30;
            }
            family mpls;
        }
    }
    xe-2/2/0 {
        unit 0 {
            family inet {
                address 10.10.5.2/30;
            }
            family mpls;
        }
    }
    xe-2/3/0 {
        unit 0 {
            family inet {
                address 10.10.1.2/30;
            }
            family mpls;
        }
    }
    lo0 {
        unit 0 {
            family inet {
                address 3.3.3.3/32;
            }
        }
    }
}
routing-options {
    static {
        route 172.0.0.0/8 next-hop 172.19.59.1;
    }
    autonomous-system 65000;
}
protocols {
```

```
rsvp {
    interface all;
    interface fxp0.0 {
        disable;
    }
}
mpls {
    label-switched-path to-RR {
        to 7.7.7.7;
    }
    }
    label-switched-path to-PE2 {
        to 2.2.2.2;
    }
    label-switched-path to-PE5 {
        to 5.5.5.5;
    }
    label-switched-path to-PE4 {
        to 4.4.4.4;
    }
    label-switched-path to-PE1 {
        to 1.1.1.1;
    }
    interface all;
    interface fxp0.0 {
        disable;
    }
}
bgp {
    group RR {
        type internal;
        local-address 3.3.3.3;
        family inet-vpn {
            unicast;
        }
        family l2vpn {
            signaling;
        }
        neighbor 7.7.7.7;
    }
}
ospf {
    traffic-engineering;
    area 0.0.0.0 {
        interface all;
        interface fxp0.0 {
            disable;
        }
    }
}
ldp {
    interface all;
    interface fxp0.0 {
        disable;
    }
}
```

```
}
policy-options {
   policy-statement direct {
      term 1 {
         from protocol direct;
         then accept;
      }
   }
}
routing-instances {
   L3VPN {
      instance-type vrf;
      interface ge-1/0/1.0;
      interface lt-1/1/10.1;
      route-distinguisher 65000:33;
      vrf-target target:65000:2;
      vrf-table-label;
      protocols {
         bgp {
            export direct;
            group ce3 {
               neighbor 90.90.90.2 {
                  peer-as 100;
               }
            }
         }
      }
   }
   l2vpn {
      instance-type l2vpn;
      interface lt-1/1/10.0;
      route-distinguisher 65000:3;
      vrf-target target:65000:2;
      protocols {
         l2vpn {
            encapsulation-type ethernet;
            site CE3 {
               site-identifier 3;
               interface lt-1/1/10.0 {
                  remote-site-id 2;
               }
            }
         }
      }
   }
}
```

Verification

Verify the Layer 2 VPN-to-Layer 3 VPN interconnection:

Verifying Router PE2 VPN Interface

Purpose Check that the Layer 2 VPN is up and working at the Router PE2 interface and that all the routes are there.

Action

1. Use the **show l2vpn connections** command to verify that the connection site ID is 3 for Router PE3 and that the status is **Up**.

```
user@PE2> show l2vpn connections

Layer-2 VPN connections:
Legend for connection status (St)
EI -- encapsulation invalid      NC -- interface encapsulation not
CCC/TCC/VPLS
EM -- encapsulation mismatch     WE -- interface and instance encaps not
 same
VC-Dn -- Virtual circuit down    NP -- interface hardware not present
CM -- control-word mismatch      -> -- only outbound connection is up
CN -- circuit not provisioned    <- -- only inbound connection is up
OR -- out of range               Up -- operational
OL -- no outgoing label          Dn -- down
LD -- local site signaled down   CF -- call admission control failure
RD -- remote site signaled down  SC -- local and remote site ID collision
LN -- local site not designated  LM -- local site ID not minimum designated
RN -- remote site not designated RM -- remote site ID not minimum
designated
XX -- unknown connection status  IL -- no incoming label
MM -- MTU mismatch               MI -- Mesh-Group ID not available
BK -- Backup connection          ST -- Standby connection
PF -- Profile parse failure      PB -- Profile busy
RS -- remote site standby

Legend for interface status
Up -- operational
Dn -- down

Instance: l2vpn
  Local site: CE2 (2)
    connection-site     Type  St     Time last up        # Up trans
    3                   rmt   Up     Jan  7 14:14:37 2010         1
      Remote PE: 3.3.3.3, Negotiated control-word: Yes (Null)
      Incoming label: 800000, Outgoing label: 800001
      Local interface: ge-1/0/2.0, Status: Up, Encapsulation: ETHERNET
```

2. Use the **show route table** command to verify that the Layer 2 VPN route is present and that there is a next hop of **10.10.5.2** through the **xe-0/2/0.0** interface. The following

output verifies that the Layer 2 VPN routes are present in the l2vpn.l2vpn.0 table. Similar output should be displayed for Router PE3.

```
user@PE2> show route table l2vpn.l2vpn.0

l2vpn.l2vpn.0: 2 destinations, 2 routes (2 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

65000:2:2:3/96
                   *[L2VPN/170/-101] 02:40:35, metric2 1
                      Indirect
65000:3:3:1/96
                   *[BGP/170] 02:40:35, localpref 100, from 7.7.7.7
                      AS path: I
                    > to 10.10.5.2 via xe-0/2/0.0
```

3. Verify that Router PE2 has a Layer 2 VPN MPLS label pointing to the LDP label to Router PE3 in both directions (PUSH and POP).

```
user@PE2> show route table mpls.0

 mpls.0: 13 destinations, 13 routes (13 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

0                  *[MPLS/0] 1w3d 08:57:41, metric 1
                      Receive
1                  *[MPLS/0] 1w3d 08:57:41, metric 1
                      Receive
2                  *[MPLS/0] 1w3d 08:57:41, metric 1
                      Receive
300560             *[LDP/9] 19:45:53, metric 1
                    > to 10.10.2.1 via xe-0/1/0.0, Pop
300560(S=0)        *[LDP/9] 19:45:53, metric 1
                    > to 10.10.2.1 via xe-0/1/0.0, Pop
301008             *[LDP/9] 19:45:53, metric 1
                    > to 10.10.4.2 via xe-0/3/0.0, Swap 299856
301536             *[LDP/9] 19:45:53, metric 1
                    > to 10.10.4.2 via xe-0/3/0.0, Pop
301536(S=0)        *[LDP/9] 19:45:53, metric 1
                    > to 10.10.4.2 via xe-0/3/0.0, Pop
301712             *[LDP/9] 16:14:52, metric 1
                    > to 10.10.5.2 via xe-0/2/0.0, Swap 315184
301728             *[LDP/9] 16:14:52, metric 1
                    > to 10.10.5.2 via xe-0/2/0.0, Pop
301728(S=0)        *[LDP/9] 16:14:52, metric 1
                    > to 10.10.5.2 via xe-0/2/0.0, Pop
800000       *[L2VPN/7] 02:40:35
         > via ge-1/0/2.0, Pop     Offset: 4
ge-1/0/2.0     *[L2VPN/7] 02:40:35, metric2 1
         > to 10.10.5.2 via xe-0/2/0.0, Push 800001 Offset: -4
```

Meaning The **l2vpn** routing instance is up at interface **ge-1/0/2** and the Layer 2 VPN route is shown in table l2vpn.l2vpn.0. Table **mpls.0** shows the Layer 2 VPN routes used to forward the traffic using an LDP label.

Verifying Router PE3 VPN Interface

Purpose Check that the Layer 2 VPN connection from Router PE2 and Router PE3 is **Up** and working.

Action 1. Verify that the BGP session with the route reflector for the family **l2vpn-signaling** and the family **inet-vpn** is established.

```
user@PE3> show bgp summary

Groups: 2 Peers: 2 Down peers: 0
Table        Tot Paths  Act Paths Suppressed    History Damp State    Pending
bgp.l2vpn.0          1          1          0          0          0          0
bgp.L3VPN.0          1          1          0          0          0          0
Peer        AS   InPkt    OutPkt   OutQ   Flaps Last Up/Dwn   State|#Active /Received/Accepted/Damped...
7.7.7.7  65000    2063      2084      0       1    15:35:16   Establ
  bgp.l2vpn.0: 1/1/1/0
  bgp.L3VPN.0: 1/1/1/0
  L3VPN.inet.0: 1/1/1/0
  l2vpn.l2vpn.0: 1/1/1/0
```

2. The following output shows the L3VPN.inet.0 routing table, which has Routers CE1, CE3, and CE5 listed.

```
user@PE3> show route table L3VPN.inet.0

L3VPN.inet.0: 5 destinations, 5 routes (5 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

70.70.70.0/24      *[Direct/0] 02:45:16
                    > via lt-1/1/10.1
70.70.70.1/32      *[Local/0] 14:45:42
                       Local via lt-1/1/10.1
80.80.80.0/24      *[BGP/170] 02:47:51, localpref 100, from 7.7.7.7
                       AS path: I
                    > to 10.10.6.2 via xe-2/1/0.0, Push 16
90.90.90.0/24      *[Direct/0] 15:26:24
                    > via ge-1/0/1.0
90.90.90.1/32      *[Local/0] 15:26:24
                       Local via ge-1/0/1.0
```

3. The following output verifies the Layer 2 VPN route and the label associated with it.

```
user@PE3> show route table l2vpn.l2vpn.0 detail

l2vpn.l2vpn.0: 2 destinations, 2 routes (2 active, 0 holddown, 0 hidden)
65000:2:2:3/96 (1 entry, 1 announced)
        *BGP    Preference: 170/-101
                Route Distinguisher: 65000:2
                Next hop type: Indirect
                Next-hop reference count: 4
                Source: 7.7.7.7
                Protocol next hop: 2.2.2.2
                Indirect next hop: 2 no-forward
                State: <Secondary Active Int Ext>
                Local AS: 65000 Peer AS: 65000
                Age: 2:45:52    Metric2: 1
                Task: BGP_65000.7.7.7.7+60585
                Announcement bits (1): 0-l2vpn-l2vpn
                AS path: I (Originator) Cluster list:  7.7.7.7
                AS path:  Originator ID: 2.2.2.2
                Communities: target:65000:2 Layer2-info: encaps:ETHERNET,
 control flags:Control-Word, mtu: 0, site preference: 100  Accepted
                Label-base: 800000, range: 2, status-vector: 0x0
                Localpref: 100
```

```
                    Router ID: 7.7.7.7
                    Primary Routing Table bgp.l2vpn.0
```

4. The following output show the L2VPN MPLS.0 route in the mpls.0 route table.

```
user@PE3> show route table mpls.0

mpls.0: 21 destinations, 21 routes (21 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

0                  *[MPLS/0] 1w3d 09:05:41, metric 1
                      Receive
1                  *[MPLS/0] 1w3d 09:05:41, metric 1
                      Receive
2                  *[MPLS/0] 1w3d 09:05:41, metric 1
                      Receive
16                 *[VPN/0] 15:59:24
                      to table L3VPN.inet.0, Pop
315184             *[LDP/9] 16:21:53, metric 1
                    > to 10.10.20.1 via xe-2/0/0.0, Pop
315184(S=0)        *[LDP/9] 16:21:53, metric 1
                    > to 10.10.20.1 via xe-2/0/0.0, Pop
315200             *[LDP/9] 01:13:44, metric 1
                      to 10.10.20.1 via xe-2/0/0.0, Swap 625297
                    > to 10.10.6.2 via xe-2/1/0.0, Swap 299856
315216             *[LDP/9] 16:21:53, metric 1
                    > to 10.10.6.2 via xe-2/1/0.0, Pop
315216(S=0)        *[LDP/9] 16:21:53, metric 1
                    > to 10.10.6.2 via xe-2/1/0.0, Pop
315232             *[LDP/9] 16:21:45, metric 1
                    > to 10.10.1.1 via xe-2/3/0.0, Pop
315232(S=0)        *[LDP/9] 16:21:45, metric 1
                    > to 10.10.1.1 via xe-2/3/0.0, Pop
315248             *[LDP/9] 16:21:53, metric 1
                    > to 10.10.5.1 via xe-2/2/0.0, Pop
315248(S=0)        *[LDP/9] 16:21:53, metric 1
                    > to 10.10.5.1 via xe-2/2/0.0, Pop
315312             *[RSVP/7] 15:02:40, metric 1
                    > to 10.10.6.2 via xe-2/1/0.0, label-switched-path
to-pe5
315312(S=0)        *[RSVP/7] 15:02:40, metric 1
                    > to 10.10.6.2 via xe-2/1/0.0, label-switched-path
to-pe5
315328             *[RSVP/7] 15:02:40, metric 1
                    > to 10.10.20.1 via xe-2/0/0.0, label-switched-path
to-RR
315360             *[RSVP/7] 15:02:40, metric 1
                    > to 10.10.20.1 via xe-2/0/0.0, label-switched-path
to-RR
316272             *[RSVP/7] 01:13:27, metric 1
                    > to 10.10.6.2 via xe-2/1/0.0, label-switched-path
Bypass->10.10.9.1
316272(S=0)        *[RSVP/7] 01:13:27, metric 1
                    > to 10.10.6.2 via xe-2/1/0.0, label-switched-path
Bypass->10.10.9.1
800001             *[L2VPN/7] 02:47:33
                    > via lt-1/1/10.0, Pop       Offset: 4
lt-1/1/10.0        *[L2VPN/7] 02:47:33, metric2 1
                    > to 10.10.5.1 via xe-2/2/0.0, Push 800000 Offset: -4
```

5. Use the **show route table mpls.0** command with the **detail** option to see the BGP attributes of the route such as next-hop type and label operations.

```
user@PE5> show route table mpls.0 detail

lt-1/1/10.0 (1 entry, 1 announced)
        *L2VPN  Preference: 7
                Next hop type: Indirect
                Next-hop reference count: 2
                Next hop type: Router, Next hop index: 607
                Next hop: 10.10.5.1 via xe-2/2/0.0, selected
                Label operation: Push 800000 Offset: -4
                Protocol next hop: 2.2.2.2
                Push 800000 Offset: -4
                Indirect next hop: 8cae0a0 1048574
                State:  <Active Int>
                Age: 2:46:34    Metric2: 1
                Task: Common L2 VC
                Announcement bits (2): 0-KRT 2-Common L2 VC
                AS path: I
                Communities: target:65000:2 Layer2-info: encaps:ETHERNET,
 control flags:Control-Word, mtu: 0, site preference: 100
```

Verifying End-to-End connectivity from Router CE2 to Router CE5 and Router CE3

Purpose Check the connectivity between Routers CE2, CE3, and CE5.

Action

1. Ping the Router CE3 IP address from Router CE2.

```
user@CE2> ping 90.90.90.2 # CE3 IP address

PING 90.90.90.2 (90.90.90.2): 56 data bytes
64 bytes from 90.90.90.2: icmp_seq=0 ttl=63 time=0.708 ms
64 bytes from 90.90.90.2: icmp_seq=1 ttl=63 time=0.610 ms
```

2. Ping the Router CE5 IP address from Router CE2.

```
user@CE2> ping 80.80.80.2 # CE5 IP address

PING 80.80.80.2 (80.80.80.2): 56 data bytes
64 bytes from 80.80.80.2: icmp_seq=0 ttl=62 time=0.995 ms
64 bytes from 80.80.80.2: icmp_seq=1 ttl=62 time=1.005 ms
```

Related Documentation

- Layer 2 VPN Overview
- Layer 3 VPN Overview on page 294
- Interconnecting Layer 2 VPNs with Layer 3 VPNs Overview on page 296

Example: Interconnecting a Layer 2 Circuit with a Layer 3 VPN

This example provides a step-by-step procedure and commands for configuring and verifying a Layer 2 circuit to Layer 3 VPN interconnection. It contains the following sections:

- Requirements on page 322
- Overview and Topology on page 322

Requirements

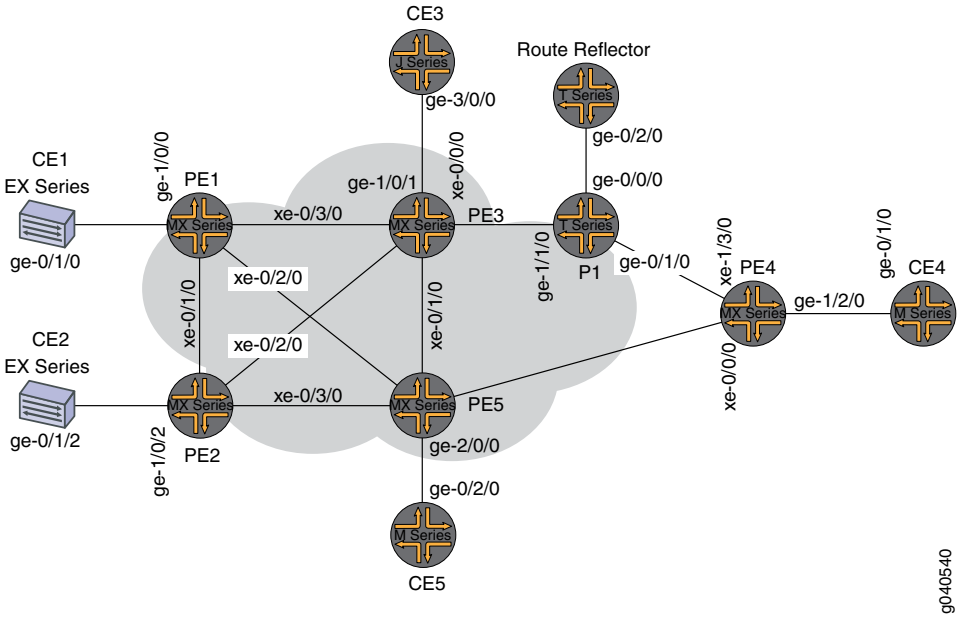
This example uses the following hardware and software components:

- Junos OS Release 9.3 or later
- 3 MX Series routers
- 1 M Series routers
- 1 T Series router
- 1 EX Series router
- 1 J Series router

Overview and Topology

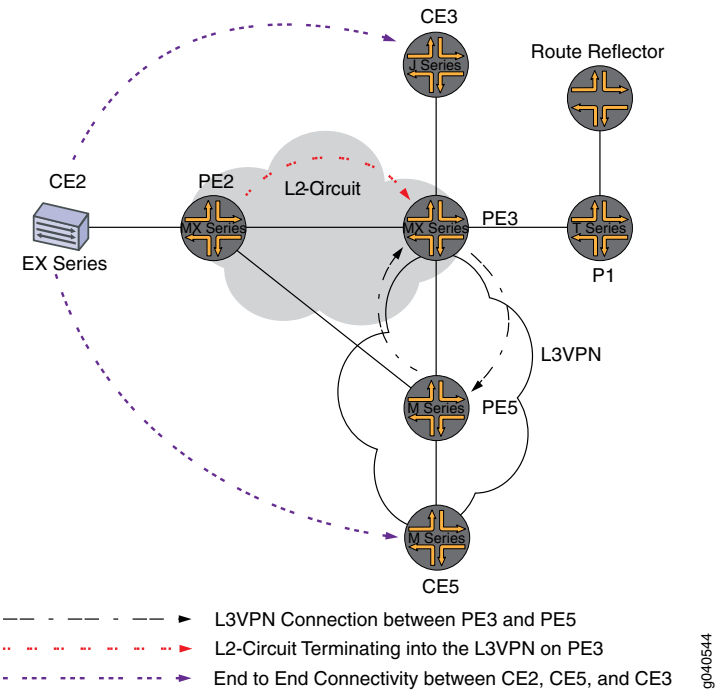
The physical topology of a Layer 2 circuit to Layer 3 VPN interconnection is shown in Figure 46 on page 322.

Figure 46: Physical Topology of a Layer 2 Circuit to Layer 3 VPN Interconnection



The logical topology of a Layer 2 circuit to Layer 3 VPN interconnection is shown in Figure 47 on page 323.

Figure 47: Logical Topology of a Layer 2 Circuit to Layer 3 VPN Interconnection



Configuration

NOTE: In any configuration session, it is good practice to verify periodically that the configuration can be committed using the commit check command.

In this example, the router being configured is identified using the following command prompts:

- **CE2** identifies the customer edge 2 (CE2) router
- **PE1** identifies the provider edge 1 (PE1) router
- **CE3** identifies the customer edge 3 (CE3) router
- **PE3** identifies the provider edge 3 (PE3) router
- **CE5** identifies the customer edge 5 (CE5) router
- **PE5** identifies the provider edge 5 (PE5) router

This example contains the following procedures:

- Configuring PE Router Customer-facing and Loopback Interfaces on page 324
- Configuring Core-facing Interfaces on page 325
- Configuring Protocols on page 327
- Configuring Routing Instances and Layer 2 Circuits on page 329

- Configuring the Route Reflector on page 331
- Interconnecting the Layer 2 Circuit with the Layer 3 VPN on page 332

Configuring PE Router Customer-facing and Loopback Interfaces

Step-by-Step Procedure

To begin building the interconnection, configure the interfaces on the PE routers. If your network contains provider (P) routers, configure the interfaces on the P routers also. This example shows the configuration for Router PE2, Router PE3, and Router PE5.

1. On Router PE2, configure the **ge-1/0/2** interface encapsulation. To configure the interface encapsulation, include the **encapsulation** statement and specify the **ethernet-ccc** option (**vlan-ccc** encapsulation is also supported). Configure the **ge-1/0/2.0** logical interface family for circuit cross-connect functionality. To configure the logical interface family, include the **family** statement and specify the **ccc** option. The encapsulation should be configured the same way for all routers in the Layer 2 circuit domain.

   ```
   [edit interfaces]
   ge-1/0/2 {
     encapsulation ethernet-ccc;
     unit 0 {
       family ccc;
     }
   }
   ```

2. On Router PE2, configure the **lo0.0** interface. Include the **family** statement and specify the **inet** option. Include the **address** statement and specify **2.2.2.2/32** as the loopback IPv4 address.

   ```
   [edit interfaces]
   lo0 {
     unit 0 {
       family inet {
         address 2.2.2.2/32;
       }
     }
   }
   ```

3. On Router PE3, configure the **ge-1/0/1** interface. Include the **family** statement and specify the **inet** option. Include the **address** statement and specify **90.90.90.1/24** as the interface address for this device.

   ```
   [edit interfaces]
   ge-1/0/1 {
     unit 0 {
       family inet {
         address 90.90.90.1/24;
       }
     }
   }
   ```

4. On Router PE3, configure the **lo0.0** loopback interface. Include the **family** statement and specify the **inet** option. Include the **address** statement and specify **3.3.3.3/32** as the loopback IPv4 address for this router.

   ```
   [edit interfaces]
   ```

```
lo0 {
  unit 0 {
    family inet {
      address 3.3.3.3/32;
    }
  }
}
```

5. On Router PE5, configure the **ge-2/0/0** interface. Include the **family** statement and specify the **inet** option. Include the **address** statement and specify **80.80.80.1/24** as the interface address.

```
[edit interfaces]
ge-2/0/0 {
  unit 0 {
    family inet {
      address 80.80.80.1/24;
    }
  }
}
```

6. On Router PE5, configure the **lo0.0** interface. Include the **family** statement and specify the **inet** option. Include the **address** statement and specify **5.5.5.5/32** as the loopback IPv4 address for this router.

```
[edit interfaces]
lo0 {
  unit 0 {
    family inet {
      address 5.5.5.5/32;
    }
  }
}
```

Configuring Core-facing Interfaces

Step-by-Step Procedure

This procedure describes how to configure the core-facing interfaces on the PE routers. This example does not include all the core-facing interfaces shown in the physical topology illustration. Enable the **mpls** and **inet** address families on the core-facing interfaces.

1. On Router PE2, configure the **xe-0/2/0** interface. Include the **family** statement and specify the **inet** address family. Include the **address** statement and specify **10.10.5.1/30** as the interface address. Include the **family** statement and specify the **mpls** address family.

```
[edit interfaces]
xe-0/2/0 {
  unit 0 {
    family inet {
      address 10.10.5.1/30;
    }
    family mpls;
  }
}
```

2. On Router PE3, configure the core-facing interfaces. Include the **family** statement and specify the **inet** address family. Include the **address** statement and specify the IPv4 addresses shown in the example as the interface addresses. Include the **family** statement and specify the **mpls** address family. In the example, the **xe-2/1/0** interface is connected to Router PE5, and the **xe-2/2/0** interface is connected to Router PE2.

```
[edit interfaces]
xe-2/0/0 {
  unit 0 {
    family inet {
      address 10.10.20.2/30;
    }
    family mpls;
  }
}
xe-2/1/0 {
  unit 0 {
    family inet {
      address 10.10.6.1/30;
    }
    family mpls;
  }
}
xe-2/2/0 {
  unit 0 {
    family inet {
      address 10.10.5.2/30;
    }
    family mpls;
  }
}
xe-2/3/0 {
  unit 0 {
    family inet {
      address 10.10.1.2/30;
    }
    family mpls;
  }
}
```

3. On Router PE5, configure the **xe-0/1/0** interface. Include the **family** statement and specify the **inet** address family. Include the **address** statement and specify **10.10.6.2/30** as the interface address. Include the **family** statement and specify the **mpls** address family.

```
[edit interfaces]
xe-0/1/0 {
  unit 0 {
    family inet {
      address 10.10.6.2/30;
    }
    family mpls;
  }
}
```

Configuring Protocols

Step-by-Step Procedure

This procedure describes how to configure the protocols used in this example. If your network contains P routers, configure the interfaces on the P routers also.

1. On Router PE3, enable OSPF as the IGP. Enable the MPLS, LDP, and BGP protocols on all interfaces except **fxp.0**. LDP is used as the signaling protocol for the Layer 2 circuit to Router PE2 . The following configuration snippet shows the protocol configuration for Router PE3:

```
[edit]
protocols {
  rsvp {
    interface all;
    interface fxp0.0 {
      disable;
    }
  }
  mpls {
    label-switched-path to-RR {
      to 7.7.7.7;
    }
    label-switched-path to-PE2 {
      to 2.2.2.2;
    }
    label-switched-path to-PE5 {
      to 5.5.5.5;
    }
    label-switched-path to-PE4 {
      to 4.4.4.4;
    }
    label-switched-path to-PE1 {
      to 1.1.1.1;
    }
    interface all;
    interface fxp0.0 {
      disable;
    }
  }
  bgp {
    group RR {
      type internal;
      local-address 3.3.3.3;
      family inet-vpn {
        unicast;
      }
      family l2vpn {
        signaling;
      }
      neighbor 7.7.7.7;
    }
  }
  ospf {
    traffic-engineering;
    area 0.0.0.0 {
```

```
        interface all;
        interface fxp0.0 {
          disable;
        }
      }
    }
    ldp {
      interface all;
      interface fxp0.0 {
        disable;
      }
    }
  }
```

2. On Router PE2, configure the MPLS, OSPF, and LDP protocols.

```
[edit ]
protocols {
  mpls {
    interface all;
    interface fxp0.0 {
      disable;
    }
  }
  ospf {
    traffic-engineering;
    area 0.0.0.0 {
      interface all;
      interface fxp0.0 {
        disable;
      }
    }
  }
  ldp {
    interface all;
    interface fxp0.0 {
      disable;
    }
  }
}
```

3. On Router PE5, enable OSPF as the IGP. Enable the MPLS, RSVP, and BGP protocols on all interfaces except **fxp.0**. Enable core-facing interfaces with the **mpls** and **inet** address families.

```
[edit]
protocols {
  rsvp {
    interface all {
      link-protection;
    }
    interface fxp0.0 {
      disable;
    }
  }
  mpls {
    label-switched-path to-RR {
```

```
        to 7.7.7.7;
      }
      label-switched-path to-PE2 {
        to 2.2.2.2;
      }
      label-switched-path to-PE3 {
        to 3.3.3.3;
      }
      label-switched-path to-PE4 {
        to 4.4.4.4;
      }
      label-switched-path to-PE1 {
        to 1.1.1.1;
      }
      interface all;
      interface fxp0.0 {
        disable;
      }
    }
    bgp {
      group to-rr {
        type internal;
        local-address 5.5.5.5;
        family inet-vpn {
          unicast;
        }
        family l2vpn {
          signaling;
        }
        neighbor 7.7.7.7;
      }
    }
    ospf {
      traffic-engineering;
      area 0.0.0.0 {
        interface all;
        interface fxp0.0 {
          disable;
        }
      }
    }
  }
```

Configuring Routing Instances and Layer 2 Circuits

Step-by-Step Procedure

This procedure describes how to configure the Layer 2 circuit and the Layer 3 VPN.

1. On Router PE2, configure the Layer 2 circuit. Include the **l2circuit** statement. Include the **neighbor** statement and specify the loopback IPv4 address of Router PE3 as the neighbor. Include the interface statement and specify **ge-1/0/2.0** as the logical interface that is participating in the Layer 2 circuit. Include the **virtual-circuit-id** statement and specify **100** as the identifier. Include the **no-control-word** statement for equipment that does not support the control word.

   [edit]

```
protocols {
  l2circuit {
    neighbor 3.3.3.3 {
      interface ge-1/0/2.0 {
        virtual-circuit-id 100;
        no-control-word;
      }
    }
  }
}
```

2. On Router PE3, configure the Layer 2 circuit to Router PE2. Include the **l2circuit** statement. Include the **neighbor** statement and specify the loopback IPv4 address of Router PE2 as the neighbor. Include the interface statement and specify **lt-1/1/10.0** as the logical tunnel interface that is participating in the Layer 2 circuit. Include the **virtual-circuit-id** statement and specify **100** as the identifier. Include the **no-control-word** statement.

```
[edit ]
protocols {
  l2circuit {
    neighbor 2.2.2.2 {
      interface lt-1/1/10.0 {
        virtual-circuit-id 100;
        no-control-word;
      }
    }
  }
}
```

3. On Router PE3, configure the Layer 3 VPN (**L3VPN**) routing instance to Router PE5 at the **[edit routing-instances]** hierarchy level. Also configure the BGP peer group at the **[edit routing-instances L3VPN protocols]** hierarchy level.

```
[edit ]
routing-instances {
  L3VPN {
    instance-type vrf;
    interface ge-1/0/1.0;
    interface lt-1/1/10.1;
    route-distinguisher 65000:33;
    vrf-target target:65000:2;
    vrf-table-label;
    protocols {
      bgp {
        export direct;
        group ce3 {
          neighbor 90.90.90.2 {
            peer-as 100;
          }
        }
      }
    }
  }
}
```

4. On Router PE5, configure the Layer 3 VPN routing instance (**L3VPN**) at the **[edit routing-instances]** hierarchy level. Also configure the BGP peer group at the **[edit routing-instances L3VPN protocols]** hierarchy level.

```
[edit ]
routing-instances {
  L3VPN {
    instance-type vrf;
    interface ge-2/0/0.0;
    route-distinguisher 65000:5;
    vrf-target target:65000:2;
    vrf-table-label;
    protocols {
      bgp {
        group ce5 {
          neighbor 80.80.80.2 {
            peer-as 200;
          }
        }
      }
    }
  }
}
```

Configuring the Route Reflector

Step-by-Step Procedure

Although a route reflector is not required to interconnect a Layer 2 circuit with a Layer 3 VPN, this examples uses a route reflector. This procedure shows the relevant portion of the route reflector configuration.

1. Configure the route reflector with RSVP, MPLS, BGP and OSPF. The route reflector is a BGP peer with the PE routers. Notice that the BGP peer group configuration includes the **family** statement and specifies the **inet-vpn** option The **inet-vpn** option enables BGP to advertise network layer reachability information (NLRI) for the Layer 3 VPN routes. The configuration also includes the **family** statement and specifies the **l2vpn** option. The **l2vpn** option enables BGP to advertise NLRI for the Layer 2 circuit. Layer 2 circuits use the same internal BGP infrastructure as Layer 2 VPNs.

```
[edit ]
protocols {
  rsvp {
    interface all;
    interface fxp0.0 {
      disable;
    }
  }
  mpls {
    label-switched-path to-pe3 {
      to 3.3.3.3;
    }
    label-switched-path to-pe5 {
      to 5.5.5.5;
    }
    interface all;
```

```
    interface fxp0.0 {
      disable;
    }
  }
  bgp {
    group RR {
      type internal;
      local-address 7.7.7.7;
      family inet {
        unicast;
      }
      family inet-vpn {
        unicast;
      }
      family l2vpn {
        signaling;
      }
      cluster 7.7.7.7;
      neighbor 1.1.1.1;
      neighbor 2.2.2.2;
      neighbor 4.4.4.4;
      neighbor 5.5.5.5;
      neighbor 3.3.3.3;
    }
  }
  ospf {
    traffic-engineering;
    area 0.0.0.0 {
      interface all;
      interface fxp0.0 {
        disable;
      }
    }
  }
}
```

Interconnecting the Layer 2 Circuit with the Layer 3 VPN

Step-by-Step Procedure

Before you can configure the logical tunnel interface in an MX Series router, you must create the tunnel services interface to be used for tunnel services.

1. Create the tunnel service interface on Router PE3. Include the **bandwidth** statement at the **[edit chassis fpc *slot-number* pic *slot-number* tunnel-services]** hierarchy level and specify the amount of bandwidth to reserve for tunnel services in gigabits per second.

```
[edit chassis]
fpc 1 {
  pic 1 {
    tunnel-services {
      bandwidth 1g;
    }
  }
}
```

2. On Router PE3, configure the **lt-1/1/10** logical tunnel interface unit 0.

   Router PE3 is the router that is *stitching* the Layer 2 circuit to the Layer 3 VPN using the logical tunnel interface. The configuration of the peer unit interfaces is what makes the interconnection.

   Include the **encapsulation** statement and specify the **ethernet-ccc** option. Include the **peer-unit** statement and specify the logical interface unit **1** as the peer tunnel interface. Include the **family** statement and specify the **ccc** option.

   Configure the **lt-1/1/10** logical interface unit **1** with **ethernet** encapsulation. Include the **peer-unit** statement and specify the logical interface unit **0** as the peer tunnel interface. Include the **family** statement and specify the **inet** option. Also include the **address** statement and specify **70.70.70.1/24** as the IPv4 address of the interface.

   NOTE: The peering logical interfaces must belong to the same logical tunnel interface derived from the Tunnel Services PIC.

   ```
   [edit interfaces]
   lt-1/1/10 {
     unit 0 {
       encapsulation ethernet-ccc;
       peer-unit 1;
       family ccc;
     }
     unit 1 {
       encapsulation ethernet;
       peer-unit 0;
       family inet {
         address 70.70.70.1/24;
       }
     }
   }
   ```

3. On each router, commit the configuration.

   ```
   user@host> commit check
   configuration check succeeds
   user@host> commit
   ```

Verifying the Layer 2 Circuit to Layer 3 VPN Interconnection

To verify that the interconnection is working properly, perform these tasks:

Verifying That the Layer 2 Circuit Connection to Router PE3 is Up

Purpose To verify that the Layer 2 circuit connection from Router PE2 to Router PE3 is **Up**. To also document the incoming and outgoing LDP labels and the circuit ID used by this Layer 2 circuit connection.

Action Verify that the Layer 2 circuit connection is up, using the **show l2circuit connections** command.

```
user@PE2> show l2circuit connections

Legend for connection status (St)
EI -- encapsulation invalid      NP -- interface h/w not present
MM -- mtu mismatch               Dn -- down
EM -- encapsulation mismatch     VC-Dn -- Virtual circuit Down
CM -- control-word mismatch      Up -- operational
VM -- vlan id mismatch           CF -- Call admission control failure
OL -- no outgoing label          IB -- TDM incompatible bitrate
NC -- intf encaps not CCC/TCC    TM -- TDM misconfiguration
BK -- Backup Connection          ST -- Standby Connection
CB -- rcvd cell-bundle size bad  SP -- Static Pseudowire
LD -- local site signaled down   RS -- remote site standby
RD -- remote site signaled down  XX -- unknown

Legend for interface status
Up -- operational
Dn -- down
Neighbor: 3.3.3.3
    Interface                 Type  St     Time last up          # Up trans
    ge-1/0/2.0(vc 100)        rmt   Up     Jan  7 02:14:13 2010           1
      Remote PE: 3.3.3.3, Negotiated control-word: No
      Incoming label: 301488, Outgoing label: 315264
      Negotiated PW status TLV: No
      Local interface: ge-1/0/2.0, Status: Up, Encapsulation: ETHERNET
```

Meaning The output shows that the Layer 2 circuit connection from Router PE2 to Router PE3 is **Up** and the connection is using the **ge-1/0/2.0** interface. Note that the outgoing label is **315264** and the incoming label is **301488**, the virtual circuit (VC) identifier is **100** and the encapsulation is **ETHERNET**.

Verifying LDP Neighbors and Targeted LDP LSPs on Router PE2

Purpose To verify that Router PE2 has a targeted LDP LSP to Router PE3 and that Router PE2 and Router PE3 are LDP neighbors.

Action Verify that Router PE2 has a targeted LDP LSP to Router PE3 and that Router PE2 and Router PE3 are LDP neighbors, using the **show ldp neighbor** command.


```
user@PE2> show ldp neighbor
Address            Interface          Label space ID         Hold time
3.3.3.3            lo0.0              3.3.3.3:0                38
```

Meaning The output shows that Router PE2 has an LDP neighbor with the IPv4 address of **3.3.3.3**. Address 3.3.3.3 is the lo0.0 interface address of Router PE3. Notice that Router PE2 uses the local **lo0.0** interface for the LSP.

Verifying that the routers are LDP neighbors also verifies that the targeted LSP is established.

Verifying the Layer 2 Circuit Routes on Router PE2

Purpose To verify that Router PE2 has a route for the Layer 2 circuit and that the route uses the LDP MPLS label to Router PE3.

Action Verify that Router PE2 has a route for the Layer 2 circuit and that the route uses the LDP MPLS label to Router PE3, using the **show route table mpls.0** command.

```
user@PE2> show route table mpls.0
mpls.0: 13 destinations, 13 routes (13 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

0                  *[MPLS/0] 1w3d 05:24:11, metric 1
                      Receive
1                  *[MPLS/0] 1w3d 05:24:11, metric 1
                      Receive
2                  *[MPLS/0] 1w3d 05:24:11, metric 1
                      Receive
300560             *[LDP/9] 16:12:23, metric 1
                    > to 10.10.2.1 via xe-0/1/0.0, Pop
300560(S=0)        *[LDP/9] 16:12:23, metric 1
                    > to 10.10.2.1 via xe-0/1/0.0, Pop
301008             *[LDP/9] 16:12:23, metric 1
                    > to 10.10.4.2 via xe-0/3/0.0, Swap 299856
301488             *[L2CKT/7] 11:07:28
                    > via ge-1/0/2.0, Pop
301536             *[LDP/9] 16:12:23, metric 1
                    > to 10.10.4.2 via xe-0/3/0.0, Pop
301536(S=0)        *[LDP/9] 16:12:23, metric 1
                    > to 10.10.4.2 via xe-0/3/0.0, Pop
301712             *[LDP/9] 12:41:22, metric 1
                    > to 10.10.5.2 via xe-0/2/0.0, Swap 315184
301728             *[LDP/9] 12:41:22, metric 1
                    > to 10.10.5.2 via xe-0/2/0.0, Pop
301728(S=0)        *[LDP/9] 12:41:22, metric 1
                    > to 10.10.5.2 via xe-0/2/0.0, Pop
ge-1/0/2.0        *[L2CKT/7] 11:07:28, metric2 1
                    > to 10.10.5.2 via xe-0/2/0.0, Push 315264
```

Meaning The output shows that Router PE2 pushes the **315264** outgoing label on the **L2CKT** route going out interface **ge-1/0/2.0**. The output also shows that Router PE2 pops the **301488** incoming label on the **L2CKT** coming from interface **ge-1/0/2.0**

Verifying That the Layer 2 Circuit Connection to Router PE2 is Up

Purpose To verify that the Layer 2 circuit connection from Router PE3 to Router PE2 is **Up**, To also document the incoming and outgoing LDP labels and the circuit ID used by this Layer 2 circuit connection.

Action Verify that the Layer 2 circuit connection is up, using the **show l2circuit connections** command.

```
user@PE3> show l2circuit connections

Layer-2 Circuit Connections:
Legend for connection status (St)
EI -- encapsulation invalid      NP -- interface h/w not present
MM -- mtu mismatch               Dn -- down
EM -- encapsulation mismatch     VC-Dn -- Virtual circuit Down
CM -- control-word mismatch      Up -- operational
VM -- vlan id mismatch           CF -- Call admission control failure
OL -- no outgoing label          IB -- TDM incompatible bitrate
NC -- intf encaps not CCC/TCC    TM -- TDM misconfiguration
BK -- Backup Connection          ST -- Standby Connection
CB -- rcvd cell-bundle size bad  XX -- unknown

Legend for interface status
Up -- operational
Dn -- down
Neighbor: 2.2.2.2
    Interface                 Type  St     Time last up          # Up trans
    lt-1/1/10.0(vc 100)       rmt   Up     Jan  7 02:15:03 2010           1
      Remote PE: 2.2.2.2, Negotiated control-word: No
      Incoming label: 315264, Outgoing label: 301488
      Local interface: lt-1/1/10.0, Status: Up, Encapsulation: ETHERNET
```

Meaning The output shows that the Layer 2 circuit connection from Router PE3 to Router PE2 is **Up** and the connection is using the logical tunnel (**lt**) interface. Note that the incoming label is **315264** and the outgoing label is **301488**, the virtual circuit (VC) identifier is **100**, and that the encapsulation is **ETHERNET**.

Verifying LDP Neighbors and Targeted LDP LSPs on Router PE3

Purpose To verify that Router PE3 has a targeted LDP LSP to Router PE2 and that Router PE3 and Router PE2 are LDP neighbors.

Action Verify that Router PE2 has a targeted LDP LSP to Router PE3 and that Router PE2 and Router PE3 are LDP neighbors, using the **show ldp neighbor** command.

```
user@PE2> show ldp neighbor
Address            Interface          Label space ID         Hold time
2.2.2.2            lo0.0              2.2.2.2:0                42
4.4.4.4            lo0.0              4.4.4.4:0                33
```

Meaning The output shows that Router PE3 has an LDP neighbor with the IPv4 address of **2.2.2.2**. Address 2.2.2.2 is the lo0.0 interface address of Router PE2. The output also shows that the interface used on Router PE3 for the LSP is **lo0.0**. Verifying that the routers are LDP neighbors also verifies that the targeted LSP is established.

Verifying a BGP Peer Session with the Route Reflector on Router PE3

Purpose To verify that Router PE3 has a peer session established with the route reflector.

Action Verify that Router PE3 has a peer session established with the route reflector, using the **show bgp summary** command.

```
user@PE2> show bgp summary

Groups: 2 Peers: 2 Down peers: 0
Table          Tot Paths  Act Paths Suppressed    History Damp State    Pending
bgp.l3vpn.0            1          1          0          0          0          0
Peer                     AS      InPkt     OutPkt    OutQ   Flaps Last Up/Dwn
State|#Active/Received/Accepted/Damped...
7.7.7.7               65000       1597       1612       0       1    12:03:21 Establ
  bgp.l2vpn.0: 0/0/0/0
  bgp.l3vpn.0: 1/1/1/0
  L3VPN.inet.0: 1/1/1/0
```

Meaning The output shows that Router PE3 has a peer session with the router with the IPv4 address of **7.7.7.7**. Address 7.7.7.7 is the lo0.0 interface address of the route reflector. The output also shows that the peer session state is **Establ**, meaning that the session is established.

Verifying the Layer 3 VPN Routes on Router PE3

Purpose To verify that Router PE3 has Layer 3 VPN routes to Router CE2, Router CE3, and Router CE5.

Action Verify that Router PE3 has routes to Router CE2, Router CE3, and Router CE5 in the Layer 3 VPN route table, using the **show route table L3VPN.inet.0** command. In this example, **L3VPN** is the name configured for the routing instance.

```
user@PE3> show route table L3VPN.inet.0
L3VPN.inet.0: 5 destinations, 5 routes (5 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

70.70.70.0/24      *[Direct/0] 11:13:59
                    > via lt-1/1/10.1
70.70.70.1/32      *[Local/0] 11:13:59
                       Local via lt-1/1/10.1
80.80.80.0/24      *[BGP/170] 11:00:41, localpref 100, from 7.7.7.7
                       AS path: I
                    > to 10.10.6.2 via xe-2/1/0.0, Push 16
90.90.90.0/24      *[Direct/0] 11:54:41
                    > via ge-1/0/1.0
90.90.90.1/32      *[Local/0] 11:54:41
                       Local via ge-1/0/1.0
```

Meaning The output shows that Router PE3 has a route to the IPv4 subnetwork address of **70.70.70.0**. Address 70.70.70.2 is the interface address of Router CE2. The output shows that Router PE3 has a route to the IPv4 subnetwork address of **80.80.80.0**. Address 80.80.80.2 is the interface address of Router CE5. The output shows that Router PE3 has a route to the IPv4 subnetwork address of **90.90.90.0**. Address 90.90.90.2 is the interface address of Router CE3.

Verifying the Layer 2 Circuit Routes on Router PE3

Purpose To verify that Router PE3 has a route to Router PE2 in the Layer 2 circuit route table.

Action Verify that Router PE3 has a route to Router PE2 in the Layer 2 circuit route table, using the **show route table l2circuit.0** command.

```
user@PE3> show route table l2circuit.0
2.2.2.2:NoCtrlWord:5:100:Local/96 (1 entry, 1 announced)
        *L2CKT  Preference: 7
                Next hop type: Indirect
                Next-hop reference count: 1
                Next hop type: Router
                Next hop: 10.10.5.1 via xe-2/2/0.0, selected
                Protocol next hop: 2.2.2.2
                Indirect next hop: 8cae0a0 -
                State: <Active Int>
                Local AS: 65000
                Age: 11:16:50   Metric2: 1
                Task: l2 circuit
                Announcement bits (1): 0-LDP
                AS path: I
                VC Label 315264, MTU 1500
```

Meaning The output shows that Router PE3 has a route to the IPv4 address of **2.2.2.2**. Address 2.2.2.2 is the lo0.0 interface address of Router PE2. Note that the VC label is **315264**. This label is the same as the incoming MPLS label displayed using the **show l2circuit connections** command.

Verifying the MPLS Routes on Router PE3

Purpose To verify that Router PE3 has a route to Router PE2 in the MPLS route table.

Action Verify Router PE3 has a route to Router PE2 in the MPLS route table, using the **show route table mpls.0** command.

```
user@PE3> show route table mpls.0
mpls.0: 21 destinations, 21 routes (21 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

0                  *[MPLS/0] 1w3d 05:29:02, metric 1
                      Receive
1                  *[MPLS/0] 1w3d 05:29:02, metric 1
                      Receive
2                  *[MPLS/0] 1w3d 05:29:02, metric 1
                      Receive
16                 *[VPN/0] 12:22:45
                      to table L3VPN.inet.0, Pop
315184             *[LDP/9] 12:45:14, metric 1
                    > to 10.10.20.1 via xe-2/0/0.0, Pop
315184(S=0)        *[LDP/9] 12:45:14, metric 1
                    > to 10.10.20.1 via xe-2/0/0.0, Pop
315200             *[LDP/9] 00:03:53, metric 1
                    > to 10.10.20.1 via xe-2/0/0.0, Swap 625297
                      to 10.10.6.2 via xe-2/1/0.0, Swap 299856
315216             *[LDP/9] 12:45:14, metric 1
                    > to 10.10.6.2 via xe-2/1/0.0, Pop
```

```
315216(S=0)        *[LDP/9] 12:45:14, metric 1
                    > to 10.10.6.2 via xe-2/1/0.0, Pop
315232             *[LDP/9] 12:45:06, metric 1
                    > to 10.10.1.1 via xe-2/3/0.0, Pop
315232(S=0)        *[LDP/9] 12:45:06, metric 1
                    > to 10.10.1.1 via xe-2/3/0.0, Pop
315248             *[LDP/9] 12:45:14, metric 1
                    > to 10.10.5.1 via xe-2/2/0.0, Pop
315248(S=0)        *[LDP/9] 12:45:14, metric 1
                    > to 10.10.5.1 via xe-2/2/0.0, Pop
315264             *[L2CKT/7] 11:11:20
                    > via lt-1/1/10.0, Pop
315312             *[RSVP/7] 11:26:01, metric 1
                    > to 10.10.6.2 via xe-2/1/0.0, label-switched-path to-pe5
315312(S=0)        *[RSVP/7] 11:26:01, metric 1
                    > to 10.10.6.2 via xe-2/1/0.0, label-switched-path to-pe5
315328             *[RSVP/7] 11:26:01, metric 1
                    > to 10.10.20.1 via xe-2/0/0.0, label-switched-path to-RR
315360             *[RSVP/7] 11:26:01, metric 1
                    > to 10.10.20.1 via xe-2/0/0.0, label-switched-path to-RR
316208             *[RSVP/7] 00:03:32, metric 1
                    > to 10.10.6.2 via xe-2/1/0.0, label-switched-path
Bypass->10.10.9.1
316208(S=0)        *[RSVP/7] 00:03:32, metric 1
                    > to 10.10.6.2 via xe-2/1/0.0, label-switched-path
Bypass->10.10.9.1
lt-1/1/10.0        *[L2CKT/7] 11:11:20, metric2 1
                    > to 10.10.5.1 via xe-2/2/0.0, Push 301488
```

Meaning The output shows that Router PE3 has a route for the Layer 2 circuit and that the route uses the LDP MPLS label to Router PE2. Notice that the **301488** label is the same as the outgoing label displayed on Router PE2 using the **show l2circuit connections** command.

Verifying Traffic Flow Between Router CE2 and Router CE3

Purpose To verify that the CE routers can send and receive traffic across the interconnection.

Action Verify that Router CE2 can send traffic to and receive traffic from Router CE3 across the interconnection, using the **ping** command.

```
user@CE2>ping 90.90.90.2
PING 90.90.90.2 (90.90.90.2): 56 data bytes
64 bytes from 90.90.90.2: icmp_seq=0 ttl=63 time=0.708 ms
64 bytes from 90.90.90.2: icmp_seq=1 ttl=63 time=0.610 ms
```

Meaning The output shows that Router CE2 can send an ICMP request to and receive a response from Router CE3 across the interconnection.

Verifying Traffic Flow Between Router CE2 and Router CE5

Purpose To verify that the CE routers can send and receive traffic across the interconnection.

Action Verify that Router CE2 can send traffic to and receive traffic from Router CE5 across the interconnection, using the **ping** command.

```
user@CE2>ping 80.80.80.2
```

```
PING 80.80.80.2 (80.80.80.2): 56 data bytes
64 bytes from 80.80.80.2: icmp_seq=0 ttl=62 time=0.995 ms
64 bytes from 80.80.80.2: icmp_seq=1 ttl=62 time=1.005 ms
```

Meaning The output shows that Router CE2 can send an ICMP request to and receive a response from Router CE5 across the interconnection.

Related Documentation

- Layer 2 Circuit Overview
- Layer 3 VPN Overview on page 294
- Applications for Interconnecting a Layer 2 Circuit with a Layer 3 VPN on page 296

PART 3

Administration

CHAPTER 7

Layer 3 VPNs Reference

Supported Layer 3 VPN Standards

The Junos OS substantially supports the following RFCs, which define standards for Layer 3 virtual private networks (VPNs).

- RFC 2283, *Multiprotocol Extensions for BGP-4*
- RFC 2685, *Virtual Private Networks Identifier*
- RFC 2858, *Multiprotocol Extensions for BGP-4*
- RFC 4364, *BGP/MPLS IP Virtual Private Networks (VPNs)*
- RFC 4379, *Detecting Multi-Protocol* [sic] *Label Switched (MPLS) Data Plane Failures*

  The traceroute functionality is supported only on transit routers.
- RFC 4576, *Using a Link State Advertisement (LSA) Options Bit to Prevent Looping in BGP/MPLS IP Virtual Private Networks (VPNs)*
- RFC 4577, *OSPF as the Provider/Customer Edge Protocol for BGP/MPLS IP Virtual Private Networks (VPNs)*
- RFC 4659, *BGP-MPLS IP Virtual Private Network (VPN) Extension for IPv6 VPN*
- RFC 4684, *Constrained Route Distribution for Border Gateway Protocol/MultiProtocol* [sic] *Label Switching (BGP/MPLS) Internet Protocol (IP) Virtual Private Networks (VPNs)*

The following RFC does not define a standard, but provides information about technology related to Layer 3 VPNs. The IETF classifies it as a "Best Current Practice."

- RFC 1918, *Address Allocation for Private Internets*

Related Documentation

- Supported Carrier-of-Carriers and Interprovider VPN Standards
- Supported Layer 2 Circuit Standards
- Supported Layer 2 VPN Standard
- Supported Multicast VPN Standards
- Supported VPLS Standards

- Supported MPLS Standards
- Supported BGP Standards
- OSPF Features in the Junos OS
- Accessing Standards Documents on the Internet

CHAPTER 8

Summary of Layer 3 VPN Configuration Statements

as-path-compare

Syntax as-path-compare;

Hierarchy Level [edit logical-systems *logical-system-name* routing-instances *routing-instance-name* routing-options multipath],
[edit routing-instances *routing-instance-name* routing-options multipath]

Release Information Statement introduced in Junos OS Release 10.1.

Description Specify to have the algorithm that is used to determine the active path compare the AS numbers in the AS path. In a VPN scenario with multiple BGP paths, the algorithm selects as the active path the route whose AS numbers match. By default, the algorithm evaluates only the length and not the contents of the AS path.

Required Privilege Level routing—To view this statement in the configuration.
routing-control—To add this statement to the configuration.

Related Documentation
- Configuring the Algorithm That Determines the Active Route to Evaluate AS Numbers in AS Paths for VPN Routes on page 85

classifiers

Syntax

```
classifiers {
  exp (classifier-name | default);
}
```

Hierarchy Level [edit class-of-service routing-instances *routing-instance-name*]

Release Information Statement introduced before Junos OS Release 7.4.

Description For routing instances with VRF table labels enabled, apply a custom MPLS EXP classifier to the routing instance. You can apply the default MPLS EXP classifier or one that is previously defined.

Default If you do not include this statement, the default MPLS EXP classifier is applied to the routing instance.

Options ***classifier-name***—Name of the behavior aggregate MPLS EXP classifier.

Required Privilege Level interface—To view this statement in the configuration.
interface-control—To add this statement to the configuration.

Related Documentation

- Applying Custom MPLS EXP Classifiers to Routing Instances in Layer 3 VPNs on page 50
- *Junos OS Network Interfaces Configuration Guide*

domain-id

Syntax domain-id *domain-id*;

Hierarchy Level [edit logical-systems *logical-system-name* routing-instances *routing-instance-name* protocols (ospf | ospf3)],
[edit routing-instances *routing-instance-name* protocols (ospf | ospf3)]

Release Information Statement introduced before Junos OS Release 7.4.

Description Specify a domain ID for a route. The domain ID identifies the OSPFv2 domain from which the route originated.

Default If the router ID is not configured in the routing instance, the router ID is derived from an interface address belonging to the routing instance.

Options ***domain-id***—IP address.

Required Privilege Level routing—To view this statement in the configuration.
routing-control—To add this statement to the configuration.

Related Documentation

- Configuring an OSPF Domain ID on page 33

domain-vpn-tag

Syntax domain-vpn-tag *number*;

Hierarchy Level [edit logical-systems *logical-system-name* routing-instances *routing-instance-name* protocols (ospf | ospf3)],
[edit routing-instances *routing-instance-name* protocols (ospf | ospf3)]

Release Information Statement introduced before Junos OS Release 7.4.

Description Set a virtual private network (VPN) tag for OSPFv2 external routes generated by the provider edge (PE) router.

Options ***number***—VPN tag.

Required Privilege Level routing—To view this statement in the configuration.
routing-control—To add this statement to the configuration.

Related Documentation

- Configuring an OSPF Domain ID on page 33

dynamic-tunnels

Syntax

```
dynamic-tunnels tunnel-name {
  destination-networks prefix;
  source-address address;
  tunnel-type gre;
}
```

Hierarchy Level [edit logical-systems *logical-system-name* routing-options],
[edit routing-options]

Release Information Statement introduced before Junos OS Release 7.4.

Description Enable dynamic tunnel creation.

Options

destination-networks *prefix*—Specifies the IP version 4 (IPv4) prefix range for the destination network by including the **destination-networks** statement. Only tunnels within the specified IPv4 prefix range are allowed to be initiated.

source-address *address*—Specifies the source address for the generic routing encapsulation (GRE) tunnels. The source address specifies the address used as the source for the local tunnel endpoint. This could be any local address on the router (typically the router ID or the loopback address).

tunnel-name—Specifies the name of the dynamic tunnel.

tunnel-type gre—Specifies that a GRE tunnel is to be dynamically created.

Required Privilege Level routing—To view this statement in the configuration.
routing-control—To add this statement to the configuration.

Related Documentation

- Example: Configuring a Two-Tiered Virtualized Data Center for Large Enterprise Networks
- Configuring GRE Tunnels Dynamically on page 76
- *Junos OS Routing Protocols Configuration Guide*

independent-domain

Syntax independent-domain;

Hierarchy Level [edit logical-systems *logical-system-name* routing-instances *routing-instance-name* routing-options autonomous-system *autonomous-system*],
[edit routing-instances *routing-instance-name* routing-options autonomous-system *autonomous-system*],

Release Information Statement introduced before Junos OS Release 7.4.

Description Improve the transparency of Layer 3 VPN services for customer networks by preventing the IBGP routes that originate within an autonomous system (AS) in the customer network from being sent to a service provider's AS. Similarly, IBGP routes that originate within an AS in the service provider's network are prevented from being sent to a customer AS.

Required Privilege Level routing—To view this statement in the configuration.
routing-control—To add this statement to the configuration.

Related Documentation
- Configuring Layer 3 VPNs to Carry IBGP Traffic on page 42
- Configuring Independent AS Domains

inet6-vpn

Syntax

```
inet6-vpn (any | multicast | unicast) {
  aggregate-label;
  prefix-limit maximum;
  rib-group rib-group-name;
}
```

Hierarchy Level

[edit logical-systems *logical-system-name* protocols bgp family],
[edit logical-systems *logical-system-name* protocols bgp group *group-name* family],
[edit protocols bgp family],
[edit protocols bgp group *group-name* family]

Release Information

Statement introduced before Junos OS Release 7.4.

Description

Enable IP version 6 (IPv6) on the provider edge (PE) router for the Layer 3 VPN.

Options

any—Configure the family type to be both multicast and unicast.

multicast—Configure the family type to be multicast. This means that the BGP peers are being used only to carry the unicast routes that are being used by multicast for resolving the multicast routes.

prefix-limit *maximum*—Maximum prefix limit.

Range: 1 through 4,294,967,295

Default: 1

rib-group *rib-group-name*—The name of the routing table group.

unicast—Configure the family type to be unicast. This means that the BGP peers only carry the unicast routes that are being used for unicast forwarding purposes.

Required Privilege Level

routing—To view this statement in the configuration.
routing-control—To add this statement to the configuration.

Related Documentation

- Configuring Layer 3 VPNs to Carry IPv6 Traffic on page 38
- *Junos OS Routing Protocols Configuration Guide*

l3vpn-composite-nexthop

Syntax l3vpn-composite-nexthop;

Hierarchy Level [edit logical-systems *logical-system-name* routing-options],
[edit routing-options]

Release Information Statement introduced in Junos OS Release 9.5.
Statement introduced on T Series routers in Junos OS Release 10.4.

Description Accept larger numbers of Layer 3 VPN BGP updates with unique inner VPN labels. This feature is available on the M120 router, M320 router with Enhanced III FPC, MX Series router, and T Series routers. The neighboring PE routers are typically non–Juniper Networks routers configured to assign a unique inner label to each Layer 3 VPN BGP route.

NOTE: On MX Series routers, this statement is not supported when the router has both MS-DPCs and MPCs installed. You must remove one or the other type of card to successfully use this statement.

Default This statement is disabled by default.

Required Privilege Level routing—To view this statement in the configuration.
routing-control—To add this statement to the configuration.

Related Documentation

- Accepting BGP Updates with Unique Inner VPN Labels in Layer 3 VPNs on page 87

label

Syntax

```
label {
  allocation label-allocation-policy;
  substitution label-substitution-policy;
}
```

Hierarchy Level

[edit logical-systems *logical-system-name* routing-instances *routing-instance-name* routing-options]
[edit routing-instances *routing-instance-name* routing-options]

Release Information

Statement introduced in Junos OS Release 10.0.

Description

Specify label allocation and substitution policies on a per-route basis.

Options

allocation *label-allocation-policy*—Specify a policy to generate labels on a per-route basis.

substitution *label-substituion-policy*—Specify a policy to substitute labels on a per-route basis. The label substitution policy is used to determine whether or not a label should be substituted on an AS border router. The results of the policy operation are either accept (label substitution is performed) or reject (label substitution is not performed).

Default: accept

Required Privilege Level

routing—To view this statement in the configuration.
routing-control—To add this statement to the configuration.

Related Documentation

- Configuring a Label Allocation and Substitution Policy for VPNs on page 69

maximum-paths

Syntax maximum-paths *path-limit* <log-interval *interval* | log-only | threshold *percentage*>;

Hierarchy Level [edit logical-systems *logical-system-name* routing-instances *routing-instance-name* routing-options],
[edit logical-systems *logical-system-name* routing-options],
[edit routing-instances *routing-instance-name* routing-options],
[edit routing-options]

Release Information Statement introduced in Junos OS Release 8.0.

Description Specify a maximum limit on the number of routes in a routing instance. Using a path limit, you can curtail the number of paths received from a CE router in a VPN. Path limits apply only to dynamic routing protocols and are not applicable to static or interface routes.

NOTE: The **maximum-paths** statement is similar to the **maximum-prefixes** statement. The **maximum-prefixes** statement limits the number of unique destinations in a routing instance. For example, suppose a routing instance has the following routes:

```
OSPF 10.10.10.0/24
ISIS 10.10.10.0/24
```

There are two routes, but only one destination (prefix). The **maximum-paths** limit applies the total number of routes (two). The **maximum-prefixes** limit applies to the total number of unique prefixes (one).

Options ***path-limit***—Specify the maximum number of paths.

Range: 1 through 4,294,967,295 paths

log-interval—Minimum interval between log messages.

Range: 5 through 86,400 seconds

log-only—Generate warning messages only. No limit is placed on the number of paths stored in the routing tables.

threshold—Percentage of the path limit at which to begin sending warning log messages.

Required Privilege Level routing—To view this statement in the configuration.
routing-control—To add this statement to the configuration.

Related Documentation

- Limiting the Number of Paths and Prefixes Accepted from CE Routers in Layer 3 VPNs on page 38

maximum-prefixes

Syntax maximum-prefixes *prefix-limit* <log-interval *interval* | log-only | threshold *percentage*>;

Hierarchy Level [edit logical-systems *logical-system-name* routing-instances *routing-instance-name* routing-options],
[edit logical-systems *logical-system-name* routing-options],
[edit routing-instances *routing-instance-name* routing-options],
[edit routing-options]

Release Information Statement introduced in Junos OS Release 8.0.

Description Specify a maximum limit on the number of prefixes that can be installed into the routing tables. Using a prefix limit, you can curtail the number of prefixes received from a CE router in a VPN. Prefix limits apply only to dynamic routing protocols and are not applicable to static or interface routes.

NOTE: The **maximum-paths** statement is similar to the **maximum-prefixes** statement. The **maximum-prefixes** statement limits the number of unique destinations in a routing instance. For example, suppose a routing instance has the following routes:

```
OSPF 10.10.10.0/24
ISIS 10.10.10.0/24
```

There are two routes, but only one destination (prefix). The **maximum-paths** limit applies the total number of routes (two). The **maximum-prefixes** limit applies to the total number of unique prefixes (one).

Options ***prefix-limit***—Specify the maximum number of prefixes.

Range: 1 through 4,294,967,295 prefixes

log-interval—Minimum interval between log messages.

Range: 5 through 86,400 seconds

log-only—Generate warning messages only. No limit is placed on the number of prefixes stored in the routing tables.

threshold—Percentage of the prefix limit at which to begin sending warning log messages.

Required Privilege Level routing—To view this statement in the configuration.
routing-control—To add this statement to the configuration.

Related Documentation

- Limiting the Number of Paths and Prefixes Accepted from CE Routers in Layer 3 VPNs on page 38

metric

Syntax metric *number*;

Hierarchy Level [edit logical-systems *logical-system-name* routing-instances *routing-instance-name* protocols ospf area *area-id* sham-link-remote],
[edit routing-instances *routing-instance-name* protocols ospf area *area-id* sham-link-remote]

Release Information Statement introduced before Junos OS Release 7.4.

Description Specify the cost of using the OSPF sham link.

Default *number*—1

Options *number*—1 through 65,535

Required Privilege Level routing—To view this statement in the configuration.
routing-control—To add this statement to the configuration.

Related Documentation

- Configuring OSPF Sham Links on page 32

multihop

Syntax multihop *ttl-value*;

Hierarchy Level
[edit logical-systems *logical-system-name* routing-instances *routing-instance-name* protocols bgp],
[edit logical-systems *logical-system-name* routing-instances *routing-instance-name* protocols bgp group *group-name*],
[edit logical-systems *logical-system-name* routing-instances *routing-instance-name* protocols bgp group *group-name* neighbor *address*],
[edit routing-instances *routing-instance-name* protocols bgp],
[edit routing-instances *routing-instance-name* protocols bgp group *group-name*],
[edit routing-instances *routing-instance-name* protocols bgp group *group-name* neighbor *address*]

Release Information Statement introduced before Junos OS Release 7.4.

Description Configure an EBGP multihop session between the PE and customer edge (CE) routers of a Layer 3 VPN. This allows you to have one or more routers between the PE and CE routers.

You cannot configure the **accept-remote-nexthop** statement at the same time.

Options ***ttl-value***—Specify the time-to-live (TTL) value for the multihop session to prevent routing loops.

Required Privilege Level
routing—To view this statement in the configuration.
routing-control—To add this statement to the configuration.

Related Documentation
- Configuring EBGP Multihop Sessions Between PE and CE Routers in Layer 3 VPNs on page 42

multipath

Syntax

```
multipath {
  vpn-unequal-cost equal-external-internal;
  as-path-compare;
}
```

Hierarchy Level

[edit logical-systems *logical-system-name* routing-instances *routing-instance-name* routing-options],
[edit logical-systems *logical-system-name* routing-instances *routing-instance-name* routing-options rib *routing-table-name*],
[edit routing-instances *routing-instance-name* routing-options],
[edit routing-instances *routing-instance-name* routing-options rib *routing-table-name*]

Release Information

Statement introduced before Junos OS Release 7.4.
equal-external-internal option added for Junos OS Release 8.4.
as-path-compare option introduced in Junos OS Release 10.1

Description

Enable protocol-independent load balancing for Layer 3 VPNs. This allows the forwarding next hops for both the active route and alternative paths to be used for load balancing. For IPv6, you must configure the **multipath** statement at both the **[edit routing-instances *routing-instance-name* routing-options]** hiearchy level and at the **[edit routing-instances *routing-instance-name* routing-options rib *routing-table-name*]** hierarchy level.

The options are explained separately.

Required Privilege Level

routing—To view this statement in the configuration.
routing-control—To add this statement to the configuration.

Related Documentation

- Configuring Protocol-Independent Load Balancing in Layer 3 VPNs on page 83

no-vrf-propagate-ttl

Syntax no-vrf-propagate-ttl;

Hierarchy Level [edit logical-systems *logical-system-name* routing-instances *routing-instance-name*], [edit protocols routing-instances *routing-instance-name*]

Release Information Statement introduced in Junos OS Release 10.4.

Description Disable normal time-to-live (TTL) decrementing in a VRF routing instance. You configure this statement once per routing instance, and it affects only RSVP-signaled or LDP-signaled LSPs in the routing instance. When this router acts as an ingress router for an LSP, it pushes an MPLS header with a TTL value of 255, regardless of the IP packet TTL. When the router acts as the penultimate router, it pops the MPLS header without writing the MPLS TTL into the IP packet.

Default Normal TTL decrementing enabled; the TTL field value is decremented by 1 as the packet passes through each label-switched router in the LSP.

Required Privilege Level routing—To view this statement in the configuration.
routing-control—To add this statement to the configuration.

Related Documentation

- Example: Disabling Normal TTL Decrementing in a VRF Routing Instance on page 183
- Disabling Normal TTL Decrementing in the *Junos OS MPLS Applications Configuration Guide*

routing-instances

Syntax

```
routing-instances routing-instance-name {
  classifiers {
    exp (classifier-name | default);
  }
}
```

Hierarchy Level [edit class-of-service]

Release Information Statement introduced before Junos OS Release 7.4.

Description For routing instances with the **vrf-table-label** statement in their configuration, apply a custom MPLS EXP classifier to the routing instance. You can apply the default MPLS EXP classifier or one that is previously defined.

Options ***routing-instance-name***—Name of the routing instance.

The other statements are explained separately.

Required Privilege Level routing—To view this statement in the configuration.
routing-control—To add this statement to the configuration.

Related Documentation

- Applying Custom MPLS EXP Classifiers to Routing Instances in Layer 3 VPNs on page 50
- *Junos OS Network Interfaces Configuration Guide*

sham-link

Syntax

```
sham-link {
  local address;
}
```

Hierarchy Level [edit logical-systems *logical-system-name* routing-instances *routing-instance-name* protocols ospf],
[edit routing-instances *routing-instance-name* protocols ospf]

Release Information Statement introduced before Junos OS Release 7.4.

Description Configure a sham link for the Layer 3 VPN routing instance.

Options **local *address***—The address for the local endpoint of the sham link.

Required Privilege Level routing—To view this statement in the configuration.
routing-control—To add this statement to the configuration.

Related Documentation

- Configuring OSPF Sham Links on page 32
- *Junos OS Routing Protocols Configuration Guide*

sham-link-remote

Syntax sham-link-remote *address* <metric *number*>;

Hierarchy Level [edit logical-systems *logical-system-name* routing-instances *routing-instance-name* protocols ospf area *area-id*],
[edit routing-instances *routing-instance-name* protocols ospf area *area-id*]

Release Information Statement introduced before Junos OS Release 7.4.

Description Configure the address for the remote end point of the sham link.

Options ***address***—Address for the remote end point of the sham link.

The other statement is explained separately.

Required Privilege Level routing—To view this statement in the configuration.
routing-control—To add this statement to the configuration.

Related Documentation
- Configuring OSPF Sham Links for Layer 3 VPNs on page 31
- *Junos OS Routing Protocols Configuration Guide*

vpn-group-address

Syntax vpn-group-address *address*;

Hierarchy Level [edit logical-systems *logical-system-name* routing-instances *routing-instance-name* protocols pim],
[edit routing-instances *routing-instance-name* protocols pim]

Release Information Statement introduced before Junos OS Release 7.4.

Description Configure the group address for the Layer 3 VPN in the service provider's network.

Options ***address***—Address for the Layer 3 VPN in the service provider's network.

Required Privilege Level routing—To view this statement in the configuration.
routing-control—To add this statement to the configuration.

Related Documentation
- Configuring Multicast Layer 3 VPNs on page 71
- *Junos OS Multicast Protocols Configuration Guide*

vpn-unequal-cost

Syntax

```
vpn-unequal-cost {
  equal-external-internal;
}
```

Hierarchy Level

[edit logical-systems *logical-system-name* routing-instances *routing-instance-name* routing-options multipath],
[edit logical-systems *logical-system-name* routing-instances *routing-instance-name* routing-options rib *routing-table-name* multipath],
[edit routing-instances *routing-instance-name* routing-options multipath],
[edit routing-instances *routing-instance-name* routing-options rib *routing-table-name* multipath]

Release Information

Statement introduced before Junos OS Release 7.4. The **equal-external-internal** option was added for Junos OS Release 8.4.

Description

Apply protocol-independent load balancing to VPN routes that are equal until their interior gateway protocol (IGP) metrics with regard to route selection. If you do not configure the **vpn-unequal-cost** statement, protocol-independent load balancing is applied to VPN routes that are equal until their router identifiers with regard to route selection.

Options

equal-external-internal—Specifies that both external and internal BGP paths can be selected for multipath.

Required Privilege Level

routing—To view this statement in the configuration.
routing-control—To add this statement to the configuration.

Related Documentation

- Configuring Load Balancing for Layer 3 VPNs on page 83
- Load Balancing and IP Header Filtering for Layer 3 VPNs on page 51

vrf-propagate-ttl

Syntax vrf-propagate-ttl;

Hierarchy Level [edit logical-systems *logical-system-name* routing-instances *routing-instance-name*], [edit routing-instances *routing-instance-name*]

Release Information Statement introduced in Junos OS Release 10.4.

Description Enable normal time-to-live (TTL) decrementing in a VRF routing instance. You configure this statement once per routing instance, and it affects only RSVP-signaled or LDP-signaled LSPs in the routing instance. When this router acts as an ingress router for an LSP, it pushes an MPLS copies the TTL value from the IP packet header. When the router acts as the penultimate router, it pops the MPLS header and writes the MPLS TTL into the IP packet.

Default Normal TTL decrementing enabled; the TTL field value is decremented by 1 as the packet passes through each label-switched router in the LSP. This statement explicitly configures the default behavior for the VRF routing instance and is useful for overriding the **no-propagate-ttl** configured globally on the router at the **[edit protocols mpls]** hierarchy level.

Required Privilege Level routing—To view this statement in the configuration.
routing-control—To add this statement to the configuration.

Related Documentation

- Example: Disabling Normal TTL Decrementing in a VRF Routing Instance on page 183
- Disabling Normal TTL Decrementing in the *Junos OS MPLS Applications Configuration Guide*

vrf-table-label

Syntax vrf-table-label;

Hierarchy Level [edit logical-systems *logical-system-name* routing-instances *routing-instance-name*], [edit routing-instances *routing-instance-name*]

Release Information Statement introduced before Junos OS Release 7.4.
Statement introduced in Junos OS Release 11.1 for EX Series switches.

Description Map the inner label of a packet to a specific VPN routing and forwarding (VRF) table. This allows the examination of the encapsulated IP header.

Required Privilege Level routing—To view this statement in the configuration.
routing-control—To add this statement to the configuration.

Related Documentation

- Filtering Packets in Layer 3 VPNs Based on IP Headers on page 43
- Configuring EXP-Based Traffic Classification for VPLS
- Load Balancing and IP Header Filtering for Layer 3 VPNs on page 51

PART 4

Troubleshooting

CHAPTER 9

Troubleshooting Layer 3 VPNs

Diagnosing Common Problems

Problem To troubleshoot problems in the Layer 3 VPN configuration, start at one end of the VPN (the local customer edge [CE] router) and follow the routes to the other end of the VPN (the remote CE router).

Solution The following troubleshooting steps should help you diagnose common problems:

1. If you configured a routing protocol between the local provider edge (PE) and CE routers, check that the peering and adjacency are fully operational. When you do this, be sure to specify the name of the routing instance. For example, to check OSPF adjacencies, enter the **show ospf neighbor instance** ***routing-instance-name*** command on the PE router.

   If the peering and adjacency are not fully operational, check the routing protocol configuration on the CE router and check the routing protocol configuration for the associated VPN routing instance on the PE router.

2. Check that the local CE and PE routers can ping each other.

   To check that the local CE router can ping the VPN interface on the local PE router, use a **ping** command in the following format, specifying the IP address or name of the PE router:

   user@host> **ping** (*ip-address* | *host-name*)

   To check that the local PE router can ping the CE router, use a **ping** command in the following format, specifying the IP address or name of the CE router, the name of the

interface used for the VPN, and the source IP address (the local address) in outgoing echo request packets:

user@host> **ping** *ip-address* **interface** *interface* **local** *echo-address*

Often, the peering or adjacency between the local CE and local PE routers must come up before a **ping** command is successful. To check that a link is operational in a lab setting, remove the interface from the VPN routing and forwarding (VRF) by deleting the **interface** statement from the **[edit routing-instance** *routing-instance-name***]** hierarchy level and recommitting the configuration. Doing this removes the interface from the VPN. Then try the **ping** command again. If the command is successful, configure the interface back into the VPN and check the routing protocol configuration on the local CE and PE routers again.

3. On the local PE router, check that the routes from the local CE router are in the VRF table (***routing-instance-name*.inet.0**):

   user@host> **show route table** *routing-instance-name*.**inet.0 <detail>**

   The following example shows the routing table entries. Here, the loopback address of the CE router is **10.255.14.155/32** and the routing protocol between the PE and CE routers is BGP. The entry looks like any ordinary BGP announcement.

   ```
   10.255.14.155/32 (1 entry, 1 announced)
           *BGP    Preference: 170/-101
                   Nexthop: 192.168.197.141 via fe-1/0/0.0, selected
                   State: <Active Ext>
                   Peer AS:     1
                   Age: 45:46
                   Task: BGP_1.192.168.197.141+179
                   Announcement bits (2): 0-BGP.0.0.0.0+179 1-KRT
                   AS path: 1 I
                   Localpref: 100
                   Router ID: 10.255.14.155
   ```

   If the routes from the local CE router are not present in the VRF routing table, check that the CE router is advertising routes to the PE router. If static routing is used between the CE and PE routers, make sure the proper static routes are configured.

4. On a remote PE router, check that the routes from the local CE router are present in the **bgp.l3vpn.0** routing table:

   ```
   user@host> show route table bgp.l3vpn.0 extensive

   10.255.14.175:3:10.255.14.155/32 (1 entry, 0 announced)
           *BGP    Preference: 170/-101
                   Route Distinguisher: 10.255.14.175:3
                   Source: 10.255.14.175
                   Nexthop: 192.168.192.1 via fe-1/1/2.0, selected
                   label-switched-path vpn07-vpn05
                   Push 100004, Push 100005(top)
                   State: <Active Int Ext>
                   Local AS:    69 Peer AS:    69
                   Age: 15:27      Metric2: 338
                   Task: BGP_69.10.255.14.175+179
                   AS path: 1 I
                   Communities: target:69:100
                   BGP next hop: 10.255.14.175
                   Localpref: 100
   ```

```
Router ID: 10.255.14.175
Secondary tables: VPN-A.inet.0
```

The output of the **show route table bgp.l3vpn.0 extensive** command contains the following information specific to the VPN:

- In the prefix name (the first line of the output), the route distinguisher is added to the route prefix of the local CE router. Because the route distinguisher is unique within the Internet, the concatenation of the route distinguisher and IP prefix provides unique VPN-IP version 4 (IPv4) routing entries.
- The **Route Distinguisher** field lists the route distinguisher separately from the VPN-IPv4 address.
- The **label-switched-path** field shows the name of the label-switched path (LSP) used to carry the VPN traffic.
- The **Push** field shows both labels being carried in the VPN-IPv4 packet. The first label is the inner label, which is the VPN label that was assigned by the PE router. The second label is the outer label, which is an RSVP label.
- The **Communities** field lists the target community.
- The **Secondary tables** field lists other routing tables on this router into which this route has been installed.

If routes from the local CE router are not present in the **bgp.l3vpn.0** routing table on the remote PE router, do the following:

- Check the VRF import filter on the remote PE router, which is configured in the **vrf-import** statement. (On the local PE router, you check the VRF export filter, which is configured with the **vrf-export** statement.)
- Check that there is an operational LSP or an LDP path between the PE routers. To do this, check that the IBGP next-hop addresses are in the **inet.3** table.
- Check that the IBGP session between the PE routers is established and configured properly.
- Check for "hidden" routes, which usually means that routes were not labeled properly. To do this, use the **show route table bgp.l3vpn.0 hidden** command.
- Check that the inner label matches the inner VPN label that is assigned by the local PE router. To do this, use the **show route table mpls** command.

The following example shows the output of this command on the remote PE router. Here, the inner label is **100004**.

```
...
Push 100004, Push 10005 (top)
```

The following example shows the output of this command on the local PE router, which shows that the inner label of **100004** matches the inner label on the remote PE router:

```
...
100004             *[VPN/7] 06:56:25, metric 1
 > to 192.168.197.141 via fe-1/0/0.0, Pop
```

5. On the remote PE router, check that the routes from the local CE router are present in the VRF table (***routing-instance-name*.inet.0**):

   ```
   user@host> show route table routing-instance-name.inet.0 detail

   10.255.14.155/32 (1 entry, 1 announced)
           *BGP    Preference: 170/-101
                   Route Distinguisher: 10.255.14.175:3
                   Source: 10.255.14.175
                   Nexthop: 192.168.192.1 via fe-1/1/2.0, selected
                   label-switched-path vpn07-vpn05
                   Push 100004, Push 100005(top)
                   State: <Secondary Active Int Ext>
                   Local AS:    69 Peer AS:    69
                   Age: 1:16:22    Metric2: 338
                   Task: BGP_69.10.255.14.175+179
                   Announcement bits (2): 1-KRT 2-VPN-A-RIP
                   AS path: 1 I
                   Communities: target:69:100
                   BGP next hop: 10.255.14.175
                   Localpref: 100
                   Router ID: 10.255.14.175
                   Primary Routing Table bgp.l3vpn.0
   ```

   In this routing table, the route distinguisher is no longer prepended to the prefix. The last line, **Primary Routing Table**, lists the table from which this route was learned.

   If the routes are not present in this routing table, but were present in the **bgp.l3vpn.0** routing table on the local CE router, the routes might have not passed the VRF import policy on the remote PE router.

   If a VPN-IPv4 route matches no **vrf-import** policy, the route does not show up in the **bgp.l3vpn** table at all and hence is not present in the VRF table. If this occurs, it might indicate that on the PE router, you have configured another **vrf-import** statement on another VPN (with a common target), and the routes show up in the **bgp.l3vpn.0** table, but are imported into the wrong VPN.

6. On the remote CE router, check that the routes from the local CE router are present in the routing table (**inet.0**):

   user@host> **show route**

   If the routes are not present, check the routing protocol configuration between the remote PE and CE routers, and make sure that peers and adjacencies (or static routes) between the PE and CE routers are correct.

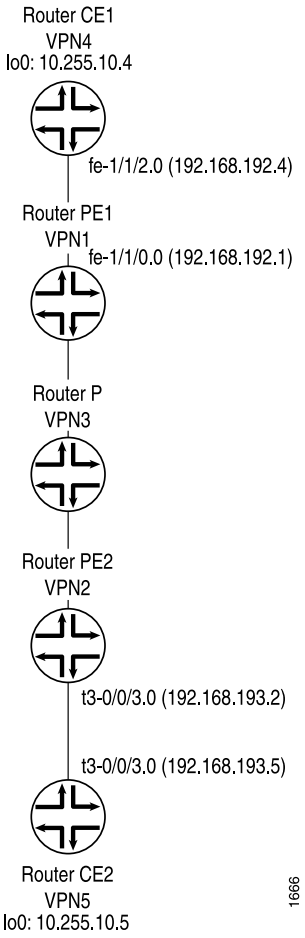
7. If you determine that routes originated from the local CE router are correct, check the routes originated from the remote CE router by repeating this procedure.

Troubleshooting Layer 3 VPNs Using ping and traceroute

This section provides examples of how to use the **ping** command to check the accessibility of various routers in a VPN topology, and how to use the **traceroute** command to check

the path that packets travel between the VPN routers. The topology shown in Figure 48 on page 371 illustrates these commands.

Figure 48: Layer 3 VPN Topology for ping and traceroute Examples



Pinging the CE Router from Another CE Router

The following sections describe how to use the **ping** and **traceroute** commands to troubleshoot Layer 3 VPN topologies. You can ping one CE router from the other by specifying the other CE router's loopback address as the IP address in the **ping** command. This **ping** command succeeds if the loopback addresses have been announced by the CE routers to their directly connected PE routers. The success of these **ping** commands also means that Router CE1 can ping any network devices beyond Router CE2, and vice versa. Figure 48 on page 371 shows the topology referenced in the following examples:

- Pinging Router CE2 from Router CE1 on page 372
- Using traceroute from Loopback to Loopback on page 372
- Pinging Router CE1 from Router CE2 on page 372
- Using traceroute from Router CE2 to Router CE1 on page 372

Pinging Router CE2 from Router CE1

Ping Router CE2 (VPN5) from Router CE1 (VPN4):

```
user@vpn4> ping 10.255.10.5 local 10.255.10.4 count 3
PING 10.255.10.5 (10.255.10.5): 56 data bytes
64 bytes from 10.255.10.5: icmp_seq=0 ttl=253 time=1.086 ms
64 bytes from 10.255.10.5: icmp_seq=1 ttl=253 time=0.998 ms
64 bytes from 10.255.10.5: icmp_seq=2 ttl=253 time=1.140 ms
--- 10.255.10.5 ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max/stddev = 0.998/1.075/1.140/0.059 ms
```

Using traceroute from Loopback to Loopback

To determine the path from Router CE1's loopback interface to Router CE2's loopback interface, use the **traceroute** command:

```
user@vpn4> traceroute 10.255.10.5 source 10.255.10.4
traceroute to 10.255.10.5 (10.255.10.5) from 10.255.10.4, 30 hops max, 40 byte
packets
 1  vpn1-fe-110.isp-core.net (192.168.192.1)  0.680 ms  0.491 ms  0.456 ms
 2  vpn2-t3-001.isp-core.net (192.168.192.110)  0.857 ms  0.766 ms  0.754 ms
     MPLS Label=100005 CoS=0 TTL=1 S=1
 3  vpn5.isp-core.net (10.255.10.5)  0.825 ms  0.886 ms  0.732 ms
```

When you use the **traceroute** command to examine the path used by a Layer 3 VPN, the provider (P) routers in the service provider's network are not displayed. As shown above, the jump from Router VPN1 to Router VPN2 is displayed as a single hop. The P router (VPN3) shown in Figure 48 on page 371 is not displayed.

Pinging Router CE1 from Router CE2

Ping Router CE1 (VPN4) from Router CE2 (VPN5):

```
user@vpn5> ping 10.255.10.4 local 10.255.10.5 count 3
PING 10.255.10.4 (10.255.10.4): 56 data bytes
64 bytes from 10.255.10.4: icmp_seq=0 ttl=253 time=1.042 ms
64 bytes from 10.255.10.4: icmp_seq=1 ttl=253 time=0.998 ms
64 bytes from 10.255.10.4: icmp_seq=2 ttl=253 time=0.954 ms
--- 10.255.10.4 ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max/stddev = 0.954/0.998/1.042/0.036 ms
```

Using traceroute from Router CE2 to Router CE1

To determine the path from Router CE2 to Router CE1, use the **traceroute** command:

```
user@vpn5> traceroute 10.255.10.4 source 10.255.10.5
traceroute to 10.255.10.4 (10.255.10.4) from 10.255.10.5, 30 hops max, 40 byte
packets
 1  vpn-08-t3-003.isp-core.net (192.168.193.2)  0.686 ms  0.519 ms  0.548 ms
 2  vpn1-so-100.isp-core.net (192.168.192.100)  0.918 ms  0.869 ms  0.859 ms
     MPLS Label=100021 CoS=0 TTL=1 S=1
 3  vpn4.isp-core.net (10.255.10.4)  0.878 ms  0.760 ms  0.739 ms
```

Pinging the Remote PE and CE Routers from the Local CE Router

From the local CE router, you can ping the VPN interfaces on the remote PE and CE routers, which are point-to-point interfaces. Figure 48 on page 371 shows the topology referenced in the following examples:

Pinging Router CE2 from Router CE1

Ping Router CE2 (VPN5) from Router CE1 (VPN4):

```
user@vpn4> ping 192.168.193.5 local 10.255.10.4 count 3
PING 192.168.193.5 (192.168.193.5): 56 data bytes
64 bytes from 192.168.193.5: icmp_seq=0 ttl=253 time=1.040 ms
64 bytes from 192.168.193.5: icmp_seq=1 ttl=253 time=0.891 ms
64 bytes from 192.168.193.5: icmp_seq=2 ttl=253 time=0.944 ms
--- 192.168.193.5 ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max/stddev = 0.891/0.958/1.040/0.062 ms
```

Using traceroute from Router CE1 to Router CE2

To determine the path from Router CE1's loopback interface to Router CE2's directly connected interface, use the **traceroute** command:

```
user@vpn4> traceroute 192.168.193.5 source 10.255.10.4
traceroute to 192.168.193.5 (192.168.193.5) from 10.255.10.4, 30 hops max, 40 byte
 packets
 1  vpn1-fe-110.isp-core.net (192.168.192.1)  0.669 ms  0.508 ms  0.457 ms
 2  vpn2-t3-001.isp-core.net (192.168.192.110)  0.851 ms  0.769 ms  0.750 ms
     MPLS Label=100000 CoS=0 TTL=1 S=1
 3  vpn5-t3-003.isp-core.net (192.168.193.5)  0.829 ms  0.838 ms  0.731 ms
```

Pinging Router PE2 from Router CE1

Ping Router PE2 (VPN2) from Router CE1 (VPN4). In this case, packets that originate at Router CE1 go to Router PE2, then to Router CE2, and back to Router PE2 before Router PE2 can respond to Internet Control Message Protocol (ICMP) requests. You can verify this by using the **traceroute** command.

```
user@vpn4> ping 192.168.193.2 local 10.255.10.4 count 3
PING 192.168.193.2 (192.168.193.2): 56 data bytes
64 bytes from 192.168.193.2: icmp_seq=0 ttl=254 time=1.080 ms
64 bytes from 192.168.193.2: icmp_seq=1 ttl=254 time=0.967 ms
64 bytes from 192.168.193.2: icmp_seq=2 ttl=254 time=0.983 ms
--- 192.168.193.2 ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max/stddev = 0.967/1.010/1.080/0.050 ms
```

Using traceroute from Router CE1 to Router PE2

To determine the path from Router CE1 to Router PE2, use the **traceroute** command:

```
user@vpn4> traceroute 192.168.193.2 source 10.255.10.4
traceroute to 192.168.193.2 (192.168.193.2) from 10.255.10.4, 30 hops max, 40 byte
 packets
 1  vpn1-fe-110.isp-core.net (192.168.192.1)  0.690 ms  0.490 ms  0.458 ms
 2  vpn2-t3-003.isp-core.net (192.168.193.2)  0.846 ms  0.768 ms  0.749 ms
     MPLS Label=100000 CoS=0 TTL=1 S=1
 3  vpn5-t3-003.isp-core.net (192.168.193.5)  0.643 ms  0.703 ms  0.600 ms
 4  vpn-08-t3-003.isp-core.net (192.168.193.2)  0.810 ms  0.739 ms  0.729 ms
```

Pinging a CE Router from a Multiaccess Interface

You cannot ping one CE router from the other if the VPN interface is a multiaccess interface, such as the **fe-1/1/2.0** interface on Router CE1. To ping Router CE1 from Router CE2, you must either include the **vrf-table-label** statement at the **[edit routing-instances *routing-instance-name*]** hierarchy level on Router PE1 or configure a static route on Router PE1 to the VPN interface of Router CE1. If you include the **vrf-table-label** statement to ping a router, you cannot configure a static route.

If you configure a static route on Router PE1 to the VPN interface of Router CE1, its next hop must point to Router CE1 (at the **[edit routing-instance *routing-instance-name*]** hierarchy level), and this route must be announced from Router PE1 to Router PE2 as shown in the following configuration:

```
[edit]
routing-instances {
    direct-multipoint {
        instance-type vrf;
        interface fe-1/1/0.0;
        route-distinguisher 69:1;
        vrf-import direct-import;
        vrf-export direct-export;
        routing-options {
            static {
                route 192.168.192.4/32 next-hop 192.168.192.4;
            }
        }
        protocols {
            bgp {
                group to-vpn4 {
                    peer-as 1;
                    neighbor 192.168.192.4;
                }
            }
        }
    }
    policy-options {
        policy-statement direct-export {
            term a {
                from protocol bgp;
                then {
                    community add direct-comm;
```

```
            accept;
          }
        }
        term b {
          from {
            protocol static;
            route-filter 192.168.192.4/32 exact;
          }
          then {
            community add direct-comm;
            accept;
          }
        }
        term d {
          then reject;
        }
      }
    }
  }
```

Now you can ping Router CE1 from Router CE2:

```
user@vpn5> ping 192.168.192.4 local 10.255.10.5 count 3
PING 192.168.192.4 (192.168.192.4): 56 data bytes
64 bytes from 192.168.192.4: icmp_seq=0 ttl=253 time=1.092 ms
64 bytes from 192.168.192.4: icmp_seq=1 ttl=253 time=1.019 ms
64 bytes from 192.168.192.4: icmp_seq=2 ttl=253 time=1.031 ms
--- 192.168.192.4 ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max/stddev = 1.019/1.047/1.092/0.032 ms
```

To determine the path between these two interfaces, use the **traceroute** command:

```
user@vpn5> traceroute 192.168.192.4 source 10.255.10.5
traceroute to 192.168.192.4 (192.168.192.4) from 10.255.10.5, 30 hops max, 40 byte
 packets
 1  vpn-08-t3003.isp-core.net (192.168.193.2)  0.678 ms  0.549 ms  0.494 ms
 2  vpn1-so-100.isp-core.net (192.168.192.100)  0.873 ms  0.847 ms  0.844 ms
     MPLS Label=100021 CoS=0 TTL=1 S=1
 3  vpn4-fe-112.isp-core.net (192.168.192.4)  0.825 ms  0.743 ms  0.764 ms
```

Pinging the Directly Connected PE Routers from the CE Routers

From the loopback interfaces on the CE routers, you can ping the VPN interface on the directly connected PE router. Figure 48 on page 371 shows the topology referenced in the following examples:

- Pinging Router PE1 from the Loopback Interface on Router CE1 on page 375
- Using traceroute from the Loopback Interface on Router CE1 to PE1 on page 376
- Pinging Router PE2 from the Loopback Interface on Router CE2 on page 376
- Using traceroute from the Loopback Interface on Router CE2 to PE2 on page 376

Pinging Router PE1 from the Loopback Interface on Router CE1

From the loopback interface on Router CE1 (VPN4), ping the VPN interface, **fe-1/1/0.0**, on Router PE1:

```
user@vpn4> ping 192.168.192.1 local 10.255.10.4 count 3
PING 192.168.192.1 (192.168.192.1): 56 data bytes
64 bytes from 192.168.192.1: icmp_seq=0 ttl=255 time=0.885 ms
64 bytes from 192.168.192.1: icmp_seq=1 ttl=255 time=0.757 ms
64 bytes from 192.168.192.1: icmp_seq=2 ttl=255 time=0.734 ms
--- 192.168.192.1 ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max/stddev = 0.734/0.792/0.885/0.066 ms
```

Using traceroute from the Loopback Interface on Router CE1 to PE1

To determine the path from the loopback interface on Router CE1 to the VPN interfaces on Router PE1, use the **traceroute** command:

```
user@vpn4> traceroute 192.168.192.1 source 10.255.10.4
traceroute to 192.168.192.1 (192.168.192.1) from 10.255.10.4, 30 hops max, 40 byte
 packets
 1  vpn1-fe-110.isp-core.net (192.168.192.1)  0.828 ms  0.657 ms  1.972 ms
```

Pinging Router PE2 from the Loopback Interface on Router CE2

From the loopback interface on Router CE2 (VPN5), ping the VPN interface, **t3-0/0/3.0**, on Router PE2:

```
user@vpn5> ping 192.168.193.2 local 10.255.10.5 count 3
PING 192.168.193.2 (192.168.193.2): 56 data bytes
64 bytes from 192.168.193.2: icmp_seq=0 ttl=255 time=0.998 ms
64 bytes from 192.168.193.2: icmp_seq=1 ttl=255 time=0.834 ms
64 bytes from 192.168.193.2: icmp_seq=2 ttl=255 time=0.819 ms
--- 192.168.193.2 ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max/stddev = 0.819/0.884/0.998/0.081 ms
```

Using traceroute from the Loopback Interface on Router CE2 to PE2

To determine the path from the loopback interface on Router CE2 to the VPN interfaces on Router PE2, use the **traceroute** command:

```
user@vpn5> traceroute 192.168.193.2 source 10.255.10.5
traceroute to 192.168.193.2 (192.168.193.2) from 10.255.10.5, 30 hops max, 40 byte
 packets
 1  vpn-08-t3003.isp-core.net (192.168.193.2)  0.852 ms  0.670 ms  0.656 ms
```

Pinging the Directly Connected CE Routers from the PE Routers

From the VPN and loopback interfaces on the PE routers, you can ping the VPN interface on the directly connected CE router. Figure 48 on page 371 shows the topology referenced in the following examples:

- Pinging the VPN Interface on Router CE1 from Router PE1 on page 377
- Pinging the Loopback Interface on Router CE1 from Router PE1 on page 377
- Using traceroute from Router PE1 to Router CE1 on page 377
- Pinging the VPN Interface on Router CE2 from Router PE2 on page 377

Pinging the VPN Interface on Router CE1 from Router PE1

From the VPN interface on the PE router, you can ping the VPN or loopback interface on the directly connected CE router.

From the VPN interface on Router PE1 (VPN1), ping the VPN interface, **fe-1/1/0.0**, on Router CE1:

```
user@vpn1> ping 192.168.192.4 interface fe-1/1/0.0 local 192.168.192.1 count 3
PING 192.168.192.4 (192.168.192.4): 56 data bytes
64 bytes from 192.168.192.4: icmp_seq=0 ttl=255 time=0.866 ms
64 bytes from 192.168.192.4: icmp_seq=1 ttl=255 time=0.728 ms
64 bytes from 192.168.192.4: icmp_seq=2 ttl=255 time=0.753 ms
--- 192.168.192.4 ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max/stddev = 0.728/0.782/0.866/0.060 ms
```

Pinging the Loopback Interface on Router CE1 from Router PE1

From the VPN interface on Router PE1 (VPN1), ping the loopback interface, **10.255.10.4**, on Router CE1:

```
user@vpn1> ping 10.255.10.4 interface fe-1/1/0.0 local 192.168.192.1 count 3
PING 10.255.10.4 (10.255.10.4): 56 data bytes
64 bytes from 10.255.10.4: icmp_seq=0 ttl=255 time=0.838 ms
64 bytes from 10.255.10.4: icmp_seq=1 ttl=255 time=0.760 ms
64 bytes from 10.255.10.4: icmp_seq=2 ttl=255 time=0.771 ms
--- 10.255.10.4 ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max/stddev = 0.760/0.790/0.838/0.034 ms
```

Using traceroute from Router PE1 to Router CE1

To determine the path from the VPN interface on Router PE1 to the VPN and loopback interfaces on Router CE1, respectively, use the following **traceroute** commands:

```
user@vpn1> traceroute 10.255.10.4 interface fe-1/1/0.0 source 192.168.192.1
traceroute to 10.255.10.4 (10.255.10.4) from 192.168.192.1, 30 hops max, 40 byte
 packets
 1  vpn4.isp-core.net (10.255.10.4)  0.842 ms  0.659 ms  0.621 ms
user@vpn1>  traceroute 192.168.192.4 interface fe-1/1/0.0 source 192.168.192.1

traceroute to 192.168.192.4 (192.168.192.4) from 192.168.192.1, 30 hops max,
40 byte packets
 1  vpn4-fe-112.isp-core.net (192.168.192.4)  0.810 ms  0.662 ms  0.640 ms
```

Pinging the VPN Interface on Router CE2 from Router PE2

From the VPN interface on Router PE2 (VPN2), ping the VPN interface, **t3-0/0/3.0**, on Router CE2:

```
user@vpn2> ping 192.168.193.5 interface t3-0/0/3.0 local 192.168.193.2 count 3
PING 192.168.193.5 (192.168.193.5): 56 data bytes
64 bytes from 192.168.193.5: icmp_seq=0 ttl=255 time=0.852 ms
64 bytes from 192.168.193.5: icmp_seq=1 ttl=255 time=0.909 ms
```

```
64 bytes from 192.168.193.5: icmp_seq=2 ttl=255 time=0.793 ms
--- 192.168.193.5 ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max/stddev = 0.793/0.851/0.909/0.047 ms
```

Pinging the Loopback Interface on Router CE2 from Router PE2

From the VPN interface on Router PE2 (VPN2), ping the loopback interface, **10.255.10.5**, on Router CE2:

```
user@vpn2> ping 10.255.10.5 interface t3-0/0/3.0 local 192.168.193.2 count 3
PING 10.255.10.5 (10.255.10.5): 56 data bytes
64 bytes from 10.255.10.5: icmp_seq=0 ttl=255 time=0.914 ms
64 bytes from 10.255.10.5: icmp_seq=1 ttl=255 time=0.888 ms
64 bytes from 10.255.10.5: icmp_seq=2 ttl=255 time=1.066 ms
--- 10.255.10.5 ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max/stddev = 0.888/0.956/1.066/0.079 ms
```

Using traceroute from Router PE2 to Router CE2

To determine the path from the VPN interface on Router PE2 to the VPN and loopback interfaces on Router CE2, respectively, use the following **traceroute** commands:

```
user@vpn2> traceroute 10.255.10.5 interface t3-0/0/3.0 source 192.168.193.2
traceroute to 10.255.10.5 (10.255.10.5) from 192.168.193.2, 30 hops max, 40 byte
 packets
 1  vpn5.isp-core.net (10.255.10.5)  1.009 ms  0.677 ms  0.633 ms
user@vpn2>  traceroute 192.168.193.5 interface t3-0/0/3.0 source 192.168.193.2
traceroute to 192.168.193.5 (192.168.193.5) from 192.168.193.2, 30 hops max,
40 byte packets
 1  vpn5-t3-003.isp-core.net (192.168.193.5)  0.974 ms  0.665 ms  0.619 ms
```

Pinging the Remote CE Router from the Local PE Router

The following procedure is effective for Layer 3 VPNs only. To ping a remote CE router from a local PE router in a Layer 3 VPN, you need to configure the following interfaces:

1. Configure a logical unit for the loopback interface.

   To configure an additional logical unit on the loopback interface of the PE router, configure the **unit** statement at the **[edit interfaces lo0]** hierarchy level:

   ```
   [edit interfaces]
   lo0 {
     unit number {
       family inet {
         address address;
       }
     }
   }
   ```

2. Configure the loopback interface for the Layer 3 VPN routing instance on the local PE router. You can associate one logical loopback interface with each Layer 3 VPN routing instance, enabling you to ping a specific routing instance on a router.

Specify the loopback interface you configured in Step 1 using the **interface** statement at the **[edit routing-instances *routing-instance-name*]** hierarchy level:

```
[edit routing-instances routing-instance-name]
interface interface-name;
```

The ***interface-name*** is the logical unit on the loopback interface (for example, **lo0.1**).

3. From the VPN interface on PE router, you can now ping the logical unit on the loopback interface on the remote CE router:

```
user@host> ping interface interface host
```

Use ***interface*** to specify the new logical unit on the loopback interface (for example, **lo0.1**). For more information about how to use the **ping interface** command, see the *Junos Interfaces Command Reference*.

Limitation on Pinging a Remote CE Router from a PE Router

If you attempt to ping a remote CE router from a PE router, ICMP echo requests are sent from the PE router, with the PE router's VPN interface as the source. Other PE routers have a route back to that address with a VPN label. When the echo replies return, they include a label. The PE router pops the VPN label and sends the packet from the VPN interface to the local CE router. The local CE router sends it back to the PE router, its actual destination.

When a Juniper Networks routing platform receives a labeled packet, the label is popped (depending on the label operation specified), and the packet is forwarded to an interface, even if the packet is destined for that particular PE router. Labeled packets are not analyzed further for the IP information under the label.

If there is a problem with the connection to the local CE router, packets are sent out but do not return to the PE router, and the ping fails. If the connection between your PE router and local CE router is down, sending a ping to the remote CE router fails even though the connection to the remote CE router might be functional.

Troubleshooting Inconsistently Advertised Routes from Gigabit Ethernet Interfaces

For direct routes on a LAN in a Layer 3 VPN, the Junos OS attempts to locate a CE router that can be designated as the next hop. If this cannot be done, advertised routes from Gigabit Ethernet interfaces are dropped.

In such instances:

- Use the **static** statement at the **[edit routing-options]** or **[edit logical-systems *logical-system-name* routing-options]** hierarchy levels in the VRF routing instance to a CE router on the LAN subnet, configuring the CE router as the next hop. All traffic to directly destinations on this LAN will go to the CE router. You can add two static routes to two CE routers on the LAN for redundancy.
- Configure the **vrf-table-label** statement at the **[edit routing-instances *routing-instance-name*]** hierarchy levels to map the inner label of a packet to a specific VRF routing table. This allows the examination of the encapsulated IP header to force IP lookups on the VRF routing instance for all traffic.

NOTE: The **vrf-table-label** statement is not available for every core-facing interface; for example, channelized interfaces are not supported. See "Support on Ethernet, SONET/SDH, and T1/T3/E3 Interfaces for IP-Based Filtering" on page 46 for information about support for the **vrf-table-label** statement over Ethernet and SONET/SDH interfaces.

PART 5

Index

Index

S

T

V