JUNIPER NETWORKS

Junos® OS

VPN Overview and Common Configuration

Release
14.1

Published: 2014-05-09

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

Junos® OS VPN Overview and Common Configuration
14.1

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

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.

Supported Platforms

For the features described in this document, the following platforms are supported:

- MX Series
- T Series
- M Series
- PTX Series

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 *CLI User Guide*.

Documentation Conventions

Table 1 on page xiii 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.
	Tip	Indicates helpful information.
	Best practice	Alerts you to a recommended use or implementation.

Table 2 on page xiv 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 or emphasizes important new terms. • Identifies guide names. • Identifies RFC and Internet draft titles.	• A policy *term* is a named structure that defines match conditions and actions. • *Junos OS CLI User Guide* • RFC 1997, *BGP Communities Attribute*
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; 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)	Encloses 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)	Encloses a variable for which you can substitute one or more values.	**community name members [** ***community-ids*** **]**

Indention and braces ({ })	Identifies 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.	

GUI Conventions

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

Convention	Description	Examples
Bold text like this	Represents 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 menu 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: http://kb.juniper.net/InfoCenter/
- 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 VPNs

Types of VPNs

A virtual private network (VPN) consists of two topological areas: the provider's network and the customer's network. The customer's network is commonly located at multiple physical sites and is also private (non-Internet). A customer site would typically consist of a group of routers or other networking equipment located at a single physical location. The provider's network, which runs across the public Internet infrastructure, consists of routers that provide VPN services to a customer's network as well as routers that provide other services. The provider's network connects the various customer sites in what appears to the customer and the provider to be a private network.

To ensure that VPNs remain private and isolated from other VPNs and from the public Internet, the provider's network maintains policies that keep routing information from different VPNs separate. A provider can service multiple VPNs as long as its policies keep routes from different VPNs separate. Similarly, a customer site can belong to multiple VPNs as long as it keeps routes from the different VPNs separate.

The Junos® Operating System (Junos OS) provides several types of VPNs; you can choose the best solution for your network environment. Each of the following VPNs has different capabilities and requires different types of configuration:

Layer 2 VPNs

Implementing a Layer 2 VPN on a router is similar to implementing a VPN using a Layer 2 technology such as ATM or Frame Relay. However, for a Layer 2 VPN on a router, traffic is forwarded to the router in Layer 2 format. It is carried by MPLS over the service provider's network and then converted back to Layer 2 format at the receiving site. You can configure different Layer 2 formats at the sending and receiving sites. The security and privacy of an MPLS Layer 2 VPN are equal to those of an ATM or Frame Relay VPN.

On a Layer 2 VPN, routing occurs on the customer's routers, typically on the CE router. The CE router connected to a service provider on a Layer 2 VPN must select the appropriate circuit on which to send traffic. The PE router receiving the traffic sends it across the service provider's network to the PE router connected to the receiving site. The PE routers do not need to store or process the customer's routes; they only need to be configured to send data to the appropriate tunnel.

For a Layer 2 VPN, customers need to configure their own routers to carry all Layer 3 traffic. The service provider needs to know only how much traffic the Layer 2 VPN needs to carry. The service provider's routers carry traffic between the customer's sites using Layer 2 VPN interfaces. The VPN topology is determined by policies configured on the PE routers.

Layer 3 VPNs

In a Layer 3 VPN, the routing occurs on the service provider's routers. Therefore, Layer 3 VPNs require more configuration on the part of the service provider, because the service provider's PE routers must store and process the customer's routes.

In the Junos OS, Layer 3 VPNs are based on RFC 4364, *BGP/MPLS IP Virtual Private Networks (VPNs)*. This RFC defines a mechanism by which service providers can use their IP backbones to provide Layer 3 VPN services to their customers. The sites that make up a Layer 3 VPN are connected over a provider's existing public Internet backbone.

VPNs based on RFC 4364 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 private addresses used by other network users. BGP/MPLS VPNs solve this problem by prefixing a VPN identifier 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.

VPLS

Virtual private LAN service (VPLS) allows you to connect geographically dispersed customer sites as if they were connected to the same LAN. In many ways, it works like a Layer 2 VPN. VPLS and Layer 2 VPNs use the same network topology and function

similarly. A packet originating within a customer's network is sent first to a CE device. It is then sent to a PE router within the service provider's network. The packet traverses the service provider's network over an MPLS LSP. It arrives at the egress PE router, which then forwards the traffic to the CE device at the destination customer site.

The key difference in VPLS is that packets can traverse the service provider's network in a point-to-multipoint fashion, meaning that a packet originating from a CE device can be broadcast to PE routers in the VPLS. In contrast, a Layer 2 VPN forwards packets in a point-to-point fashion only. The destination of a packet received from a CE device by a PE router must be known for the Layer 2 VPN to function properly.

VPLS is designed to carry Ethernet traffic across an MPLS-enabled service provider network. In certain ways, VPLS mimics the behavior of an Ethernet network. When a PE router configured with a VPLS routing instance receives a packet from a CE device, it first checks the appropriate routing table for the destination of the VPLS packet. If the router has the destination, it forwards it to the appropriate PE router. If it does not have the destination, it broadcasts the packet to all the other PE routers that are members of the same VPLS routing instance. The PE routers forward the packet to their CE devices. The CE device that is the intended recipient of the packet forwards it to its final destination. The other CE devices discard it.

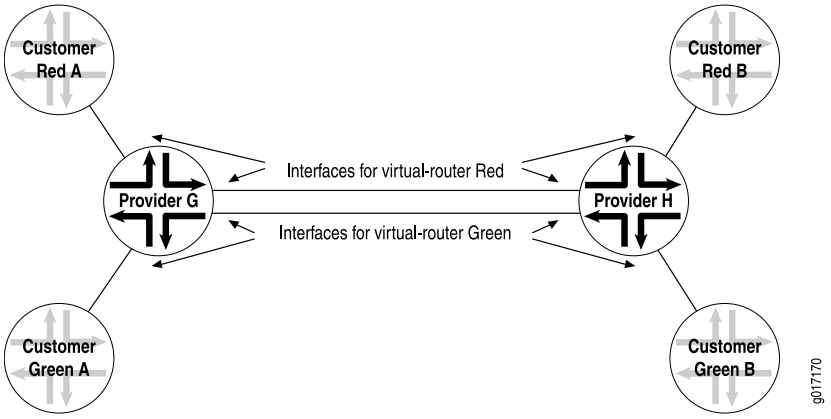
Virtual-Router Routing Instances

A virtual-router routing instance, like a VPN routing and forwarding (VRF) routing instance, maintains separate routing and forwarding tables for each instance. However, many configuration steps required for VRF routing instances are not required for virtual-router routing instances. Specifically, you do not need to configure a route distinguisher, a routing table policy (the **vrf-export**, **vrf-import**, and **route-distinguisher** statements), or MPLS between the P routers.

However, you need to configure separate logical interfaces between each of the service provider routers participating in a virtual-router routing instance. You also need to configure separate logical interfaces between the service provider routers and the customer routers participating in each routing instance. Each virtual-router instance requires its own unique set of logical interfaces to all participating routers.

Figure 1 on page 6 shows how this works. The service provider routers G and H are configured for virtual-router routing instances Red and Green. Each service provider router is directly connected to two local customer routers, one in each routing instance. The service provider routers are also connected to each other over the service provider network. These routers need four logical interfaces: a logical interface to each of the locally connected customer routers and a logical interface to carry traffic between the two service provider routers for each virtual-router instance.

Figure 1: Logical Interface per Router in a Virtual-Router Routing Instance



Layer 3 VPNs do not have this configuration requirement. If you configure several Layer 3 VPN routing instances on a PE router, all the instances can use the same logical interface to reach another PE router. This is possible because Layer 3 VPNs use MPLS (VPN) labels that differentiate traffic going to and from various routing instances. Without MPLS and VPN labels, as in a virtual-router routing instance, you need separate logical interfaces to separate traffic from different instances.

One method of providing this logical interface between the service provider routers is by configuring tunnels between them. You can configure IP Security (IPsec), generic routing encapsulation (GRE), or IP-IP tunnels between the service provider routers, terminating the tunnels at the virtual-router instance.

VPNs and Class of Service

You can configure Junos class-of-service (CoS) features to provide multiple classes of service for VPNs. The CoS features are supported on Layer2 VPNs, Layer 3 VPNs, and VPLS. On the router, you can configure multiple forwarding classes for transmitting packets, define which packets are placed into each output queue, schedule the transmission service level for each queue, and manage congestion using a random early detection (RED) algorithm.

VPNs use the standard CoS configuration. For information about how to configure CoS, see the *Junos OS Class of Service Library for Routing Devices*.

VPNs and Logical Systems

You can partition a single physical router into multiple logical systems that perform independent routing tasks. Because logical systems perform a subset of the tasks once handled by the physical router, logical systems offer an effective way to maximize the use of a single routing platform.

Logical systems perform a subset of the actions of a physical router and have their own unique routing tables, interfaces, policies, and routing instances. A set of logical systems within a single router can handle the functions previously performed by several small routers.

You can configure Layer 2 VPNs, Layer 3 VPNs, VPLS, and Layer 2 circuits within a logical system. For more information about logical systems, see the *Logical Systems Feature Guide for Routing Devices*.

NOTE: Beginning with Junos OS Release 9.3, the logical router feature has been renamed logical system.

All configuration statements, operational commands, show command outputs, error messages, log messages, and SNMP MIB objects that contain the string logical-router or logical-routers have been changed to logical-system and logical-systems, respectively.

VPN Graceful Restart

VPN graceful restart allows a router whose VPN control plane is undergoing a restart to continue to forward traffic while recovering its state from neighboring routers. Without graceful restart, a control plane restart disrupts any VPN services provided by the router.

For VPN graceful restart to function properly, the following items need to be configured on the PE router:

- BGP graceful restart must be active on the PE-to-PE sessions carrying any service-signaling data in the session's network layer reachability information (NLRI).
- OSPF, IS-IS, LDP, and RSVP graceful restart must be active, because routes added by these protocols are used to resolve VPN NLRIs.
- For other protocols (static, Routing Information Protocol [RIP], and so on), graceful restart functionality must also be active when these protocols are run between the PE and CE routers. Layer 2 VPNs do not rely on this because protocols are not configured between the PE and CE routers.

In VPN graceful restart, a restarting router completes the following procedures:

- Waits for all the BGP NLRI information from other PE routers before it starts advertising routes to its CE routers.
- Waits for all protocols in all routing instances to converge (or finish graceful restart) before sending CE router information to the other PE routers.
- Waits for all routing instance information (whether it is local configuration or advertisements from a remote peer router) to be processed before sending it to the other PE routers.
- Preserves all forwarding state information in the MPLS routing tables until new labels and transit routes are allocated and then advertises them to other PE routers (and CE routers in carrier-of-carriers VPNs).

Graceful restart is supported on Layer 2 VPNs, Layer 3 VPNs, and virtual-router routing instances.

Reducing Network Resource Use with Static Route Target Filtering

The BGP VPN route target extended community (RFC 4360, *BGP Extended Communities Attribute*) is used to determine VPN membership. Static route target filtering helps to prevent resources from being consumed in portions of the network where the VPN routes are not needed due to the lack of member PE routers (RFC 4684, *Constrained Route Distribution for Border Gateway Protocol/MultiProtocol Label Switching (BGP/MPLS) Internet Protocol (IP) Virtual Private Networks (VPNs)*). Routers can originate routes into the RT-Constrain protocol to indicate their interest in receiving VPN routes containing route targets that match the RT-Constrain NLRI.

Normally, for the RT-Constrain feature to function properly, it must be broadly deployed throughout a network. If this is not the case, the feature is less useful, because the RT-Constrain BGP speaker facing a non-RT-Constrain speaker must advertise a default RT-Constrain route to the other RT-Constrain speakers on behalf of the peer that does not support the feature. This effectively removes the resource saving benefits of the feature in portions of the network where it is not supported since a default RT-Constrain route causes the PE router and all intervening PE routers to need to receive all VPN routes.

The static RT-Constrain feature enables you to partially deploy the RT-Constrain feature in a network. The feature is enabled at a boundary in the network where RT-Constrain is configured. However, some BGP VPN peers do not support RT-Constrain, typically PE routers. The route targets of those PE routers must be statically configured on the router. These route targets are disseminated using the RT-Constrain protocol.

The proxy RT-Constrain feature permits BGP VPN peers that do not support the protocol to have their route-targets discovered and disseminated automatically. However, this feature can only support symmetric route-targets. For example, the import and export route-targets for a VRF routing instance are identical. However, for a hub-and-spoke VPN, the import and export route-targets are not identical. In this scenario, the import and export route-target may be statically configured to be disseminated in the RT-Constrain protocol.

Related Documentation

- Configuring Static Route Target Filtering for VPNs on page 40
- Configuring BGP Route Target Filtering for VPNs on page 38

Redundant Pseudowires for Layer 2 Circuits and VPLS

A redundant pseudowire can act as a backup connection between PE routers and CE devices, maintaining Layer 2 circuit and VPLS services after certain types of failures. This feature can help improve the reliability of certain types of networks (metro for example) where a single point of failure could interrupt service for multiple customers. Redundant pseudowires cannot reduce traffic loss to zero. However, they provide a way to gracefully recover from pseudowire failures in such a way that service can be restarted within a known time limit.

NOTE: VPLS is not supported on ACX Series routers.

When you configure redundant pseudowires to remote PE routers, you configure one to act as the primary pseudowire over which customer traffic is being transmitted and you configure another pseudowire to act as a backup in the event the primary fails. You configure the two pseudowires statically. A separate label is allocated for the primary and backup neighbors.

For information about how to configure redundant pseudowires, see "Configuring Redundant Pseudowires for Layer 2 Circuits and VPLS" on page 43.

The following sections provide an overview of redundant pseudowires for Layer 2 circuits and VPLS:

Types of Redundant Pseudowire Configurations

You can configure redundant pseudowires for Layer 2 circuits and VPLS in either of the following manners:

NOTE: VPLS is not supported on ACX Series routers.

- You can configure a single active pseudowire. The PE router configured as the primary neighbor is given preference and this connection is the one used for customer traffic. For the LDP signalling, labels are exchanged for both incoming and outgoing traffic with the primary neighbor. The LDP label advertisement is accepted from the backup neighbor, but no label advertisement is forwarded to it, leaving the pseudowire in an incomplete state. The pseudowire to the backup neighbor is completed only when the primary neighbor fails. The decision to switch between the two pseudowires is made by the device configured with the redundant pseudowires. The primary remote PE router is unaware of the redundant configuration, ensuring that traffic is always switched using just the active pseudowire.
- Alternatively, you can configure two active pseudowires, one to each of the PE routers. Using this approach, control plane signalling is completed and active pseudowires are

established with both the primary and backup neighbors. However, the data plane forwarding is done only over a one of the pseudowires (designated as the active pseudowire by the local device). The other pseudowire is on standby. The active pseudowire is preferably established with the primary neighbor and can switch to the backup pseudowire if the primary fails.

The decision to switch between the active and standby pseudowires is controlled by the local device. The remote PE routers are unaware of the redundant connection, and so both remote PE routers send traffic to the local device. The local device only accepts traffic from the active pseudowire and drops the traffic from the standby. In addition, the local device only sends traffic to the active pseudowire. If the active pseudowire fails, traffic is immediately switched to the standby pseudowire.

The two configurations available for pseudowire redundancy have the following limitations:

- For the single active pseudowire configuration, it takes more time (compared to the two active pseudowire configuration) to switchover to the backup pseudowire when a failure is detected. This approach requires additional control plane signalling to complete the pseudowire with the backup neighbor and traffic can be lost during the switchover from primary to backup.
- If you configure two active pseudowires, bandwidth is lost on the link carrying the backup pseudowire between the remote PE router and the local device. Traffic is always duplicated over both the active and standby pseudowires. The single active pseudowire configuration does not waste bandwidth in this fashion.
- You cannot enable GRES (graceful Routing Engine switchover) for redundant pseudowires.
- You cannot enable NSR (nonstop active routing) for redundant pseudowires.

NOTE: GRES and NSR are not supported on ACX Series routers.

Pseudowire Failure Detection

The following events are used to detect a failure (control and data plane) of the pseudowire configured between a local device and a remote PE router and initiates the switch to a redundant pseudowire:

- Manual switchover (user initiated)
- Remote PE router withdraws the label advertisement
- LSP to the remote PE router goes down
- LDP session with the remote PE router goes down
- Local configuration changes
- Periodic pseudowire OAM procedure fails (Layer 2 circuit-based MPLS ping to the PE router fails)

When you configure a redundant pseudowire between a CE device and a PE router, a periodic (once a minute) ping packet is forwarded through the active pseudowire to verify data plane connectivity. If the ping fails, traffic is automatically switched to the redundant pseudowire.

When a failure is detected, traffic is switched from the failed active pseudowire to the redundant pseudowire. The redundant pseudowire is then designated as the active pseudowire. The switch is nonreversible, meaning that once the redundant pseudowire assumes the role of the active pseudowire at the time of a failover, it remains as the active pseudowire even though the previously active pseudowire comes up again.

For example, a primary pseudowire has failed and traffic has been successfully switched to the redundant pseudowire. After a period of time, the cause of the failure of the primary pseudowire has been resolved and it is now possible to reestablish the original connection. However, traffic is not switched back to the original pseudowire unless a failure is detected on the currently active pseudowire.

Related Documentation

- *Example: Configuring H-VPLS Without VLANs*

BFD Support for VCCV for Layer 2 VPNs, Layer 2 Circuits, and VPLS

Bidirectional Forwarding Detection (BFD) support for virtual circuit connectvity verification (VCCV) on MX Series devices enables you to configure a control channel for a pseudowire, in addition to the corresponding operations, administration, and management functions to be used over that control channel.

BFD provides a low resource mechanism for the continuous monitoring of the pseudowire data path and for detecting data plane failures. This feature provides support for asynchronous mode BFD for VCCV as described in RFC 5885, *Bidirectional Forwarding Detection (BFD) for the Pseudowire Virtual Circuit Connectivity Verification (VCCV)*. Alternatively, you can use a ping operation to detect pseudowire failures. However, the processing resources required for a ping operation are greater than what is needed for BFD. In addition, BFD is capable of detecting data plane failure faster than a VCCV ping. BFD for pseudowires is supported for Layer 2 circuits (LDP-based), Layer 2 VPNs (BGP-based), and VPLS (LDP-based or BGP-based).

Starting with Release 12.1, Junos OS introduces a distributed model for the BFD for VCCV. Unlike in previous releases where the BFD for VCCV followed a Routing Engine-based implementation, in Release 12.1 and later, the BFD for VCCV follows a distributed implementation over PIC concentrators, such as DPC, FPC, and MPC.

NOTE: **For the distributed BFD for VCCV to work, you must configure MPLS family (family mpls) on the loopback interface.**

In Junos OS Release 12.1 and later, the periodic packet management process (ppmd) on the PIC concentrators handles the periodic packet management (send and receive) for BFD for VCCV. This enables Junos OS to create more BFD for VCCV sessions, and to reduce the time taken for error detection. Similarly, the distributed implementation

improves the performance of Routing Engines because the Routing Engine resources used for BFD for VCCV implementation become available for Routing Engine-related applications when the BFD for VCCV-related processing moves to the PIC concentrators. The distributed BFD for VCCV implementation also enables the BFD for VCCV sessions to remain across graceful restarts.

Related Documentation

- Configuring BFD for VCCV for Layer 2 VPNs, Layer 2 Circuits, and VPLS on page 46

Chained Composite Next Hops for Transit Devices

The Juniper Networks PTX Series Packet Transport Routers, MX Series 3D Universal Edge Routers with MIC and MPC interfaces, and T4000 Core Routers are principally designed to handle large volumes of transit traffic in the core of large networks. Chained composite next hops help to facilitate this capability by allowing the router to process much larger volumes of routes. A chained composite next hop allows the router to direct sets of routes sharing the same destination to a common forwarding next hop, rather than having each route also include the destination. In the event that a network destination is changed, rather than having to update all of the routes sharing that destination with the new information, just the shared forwarding next hop is updated with the new information. The chained composite next hops continue to point to this forwarding next hop which now contains the new destination.

When the next hops for MPLS LSPs are created on the routers, the tag information corresponding to the inner-most MPLS label is extracted into a chained composite next hop. The chained composite next hop is stored in the ingress PFE. The chained composite next hop points to a next hop called the forwarding next hop that resides on the egress PFE. The forwarding next hop contains all of the other information (all of the labels except for the inner-most labels; and the IFA/IP information corresponding to the actual next hop node). Many chained composite next hops can share the same forwarding next hop. Additionally, separating the label from the forwarding next hop and storing it on the ingress PFE (within the chained composite next hop) helps to conserve egress PFE memory by reducing the number of rewrite strings stored on the egress PFE.

The support of chained composite next hops for directly connected Provider Edge (PE) routers varies from one platform to another.

- On platforms containing only MPCs, such as PTX Series Packet Transport Routers, the MX80 router, and the MX2020 router, chained composite next hops are enabled by default for the following MPLS and VPN protocols and applications:
  - Labeled BGP
  - Layer 2 VPNs
  - Layer 3 VPNs
  - LDP
  - MPLS
  - Point-to-Multipoint LSPs

  - RSVP
  - Static LSPs
- On MX Series 3D Universal Edge Routers containing both DPC and MPC FPCs, chained composite next hops are disabled by default.

  To enable chained composite next hops on the MX240, MX480, and MX960, the chassis must be configured to use the **enhanced-ip** option in network services mode.
- On T4000 Core Routers containing MPC and FPCs, chained composite next hops are disabled by default.

  To enable chained composite next hops on a T4000 router, the chassis must be configured to use the **enhanced-mode** option in network services mode.

For more information about configuring chassis network services, see the *Junos OS Administration Library for Routing Devices*.

Related Documentation

- *Accepting Route Updates with Unique Inner VPN Labels in Layer 3 VPNs*
- Example: Configuring Chained Composite Next Hops for Direct PE-PE Connections on page 97

CHAPTER 2

Introduction to Configuring VPNs

Rewriting Markers and VPNs

A marker reads the current forwarding class and loss priority information associated with a packet and finds the chosen code point from a table. It then writes the code point information into the packet header. Entries in a marker configuration represent the mapping of the current forwarding class into a new forwarding class, to be written into the header.

You define markers in the rewrite rules section of the class-of-service (CoS) configuration hierarchy and reference them in the logical interface configuration. You can configure different rewrite rules to handle VPN traffic and non-VPN traffic. The rewrite rule can be applied to MPLS and IPv4 packet headers simultaneously, making it possible to initialize MPLS experimental (EXP) and IP precedence bits at LSP ingress.

For a detailed example of how to configure rewrite rules for MPLS and IPv4 packets and for more information about how to configure statements at the **[edit class-of-service]** hierarchy level, see the *Junos OS Class of Service Library for Routing Devices*.

Transmitting Nonstandard BPDUs

Circuit cross-connect (CCC) protocol, Layer 2 circuit, and Layer 2 VPN configurations can transmit nonstandard bridge protocol data units (BPDUs) generated by other vendors' equipment. This is the default behavior on all supported PICs and requires no additional configuration.

The following PICs are supported on T Series Core Routers and the M320 Multiservice Edge router and can transmit nonstandard BPDUs:

- 1-port Gigabit Ethernet PIC
- 2-port Gigabit Ethernet PIC

- 4-port Gigabit Ethernet PIC
- 10-port Gigabit Ethernet PIC

Pinging VPNs, VPLS, and Layer 2 Circuits

For testing purposes, you can ping Layer 2 VPNs, Layer 3 VPNs, and Layer 2 circuits by using the **ping mpls** command. The **ping mpls** command helps to verify that a VPN or circuit has been enabled and tests the integrity of the VPN or Layer 2 circuit connection between the PE routers. It does not test the connection between a PE router and a CE router. To ping a VPLS routing instance, you issue a **ping vpls instance** command (see "Pinging a VPLS Routing Instance" on page 17).

You issue the **ping mpls** command from the ingress PE router of the VPN or Layer 2 circuit to the egress PE router of the same VPN or Layer 2 circuit. When you execute the **ping mpls** command, echo requests are sent as MPLS packets.

The payload is a User Datagram Protocol (UDP) packet forwarded to the address **127.0.0.1**. The contents of this packet are defined in RFC 4379, *Detecting Multi-Protocol Label Switched (MPLS) Data Plane Failures*. The label and interface information for building and sending this information as an MPLS packet is the same as for standard VPN traffic, but the time-to-live (TTL) of the innermost label is set to 1.

When the echo request arrives at the egress PE router, the contents of the packet are checked, and then a reply that contains the correct return is sent by means of UDP. The PE router sending the echo request waits to receive an echo reply after a timeout of 2 seconds (you cannot configure this value).

You must configure MPLS at the **[edit protocols mpls]** hierarchy level on the egress PE router (the router receiving the MPLS echo packets) to be able to ping the VPN or Layer 2 circuit. You must also configure the address **127.0.0.1/32** on the egress PE router's **lo0** interface. If this is not configured, the egress PE router does not have this forwarding entry and therefore simply drops the incoming MPLS pings.

The **ping mpls** command has the following limitations:

- You cannot ping an IPv6 destination prefix.
- You cannot ping a VPN or Layer 2 circuit from a router that is attempting a graceful restart.
- You cannot ping a VPN or Layer 2 circuit from a logical system.

You can also determine whether an LSP linking two PE routers in a VPN is up by pinging the end point address of the LSP. The command you use to ping an MPLS LSP end point is **ping mpls lsp-end-point *address***. This command tells you what type of LSP (RSVP or LDP) terminates at the address specified and whether that LSP is up or down.

For a detailed description of this command, see the *Junos Routing Protocols and Policies Command Reference*.

Setting the Forwarding Class of the Ping Packets

When you execute the **ping mpls** command, the ping packets forwarded to the destination include MPLS labels. It is possible to set the value of the forwarding class for these ping packets by using the **exp** option with the **ping mpls** command. For example, to set the forwarding class to 5 when pinging a Layer 3 VPN, issue the following command:

```
ping mpls l3vpn westcoast source 1.1.1.1 prefix 2.2.2.2 exp 5 count 20 detail
```

This command would makes the router attempt to ping the Layer 3 VPN **westcoast** using ping packets with an EXP forwarding class of 5. The default forwarding class used for the **ping mpls** command packets is 7.

Pinging a VPLS Routing Instance

The **ping vpls instance** command uses a difference command structure and operates in a different fashion than the **ping mpls** command used for VPNs and Layer 2 circuits. The **ping vpls instance** command is only supported on MX Series routers, the M120 router, the M320 router, and the T1600 router.

To ping a VPLS routing instance, use the following command:

```
ping vpls instance instance-name destination-mac address source-ip address <count
  number> <data-plane-response> <detail> <learning-vlan-id number> <logical-system
  logical-system-name>
```

You ping a combination of the routing instance name, the destination MAC address, and the source IP address. When you enter this command, you are provided feedback on the status of your request. An exclamation point (!) indicates that an echo reply was received. A period (.) indicates that an echo reply was not received within the timeout period. An **x** indicates that an echo reply was received with an error code these packets are not counted in the received packets count. They are accounted for separately.

PART 2

Configuration

CHAPTER 3

Configuring VPNs

Configuring an IGP on the PE and P Routers

For Layer 2 VPNs, Layer 3 VPNs, virtual-router routing instances, VPLS, EVPNs, and Layer 2 circuits to function properly, the service provider's PE and P routers must be able to exchange routing information. For this to happen, you must configure either an IGP (such as OSPF or IS-IS) or static routes on these routers. You configure the IGP on the master instance of the routing protocol process at the **[edit protocols]** hierarchy level, not within the routing instance used for the VPN—that is, not at the **[edit routing-instances]** hierarchy level.

When you configure the PE router, do not configure any summarization of the PE router's loopback addresses at the area boundary. Each PE router's loopback address should appear as a separate route.

Related Documentation

- *Example: Configuring IS-IS*
- *Examples: Configuring Static Routes*
- *OSPF Feature Guide for Routing Devices*

Configuring IBGP Sessions Between PE Routers in VPNs

You must configure an IBGP session between the PE routers to allow the PE routers to exchange information about routes originating and terminating in the VPN. The PE routers rely on this information to determine which labels to use for traffic destined for remote sites.

Configure an IBGP session for the VPN as follows:

```
[edit protocols]
bgp {
  group group-name {
    type internal;
    local-address ip-address;
    family evpn {
      signaling;
    }
    family (inet-vpn | inet6-vpn) {
      unicast;
    }
    family l2vpn {
      signaling;
    }
    neighbor ip-address;
  }
}
```

The IP address in the **local-address** statement is the address of the loopback interface on the local PE router. The IBGP session for the VPN runs through the loopback address. (You must also configure the loopback interface at the **[edit interfaces]** hierarchy level.)

The IP address in the **neighbor** statement is the loopback address of the neighboring PE router. If you are using RSVP signaling, this IP address is the same address you specify in the **to** statement at the **[edit mpls label-switched-path *lsp-path-name*]** hierarchy level when you configure the MPLS LSP.

The **family** statement allows you to configure the IBGP session for Layer 2 VPNs, VPLS, EVPNs or for Layer 3 VPNs.

- To configure an IBGP session for Layer 2 VPNs and VPLS, include the **signaling** statement at the **[edit protocols bgp group *group-name* family l2vpn]** hierarchy level:

  ```
  [edit protocols bgp group group-name family l2vpn]
  signaling;
  ```

- To configure an IBGP session for EVPNs, include the **signaling** statement at the **[edit protocols bgp group *group-name* family evpn]** hierarchy level:

  ```
  [edit protocols bgp group group-name family evpn]
  signaling;
  ```

- To configure an IPv4 IBGP session for Layer 3 VPNs, configure the **unicast** statement at the **[edit protocols bgp group *group-name* family inet-vpn]** hierarchy level:

  ```
  [edit protocols bgp group group-name family inet-vpn]
  ```

```
unicast;
```

- To configure an IPv6 IBGP session for Layer 3 VPNs, configure the **unicast** statement at the **[edit protocols bgp group *group-name* family inet6-vpn]** hierarchy level:

  ```
  [edit protocols bgp group group-name family inet6-vpn]
  unicast;
  ```

NOTE: You can configure both **family inet** and **family inet-vpn** or both **family inet6** and **family inet6-vpn** within the same peer group. This allows you to enable support for both IPv4 and IPv4 VPN routes or both IPv6 and IPv6 VPN routes within the same peer group.

Related Documentation

- Configuring an IGP on the PE and P Routers on page 21
- Configuring a Signaling Protocol and LSPs for VPNs on page 23

Configuring a Signaling Protocol and LSPs for VPNs

For VPNs to function, you must enable a signaling protocol, either the LDP or RSVP on the provider edge (PE) routers and on the provider (P) routers. You also need to configure label-switched paths (LSPs) between the ingress and egress routers. In a typical VPN configuration, you need to configure LSPs from each PE router to all of the other PE routers participating in the VPN in a full mesh.

NOTE: As with any configuration involving MPLS, you cannot configure any of the core-facing interfaces on the PE routers over dense Fast Ethernet PICs.

To enable a signaling protocol, perform the steps in one of the following sections:

- Using LDP for VPN Signaling on page 23
- Using RSVP for VPN Signaling on page 25

Using LDP for VPN Signaling

To use LDP for VPN signaling, perform the following steps on the PE and provider (P) routers:

1. Configure LDP on the interfaces in the core of the service provider's network by including the **ldp** statement at the **[edit protocols]** hierarchy level.

   You need to configure LDP only on the interfaces between PE routers or between PE and P routers. You can think of these as the "core-facing" interfaces. You do not need to configure LDP on the interface between the PE and customer edge (CE) routers.

   ```
   [edit]
   protocols {
     ldp {
       interface type-fpc/pic/port;
   ```

```
  }
}
```

2. Configure the MPLS address family on the interfaces on which you enabled LDP (the interfaces you configured in Step 1) by including the **family mpls** statement at the **[edit interfaces *type-fpc/pic/port* unit *logical-unit-number*]** hierarchy level.

```
[edit]
interfaces {
  type-fpc/pic/port {
    unit logical-unit-number {
      family mpls;
    }
  }
}
```

3. Configure OSPF or IS-IS on each PE and P router.

   You configure these protocols at the master instance of the routing protocol, not within the routing instance used for the VPN.

   - To configure OSPF, include the **ospf** statement at the **[edit protocols]** hierarchy level. At a minimum, you must configure a backbone area on at least one of the router's interfaces.

```
[edit]
protocols {
  ospf {
    area 0.0.0.0 {
      interface type-fpc/pic/port;
    }
  }
}
```

   - To configure IS-IS, include the **isis** statement at the **[edit protocols]** hierarchy level and configure the loopback interface and International Organization for Standardization (ISO) family at the **[edit interfaces]** hierarchy level. At a minimum, you must enable IS-IS on the router, configure a network entity title (NET) on one of the router's interfaces (preferably the loopback interface, lo0), and configure the ISO family on all interfaces on which you want IS-IS to run. When you enable IS-IS, Level 1 and Level 2 are enabled by default. The following is the minimum IS-IS configuration. In the **address** statement, ***address*** is the NET.

```
[edit]
interfaces {
  lo0 {
    unit logical-unit-number {
      family iso {
        address address;
      }
    }
  }
  type-fpc/pic/port {
    unit logical-unit-number {
      family iso;
    }
  }
```

```
}
protocols {
   isis {
      interface all;
   }
}
```

For more information about configuring OSPF and IS-IS, see the *OSPF Feature Guide for Routing Devices* and *IS-IS Feature Guide for Routing Devices*.

Using RSVP for VPN Signaling

To use RSVP for VPN signaling, perform the following steps:

1. On each PE router, configure traffic engineering.

   To do this, you must configure an interior gateway protocol (IGP) that supports traffic engineering (either IS-IS or OSPF) and enable traffic engineering support for that protocol.

   To enable OSPF traffic engineering support, include the **traffic-engineering** statement at the **[edit protocols ospf]** hierarchy level:

   ```
   [edit protocols ospf]
   traffic-engineering {
      shortcuts;
   }
   ```

   For IS-IS, traffic engineering support is enabled by default.

2. On each PE and P router, enable RSVP on the interfaces that participate in the label-switched path (LSP).

   On the PE router, these interfaces are the ingress and egress points to the LSP. On the P router, these interfaces connect the LSP between the PE routers. Do not enable RSVP on the interface between the PE and the CE routers, because this interface is not part of the LSP.

   To configure RSVP on the PE and P routers, include the **interface** statement at the **[edit protocols rsvp]** hierarchy level. Include one **interface** statement for each interface on which you are enabling RSVP.

   ```
   [edit protocols]
   rsvp {
      interface interface-name;
      interface interface-name;
   }
   ```

3. On each PE router, configure an MPLS LSP to the PE router that is the LSP's egress point.

   To do this, include the **interface** and **label-switched-path** statements at the **[edit protocols mpls]** hierarchy level:

   ```
   [edit protocols]
   mpls {
      interface interface-name;
      label-switched-path path-name {
   ```

```
        to ip-address;
    }
}
```

In the **to** statement, specify the address of the LSP's egress point, which is an address on the remote PE router.

In the **interface** statement, specify the name of the interface (both the physical and logical portions). Include one **interface** statement for the interface associated with the LSP.

When you configure the logical portion of the same interface at the **[edit interfaces]** hierarchy level, you must also configure the **family inet** and **family mpls** statements:

```
[edit interfaces]
interface-name {
    unit logical-unit-number {
        family inet;
        family mpls;
    }
}
```

4. On all P routers that participate in the LSP, enable MPLS by including the **interface** statement at the **[edit mpls]** hierarchy level.

   Include one **interface** statement for each connection to the LSP.

   ```
   [edit]
   mpls {
       interface interface-name;
       interface interface-name;
   }
   ```

5. Enable MPLS on the interface between the PE and CE routers by including the **interface** statement at the **[edit mpls]** hierarchy level.

   Doing this allows the PE router to assign an MPLS label to traffic entering the LSP or to remove the label from traffic exiting the LSP.

   ```
   [edit]
   mpls {
       interface interface-name;
   }
   ```

   For information about configuring MPLS, see the *Minimum MPLS Configuration*.

Related Documentation

- *Minimum MPLS Configuration*9

Configuring Routing Instances on PE Routers in VPNs

You need to configure a routing instance for each VPN on each of the PE routers participating in the VPN. The configuration procedures outlined in this section are applicable to Layer 2 VPNs, Layer 3 VPNs, and VPLS. The configuration procedures specific to each type of VPN are described in the corresponding sections in the other configuration chapters.

To configure routing instances for VPNs, include the following statements:

```
description text;
instance-type type;
interface interface-name;
route-distinguisher (as-number:number | ip-address:number);
vrf-import [ policy-names ];
vrf-export [ policy-names ];
vrf-target {
    export community-name;
    import community-name;
}
```

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*]**

To configure VPN routing instances, you perform the steps in the following sections:

Configuring the Routing Instance Name for a VPN

The name of the routing instance for a VPN can be a maximum of 128 characters and can contain letters, numbers, and hyphens. In Junos OS Release 9.0 and later, you can no longer specify **default** as the actual routing-instance name. You also cannot use any special characters (! @ # $ % ^ & * , +< > : ;) within the name of a routing instance.

NOTE: In Junos OS Release 9.6 and later, you can include a slash (/) in a routing instance name only if a logical system is not configured. That is, you cannot include the slash character in a routing instance name if a logical system other than the default is explicitly configured.

Specify the routing-instance name with the **routing-instance** statement:

```
routing-instance routing-instance-name {...}
```

You can include this statement at the following hierarchy levels:

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

Configuring the Description

To provide a text description for the routing instance, include the **description** statement. If the text includes one or more spaces, enclose them in quotation marks (" "). Any descriptive text you include is displayed in the output of the **show route instance detail** command and has no effect on the operation of the routing instance.

To configure a text description, include the **description** statement:

```
description text;
```

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 Instance Type

The instance type you configure varies depending on whether you are configuring Layer 2 VPNs, Layer 3 VPNs, VPLS, or virtual routers. Specify the instance type by including the **instance-type** statement:

- To enable Layer 2 VPN routing on a PE router, include the **instance-type** statement and specify the value **l2vpn**:

  ```
  instance-type l2vpn;
  ```

- To enable VPLS routing on a PE router, include the **instance-type** statement and specify the value **vpls**:

  ```
  instance-type vpls;
  ```

- Layer 3 VPNs require that each PE router have a VPN routing and forwarding (VRF) table for distributing routes within the VPN. To create the VRF table on the PE router, include the **instance-type** statement and specify the value **vrf**:

  ```
  instance-type vrf;
  ```

  NOTE: Routing Engine based sampling is not supported on VRF routing instances.

- To enable the virtual-router routing instance, include the **instance-type** statement and specify the value **virtual-router**:

  ```
  instance-type virtual-router;
  ```

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 Interfaces for VPN Routing

On each PE router, you must configure an interface over which the VPN traffic travels between the PE and CE routers.

The sections that follow describe how to configure interfaces for VPNs:

General Configuration for VPN Routing

The configuration described in this section applies to all types of VPNs. For Layer 3 VPNs and carrier-of-carriers VPNs, complete the configuration described in this section before proceeding to the interface configuration sections specific to those topics.

To configure interfaces for VPN routing, include the **interface** statement:

```
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***]**

Specify both the physical and logical portions of the interface name, in the following format:

```
physical.logical
```

For example, in **at-1/2/1.2**, **at-1/2/1** is the physical portion of the interface name and **2** is the logical portion. If you do not specify the logical portion of the interface name, the value **0** is set by default.

A logical interface can be associated with only one routing instance. If you enable a routing protocol on all instances by specifying **interfaces all** when configuring the master instance of the protocol at the **[edit protocols]** hierarchy level, and if you configure a specific interface for VPN routing at the **[edit routing-instances** *routing-instance-name***]** hierarchy level or at the **[edit logical-systems** *logical-system-name* **routing-instances** *routing-instance-name***]** hierarchy level, the latter interface statement takes precedence and the interface is used exclusively for the VPN.

If you explicitly configure the same interface name at the **[edit protocols]** hierarchy level and at either the **[edit routing-instances** *routing-instance-name***]** or **[edit logical-systems** *logical-system-name* **routing-instances** *routing-instance-name***]** hierarchy levels, an attempt to commit the configuration fails.

Configuring Interfaces for Layer 3 VPNs

When you configure the Layer 3 VPN interfaces at the **[edit interfaces]** hierarchy level, you must also configure **family inet** when configuring the logical interface:

```
[edit interfaces]
interface-name {
    unit logical-unit-number {
        family inet;
    }
}
```

Configuring Interfaces for Carrier-of-Carriers VPNs

When you configure carrier-of-carriers VPNs, you need to configure the **family mpls** statement in addition to the **family inet** statement for the interfaces between the PE and CE routers. For carrier-of-carriers VPNs, configure the logical interface as follows:

```
[edit interfaces]
interface-name {
    unit logical-unit-number {
        family inet;
        family mpls;
    }
}
```

If you configure **family mpls** on the logical interface and then configure this interface for a non-carrier-of-carriers routing instance, the **family mpls** statement is automatically removed from the configuration for the logical interface, since it is not needed.

Configuring Unicast RPF on VPN Interfaces

For VPN interfaces that carry IP version 4 or version 6 (IPv4 or IPv6) traffic, you can reduce the impact of denial-of-service (DoS) attacks by configuring unicast reverse path forwarding (RPF). Unicast RPF helps determine the source of attacks and rejects packets from unexpected source addresses on interfaces where unicast RPF is enabled.

You can configure unicast RPF on a VPN interface by enabling unicast RPF on the interface and including the **interface** statement at the **[edit routing-instances routing-instance-name]** hierarchy level.

You cannot configure unicast RPF on the core-facing interfaces. You can only configure unicast RPF on the CE router-to-PE router interfaces on the PE router. However, for virtual-router routing instances, unicast RPF is supported on all interfaces you specify in the routing instance.

For information about how to configure unicast RPF on VPN interfaces, see *Configuring Unicast RPF*.

Configuring the Route Distinguisher

Each routing instance that you configure on a PE router must have a unique route distinguisher associated with it. VPN routing instances need a route distinguisher to help BGP to distinguish between potentially identical network layer reachability information

(NLRI) messages received from different VPNs. If you configure different VPN routing instances with the same route distinguisher, the commit fails.

For Layer 2 VPNs and VPLS, if you have configured the **l2vpn-use-bgp-rules** statement, you must configure a unique route distinguisher for each PE router participating in a specific routing instance.

For other types of VPNs, we recommend that you use a unique route distinguisher for each PE router participating in the routing instance. Although you can use the same route distinguisher on all PE routers for the same VPN routing instance (except for Layer 2 VPNs and VPLS), if you use a unique route distinguisher, you can determine the CE router from which a route originated within the VPN.

To configure a route distinguisher on a PE router, include the **route-distinguisher** statement:

```
route-distinguisher (as-number:number | ip-address:number);
```

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

The 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 autonomous system (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.

Configuring Automatic Route Distinguishers

If you configure the **route-distinguisher-id** statement at the **[edit routing-options]** hierarchy level, a route distinguisher is automatically assigned to the routing instance. If you also configure the **route-distinguisher** statement in addition to the **route-distinguisher-id** statement, the value configured for **route-distinguisher** supersedes the value generated from **route-distinguisher-id**.

To assign a route distinguisher automatically, include the **route-distinguisher-id** statement:

```
route-distinguisher-id ip-address;
```

You can include this statement at the following hierarchy levels:

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

A type 1 route distinguisher is automatically assigned to the routing instance using the format ***ip-address*:*number***. The IP address is specified by the **route-distinguisher-id** statement and the number is unique for the routing instance.

Related Documentation

- Configuring Policies for the VRF Table on PE Routers in VPNs on page 32
- Configuring BGP Route Target Filtering for VPNs on page 38

Configuring Policies for the VRF Table on PE Routers in VPNs

On each PE router, you must define policies that define how routes are imported into and exported from the router's VRF table. In these policies, you must define the route target, and you can optionally define the route origin.

To configure policy for the VRF tables, you perform the steps in the following sections:

- Configuring the Route Target on page 32
- Configuring the Route Origin on page 33
- Configuring an Import Policy for the PE Router's VRF Table on page 34
- Configuring an Export Policy for the PE Router's VRF Table on page 35
- Applying Both the VRF Export and the BGP Export Policies on page 37
- Configuring a VRF Target on page 37

Configuring the Route Target

As part of the policy configuration for the VPN routing table, you must define a route target, which defines which VPN the route is a part of. When you configure different types of VPN services (Layer 2 VPNs, Layer 3 VPNs, EVPNs, or VPLS) on the same PE router, be sure to assign unique route target values to avoid the possibility of adding route and signaling information to the wrong VPN routing table.

To configure the route target, include the **target** option in the **community** statement:

community *name* members target:*community-id*;

You can include this statement at the following hierarchy levels:

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

name is the name of the community.

community-id is the identifier of the community. Specify it in one of the following formats:

- ***as-number*:*number***, where ***as-number*** is an AS number (a 2-byte value) and ***number*** is a 4-byte community value. The AS number can be in the range 1 through 65,535. We recommend that you use an IANA-assigned, nonprivate AS number, preferably the ISP's own or the customer's own AS number. The community value can be a number in the range 0 through 4,294,967,295 ($2^{32} - 1$).

- ***ip-address*:*number***, where ***ip-address*** is an IPv4 address (a 4-byte value) and ***number*** is a 2-byte community 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. The community value can be a number in the range 1 through 65,535.

Configuring the Route Origin

In the import and export policies for the PE router's VRF table, you can optionally assign the route origin (also known as the site of origin) for a PE router's VRF routes using a VRF export policy applied to multiprotocol external BGP (MP-EBGP) VPN IPv4 route updates sent to other PE routers.

Matching on the assigned route origin attribute in a receiving PE's VRF import policy helps ensure that VPN-IPv4 routes learned through MP-EBGP updates from one PE are not reimported to the same VPN site from a different PE connected to the same site.

To configure a route origin, complete the following steps:

1. Include the **community** statement with the **origin** option:

   community *name* members origin:*community-id*;

   You can include this statement at the following hierarchy levels:

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

   name is the name of the community.

   community-id is the identifier of the community. Specify it in one of the following formats:

   - ***as-number*:*number***, where ***as-number*** is an AS number (a 2-byte value) and ***number*** is a 4-byte community value. The AS number can be in the range 1 through 65,535. We recommend that you use an IANA-assigned, nonprivate AS number, preferably the ISP's own or the customer's own AS number. The community value can be a number in the range 0 through 4,294,967,295 ($2^{32} - 1$).
   - ***ip-address*:*number***, where ***ip-address*** is an IPv4 address (a 4-byte value) and ***number*** is a 2-byte community 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. The community value can be a number in the range 1 through 65,535.

2. Include the community in the import policy for the PE router's VRF table by configuring the **community** statement with the ***community-id*** identifier defined in Step 1 at the **[edit policy-options policy-statement *import-policy-name* term *import-term-name* from]** hierarchy level. See "Configuring an Import Policy for the PE Router's VRF Table" on page 34.

If the policy's **from** clause does not specify a community condition, the **vrf-import** statement in which the policy is applied cannot be committed. The Junos OS commit operation does not pass the validation check.

3. Include the community in the export policy for the PE router's VRF table by configuring the **community** statement with the ***community-id*** identifier defined in Step 1 at the **[edit policy-options policy-statement *export-policy-name* term *export-term-name* then]** hierarchy level. See "Configuring an Export Policy for the PE Router's VRF Table" on page 35.

See "Route Origin for VPNs" on page 61 for a configuration example.

Configuring an Import Policy for the PE Router's VRF Table

Each VPN can have a policy that defines how routes are imported into the PE router's VRF table. An import policy is applied to routes received from other PE routers in the VPN. A policy must evaluate all routes received over the IBGP session with the peer PE router. If the routes match the conditions, the route is installed in the PE router's ***routing-instance-name*.inet.0** VRF table. An import policy must contain a second term that rejects all other routes.

Unless an import policy contains only a **then reject** statement, it must include a reference to a community. Otherwise, when you try to commit the configuration, the commit fails. You can configure multiple import policies.

An import policy determines what to import to a specified VRF table based on the VPN routes learned from the remote PE routers through IBGP. The IBGP session is configured at the **[edit protocols bgp]** hierarchy level. If you also configure an import policy at the **[edit protocols bgp]** hierarchy level, the import policies at the **[edit policy-options]** hierarchy level and the **[edit protocols bgp]** hierarchy level are combined through a logical AND operation. This allows you to filter traffic as a group.

To configure an import policy for the PE router's VRF table, follow these steps:

1. To define an import policy, include the **policy-statement** statement. For all PE routers, an import policy must always include the **policy-statement** statement, at a minimum:

```
policy-statement import-policy-name {
  term import-term-name {
    from {
      protocol bgp;
      community community-id;
    }
    then accept;
  }
  term term-name {
    then reject;
  }
}
```

You can include the **policy-statement** statement at the following hierarchy levels:

- **[edit policy-options]**

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

The ***import-policy-name*** policy evaluates all routes received over the IBGP session with the other PE router. If the routes match the conditions in the **from** statement, the route is installed in the PE router's *routing-instance-name*.inet.0 VRF table. The second term in the policy rejects all other routes.

For more information about creating policies, see the *Routing Policy Feature Guide for Routing Devices*.

2. You can optionally use a regular expression to define a set of communities to be used for the VRF import policy.

   For example you could configure the following using the **community** statement at the **[edit policy-options policy-statement *policy-statement-name*]** hierarchy level:

   ```
   [edit policy-options vrf-import-policy-sample]
   community high-priority members *:50
   ```

   Note that you cannot configure a regular expression as a part of a route target extended community. For more information about how to configure regular expressions for communities, see *Understanding How to Define BGP Communities and Extended Communities* .

3. To configure an import policy, include the **vrf-import** statement:

   ```
   vrf-import import-policy-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 an Export Policy for the PE Router's VRF Table

Each VPN can have a policy that defines how routes are exported from the PE router's VRF table. An export policy is applied to routes sent to other PE routers in the VPN. An export policy must evaluate all routes received over the routing protocol session with the CE router. (This session can use the BGP, OSPF, or Routing Information Protocol [RIP] routing protocols, or static routes.) If the routes match the conditions, the specified community target (which is the route target) is added to them and they are exported to the remote PE routers. An export policy must contain a second term that rejects all other routes.

Export policies defined within the VPN routing instance are the only export policies that apply to the VRF table. Any export policy that you define on the IBGP session between the PE routers has no effect on the VRF table. You can configure multiple export policies.

To configure an export policy for the PE router's VRF table, follow these steps:

1. For all PE routers, an export policy must distribute VPN routes to and from the connected CE routers in accordance with the type of routing protocol that you configure between the CE and PE routers within the routing instance.

To define an export policy, include the **policy-statement** statement. An export policy must always include the **policy-statement** statement, at a minimum:

```
policy-statement export-policy-name {
  term export-term-name {
    from protocol (bgp | ospf | rip | static);
    then {
      community add community-id;
      accept;
    }
  }
  term term-name {
    then reject;
  }
}
```

NOTE: Configuring the **community add** statement is a requirement for Layer 2 VPN VRF export policies. If you change the **community add** statement to the **community set** statement, the router at the egress of the Layer 2 VPN link might drop the connection.

NOTE: When configuring draft-rosen multicast VPNs operating in source-specific mode and using the **vrf-export** statement to specify the export policy, the policy must have a term that accepts routes from the vrf-name.mdt.0 routing table. This term ensures proper PE autodiscovery using the **inet-mdt** address family.

When configuring draft-rosen multicast VPNs operating in source-specific mode and using the **vrf-target** statement, the VRF export policy is automatically generated and automatically accepts routes from the vrf-name.mdt.0 routing table.

You can include the **policy-statement** statement at the following hierarchy levels:

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

The ***export-policy-name*** policy evaluates all routes received over the routing protocol session with the CE router. (This session can use the BGP, OSPF, or RIP routing protocols, or static routes.) If the routes match the conditions in the **from** statement, the community target specified in the **then community add** statement is added to them and they are exported to the remote PE routers. The second term in the policy rejects all other routes.

For more information about creating policies, see the *Routing Policy Feature Guide for Routing Devices*.

2. To apply the policy, include the **vrf-export** statement:

   vrf-export *export-policy-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***]**

Applying Both the VRF Export and the BGP Export Policies

When you apply a VRF export policy as described in "Configuring an Export Policy for the PE Router's VRF Table" on page 35, routes from VPN routing instances are advertised to other PE routers based on this policy, whereas the BGP export policy is ignored.

If you include the **vpn-apply-export** statement in the BGP configuration, both the VRF export and BGP group or neighbor export policies are applied (VRF first, then BGP) before routes are advertised in the VPN routing tables to other PE routers.

When you include the **vpn-apply-export** statement, be aware of the following:

- Routes imported into the l3vpn.bgp.0 routing table retain the attributes of the original routes (for example, an OSPF route remains an OSPF route even when it is stored in the l3vpn.bgp.0 routing table). You should be aware of this when you configure an export policy for connections between an IBGP PE router and a PE router, a route reflector and a PE router, or AS boundary router (ASBR) peer routers.
- By default, all routes in the l3vpn.bgp.0 routing table are exported to the IBGP peers. If the last statement of the export policy is deny all and if the export policy does not specifically match on routes in the l3vpn.bgp.0 routing table, no routes are exported.

To apply both the VRF export and BGP export policies to VPN routes, include the **vpn-apply-export** statement:

```
vpn-apply-export;
```

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

Configuring a VRF Target

Including the **vrf-target** statement in the configuration for a VRF target community causes default VRF import and export policies to be generated that accept and tag routes with the specified target community. You can still create more complex policies by explicitly configuring VRF import and export policies. These policies override the default policies generated when you configure the **vrf-target** statement.

If you do not configure the **import** and **export** options of the **vrf-target** statement, the specified community string is applied in both directions. The **import** and **export** keywords give you more flexibility, allowing you to specify a different community for each direction.

The syntax for the VRF target community is not a name. You must specify it in the format **target:***x*:*y*. A community name cannot be specified because this would also require you to configure the community members for that community using the **policy-options** statement. If you define the **policy-options** statements, then you can just configure VRF import and export policies as usual. The purpose of the **vrf-target** statement is to simplify

the configuration by allowing you to configure most statements at the **[edit routing-instances]** hierarchy level.

To configure a VRF target, include the **vrf-target** statement:

```
vrf-target community;
```

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***]**

An example of how you might configure the **vrf-target** statement follows:

```
[edit routing-instances sample]
vrf-target target:69:102;
```

To configure the **vrf-target** statement with the **export** and **import** options, include the following statements:

```
vrf-target {
    export community-name;
    import community-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 BGP Route Target Filtering for VPNs

BGP route target filtering allows you to distribute VPN routes to only the routers that need them. In VPN networks without BGP route target filtering configured, BGP distributes all VPN routes to all VPN peer routers.

For more information about BGP route target filtering, see RFC 4684, *Constrained Route Distribution for Border Gateway Protocol/MultiProtocol Label Switching (BGP/MPLS) Internet Protocol (IP) Virtual Private Networks (VPNs)*.

The following sections provide an overview of BGP route target filtering and how to configure it for VPNs:

BGP Route Target Filtering Overview

PE routers, unless they are configured as route reflectors or are running an EBGP session, discard any VPN routes that do not include a route target extended community as specified in the local VRF import policies. This is the default behavior of the Junos OS.

However, unless it is explicitly configured not to store VPN routes, any router configured either as a route reflector or border router for a VPN address family must store all of the VPN routes that exist in the service provider's network. Also, though PE routers can automatically discard routes that do not include a route target extended community, route updates continue to be generated and received.

By reducing the number of routers receiving VPN routes and route updates, BGP route target filtering helps to limit the amount of overhead associated with running a VPN. BGP route target filtering is most effective at reducing VPN-related administrative traffic in networks where there are many route reflectors or AS border routers that do not participate in the VPNs directly (not acting as PE routers for the CE devices).

BGP route target filtering uses standard UPDATE messages to distributes route target extended communities between routers. The use of UPDATE messages allows BGP to use its standard loop detection mechanisms, path selection, policy support, and database exchange implementation.

Configuring BGP Route Target Filtering for VPNs

BGP route target filtering is enabled through the exchange of the **route-target** address family, stored in the bgp.rtarget.0 routing table. Based on the **route-target** address family, the route target NLRI (address family indicator [AFI]=1, subsequent AFI [SAFI]=132) is negotiated with its peers.

On a system that has locally configured VRF instances, BGP automatically generates local routes corresponding to targets referenced in the **vrf-import** policies.

To configure BGP route target filtering, include the **family route-target** statement:

```
family route-target {
    advertise-default;
    external-paths number;
    prefix-limit number;
}
```

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

The **advertise-default**, **external-paths**, and **prefix-limit** statements affect the BGP route target filtering configuration as follows:

- The **advertise-default** statement causes the router to advertise the default route target route (0:0:0/0) and suppress all routes that are more specific. This can be used by a route reflector on BGP groups consisting of neighbors that act as PE routers only. PE routers often need to advertise all routes to the route reflector.

  Suppressing all route target advertisements other than the default route reduces the amount of information exchanged between the route reflector and the PE routers. The Junos OS further helps to reduce route target advertisement overhead by not maintaining dependency information unless a nondefault route is received.

- The **external-paths** statement (which has a default value of 1) causes the router to advertise the VPN routes that reference a given route target. The number you specify

determines the number of external peer routers (currently advertising that route target) that receive the VPN routes.

- The **prefix-limit** statement limits the number of prefixes that can be received from a peer router.

The **route-target**, **advertise-default**, and **external-path** statements affect the RIB-OUT state and must be consistent between peer routers that share the same BGP group. The **prefix-limit** statement affects the receive side only and can have different settings between different peer routers in a BGP group.

Related Documentation

- Example: Configuring Proxy BGP Route Target Filtering on page 65
- Example: Configuring an Export Policy for BGP Route Target Filtering on page 81
- Example: Configuring BGP Route Target Filtering for VPNs on page 53
- Route Origin for VPNs on page 61
- Configuring Static Route Target Filtering for VPNs on page 40

Configuring Static Route Target Filtering for VPNs

The BGP VPN route target extended community (RFC 4360, *BGP Extended Communities Attribute*) is used to determine VPN membership. Static route target filtering helps to prevent resources from being consumed in portions of the network where the VPN routes are not needed due to the lack of member PE routers (RFC 4684, *Constrained Route Distribution for Border Gateway Protocol/MultiProtocol Label Switching (BGP/MPLS) Internet Protocol (IP) Virtual Private Networks (VPNs)*). Routers can originate routes into the RT-Constrain protocol to indicate their interest in receiving VPN routes containing route targets that match the RT-Constrain NLRI.

To configure static route target filtering for VPNs:

- Configure the **route-target-filter** statement at the **[edit routing-options rib bgp.rtarget.0 static]** hierarchy level.

  The following example illustrates how you could configure the **route-target-filter** statement:

  ```
  [edit routing-options rib bgp.rtarget.0 static]
  route-target-filter destination {
     group bgp-group;
     local;
     neighbor bgp-peer;
  }
  ```

- You can display route target filtering information using the **show bgp group rtf detail** command.

Related Documentation

- Reducing Network Resource Use with Static Route Target Filtering on page 8
- Configuring BGP Route Target Filtering for VPNs on page 38

Configuring Virtual-Router Routing Instances in VPNs

A virtual-router routing instance, like a VRF routing instance, maintains separate routing and forwarding tables for each instance. However, many of the configuration steps required for VRF routing instances are not required for virtual-router routing instances. Specifically, you do not need to configure a route distinguisher, a routing table policy (the **vrf-export**, **vrf-import**, and **route-distinguisher** statements), or MPLS between the service provider routers.

Configure a virtual-router routing instance by including the following statements:

```
description text;
instance-type virtual-router;
interface interface-name;
protocols { ... }
```

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*]**

The following sections explain how to configure a virtual-router routing instance:

- Configuring a Routing Protocol Between the Service Provider Routers on page 41
- Configuring Logical Interfaces Between Participating Routers on page 42

Configuring a Routing Protocol Between the Service Provider Routers

The service provider routers need to be able to exchange routing information. You can configure the following protocols for the virtual-router routing instance **protocols** statement configuration at the **[edit routing-instances *routing-instance-name*]** hierarchy level:

- BGP
- IS-IS
- LDP
- OSPF
- Protocol Independent Multicast (PIM)
- RIP

You can also configure static routes.

IBGP route reflection is not supported for virtual-router routing instances.

If you configure LDP under a virtual-router instance, LDP routes are placed by default in the routing instance's inet.0 and inet.3 routing tables (for example, sample.inet.0 and sample.inet.3). To restrict LDP routes to only the routing instance's inet.3 table, include the **no-forwarding** statement:

no-forwarding;

You can include this statement at the following hierarchy levels:

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

When you restrict the LDP routes to only the inet.3 routing table, the corresponding IGP route in the inet.0 routing table can be redistributed and advertised into other routing protocols.

For information about routing tables, see *Understanding Junos OS Routing Tables*.

Configuring Logical Interfaces Between Participating Routers

You must configure an interface to each customer router participating in the routing instance and to each P router participating in the routing instance. Each virtual-router routing instance requires its own separate logical interfaces to all P routers participating in the instance. To configure interfaces for virtual-router instances, include the **interface** statement:

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*]**

Specify both the physical and logical portions of the interface name, in the following format:

physical.logical

For example, in **at-1/2/1.2**, **at-1/2/1** is the physical portion of the interface name and **2** is the logical portion. If you do not specify the logical portion of the interface name, **0** is set by default.

You must also configure the interfaces at the **[edit interfaces]** hierarchy level.

One method of providing this logical interface between the provider routers is by configuring tunnels between them. You can configure IP Security (IPsec), generic routing encapsulation (GRE), or IP-IP tunnels between the provider routers, terminating the tunnels at the virtual-router instance.

For information about how to configure tunnels and interfaces, see the *Junos OS Services Interfaces Library for Routing Devices*.

Configuring Graceful Restart for VPNs

Graceful restart allows a router whose VPN control plane is undergoing a restart to continue to forward traffic while recovering its state from neighboring routers. Without graceful restart, a control plane restart disrupts any VPN services provided by the router.

Graceful restart is supported on Layer 2 VPNs, Layer 3 VPNs, virtual-router routing instances, and VPLS.

To enable VPN graceful restart, include the **graceful-restart** statement:

```
graceful-restart {
    disable;
    restart-duration time-limit;
}
```

To configure graceful restart globally, include the **graceful-restart** statement at the following hierarchy levels:

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

To configure in graceful restart in a particular routing instance, include the **graceful-restart** 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 **restart-duration** option sets the period of time that the router waits for a graceful restart to be completed. You can configure a time between 1 through 600 seconds. The default value is 300 seconds. At the end of the configured time period, the router performs a standard restart without recovering its state from the neighboring routers. This disrupts VPN services, but is probably necessary if the router is not functioning normally.

You can include the **restart-duration** option at either the global or routing instance level. The routing instance value overrides the global value if both are configured.

Configuring Redundant Pseudowires for Layer 2 Circuits and VPLS

A redundant pseudowire can act as a backup connection between PE routers and CE devices, maintaining Layer 2 circuit and VPLS services after certain types of failures. This feature can help improve the reliability of certain types of networks (metro for example) where a single point of failure could interrupt service for multiple customers. Redundant pseudowires cannot reduce traffic loss to zero. However, they provide a way to gracefully recover from pseudowire failures in such a way that service can be restarted within a known time limit.

NOTE: VPLS is not supported on ACX Series routers.

For an overview of how redundant pseudowires work, see "Redundant Pseudowires for Layer 2 Circuits and VPLS" on page 9.

To configure pseudowire redundancy for Layer 2 circuits and VPLS, complete the procedures in the following sections:

- Configuring Pseudowire Redundancy on the PE Router on page 44
- Configuring the Switchover Delay for the Pseudowires on page 44
- Configuring a Revert Time for the Redundant Pseudowire on page 45

Configuring Pseudowire Redundancy on the PE Router

You configure pseudowire redundancy on the PE router acting as the egress for the primary and standby pseudowires using the **backup-neighbor** statement.

To configure pseudowire redundancy on the PE router, include the **backup-neighbor** statement:

```
backup-neighbor {
    community name;
    psn-tunnel-endpoint address;
    standby;
    virtual-circuit-id number;
}
```

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

The **backup-neighbor** statement includes the following configuration options:

- **community**—Specifies the community for the backup neighbor.
- **psn-tunnel-endpoint**—Specifies the endpoint address for the packet switched network (PSN) tunnel on the remote PE router. The PSN tunnel endpoint address is the destination address for the LSP on the remote PE router.
- **standby**—Configures the pseudowire to the specified backup neighbor as the standby. When you configure this statement, traffic flows over both the active and standby pseudowires to the CE device. The CE device drops the traffic from the standby pseudowire, unless the active pseudowire fails. If the active pseudowire fails, the CE device automatically switches to the standby pseudowire.
- **virtual-circuit-id**—Uniquely identifies the primary and standby Layer 2 circuits. This option is configurable for Layer 2 circuits only.

Configuring the Switchover Delay for the Pseudowires

To configure the time the router waits before switching traffic from the failed primary pseudowire to a backup pseudowire, include the **switchover-delay** statement:

```
switchover-delay milliseconds;
```

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

Configuring a Revert Time for the Redundant Pseudowire

You can specify a revert time for redundant Layer 2 circuit and VPLS pseudowires. When you have configured redundant pseudowires for Layer 2 circuits or VPLS, traffic is switched to the backup pseudowire in the event that the primary pseudowire fails. If you configure a revert time, when the configured time expires traffic is reverted back to the primary pseudowire, assuming the primary pseudowire has been restored.

To configure a revert time for redundant pseudowires, specify the time in seconds using the **revert-time** statement:

revert-time (Protocols Layer 2 Circuits) *seconds* maximum *seconds*;

With the **maximum** option, specify a maximum reversion interval to add after the **revert-time** delay. If a revert-time delay is defined but a maximum timer is not defined, VCs are restored upon the revert-timer's expiration.

To reduce as much as possible the amount of traffic discarded, and potential data-path asymmetries observed during primary-to-backup transition periods, you can use this restoration timer. This restoration timer is activated when the backup path is performing as active, and then the primary path is restored. The goal is to avoid moving traffic back to the primary path right away, to make sure that the control plane's related tasks (such as IGP, LDP, RSVP, and internal BGP) have enough time to complete their updating cycle.

By enabling a gradual return of traffic to the primary path, you can ensure that the relatively-slow control-plane processing and updating does not have a negative impact on the restoration process.

The **maximum** option extends the revert timer's functionality to provide a jittered interval over which a certain number of circuits can be transitioned back to the primary path. By making use of this maximum value, you can define a time interval during which circuits are expected to switch over. As a consequence, circuits' effective transitions are scattered during restoration periods.

When making use of **revert-time *x* maximum *y*** statement, you can ensure that the corresponding circuit that is active is moved to the primary path within a time-slot (t1) such as that: x <= t1 <= y. In other words, by activating this statement, you can ensure the following:

- VCs stay in the backup path for at least x seconds after the primary path comes back up.
- VCs are moved back to the primary path before y seconds have elapsed.
- y maximum value = x maximum value * 2 = 1200 seconds.

The ideal values for x and y will are conditioned to internal aspects of your network. For this reason, there are no default values for these settings. If no revert-time is set, the default behavior is non-revertive. That is, circuits are not returned to the primary path upon restoration. They are kept on the backup path.

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

Related Documentation

- *Example: Configuring Pseudowire Redundancy for Mobile Backhaul Scenarios*
- *Example: Configuring H-VPLS Without VLANs*

Configuring BFD for VCCV for Layer 2 VPNs, Layer 2 Circuits, and VPLS

Bidirectional Forwarding Detection (BFD) support for virtual circuit connection verification (VCCV) allows you to configure a control channel for a pseudowire, in addition to the corresponding operations and management functions to be used over that control channel. BFD provides a low resource mechanism for the continuous monitoring of the pseudowire data path and for detecting data plane failures.

This feature provides support for asynchronous mode BFD for VCCV as described in RFC 5885, *Bidirectional Forwarding Detection (BFD) for the Pseudowire Virtual Circuit Connectivity Verification (VCCV)*. You can also use a ping operation to detect pseudowire failures. However, the processing resources required for a ping operation are greater than what is needed for BFD. In addition, BFD is capable of detecting data plane failure faster than VCCV ping. BFD for pseudowires is supported for Layer 2 circuits (LDP-based), Layer 2 VPNs (BGP-based), and VPLS (LDP-based or BGP-based).

To configure OAM and BFD for Layer 2 VPNs, include the **oam** statement and sub-statements at the **[edit routing-instances *routing-instance-name* protocols l2vpn]** hierarchy level:

```
oam {
  ping-interval ;
  bfd-liveness-detection {
    detection-time {
      threshold milliseconds;
    }
    minimum-interval milliseconds;
    minimum-receive-interval milliseconds;
    multiplier number;
    no-adaptation;
    transmit-interval {
      minimum-interval milliseconds;
      threshold milliseconds;
    }
    version bfd-protocol-version;
  }
  control-channel {
    pwe3-control-word;
    pseudowire-label-ttl-1;
    router-alert-label;
  }
}
```

For more information about how to configure BFD, see the *Junos OS Routing Protocols Library for Routing Devices*.

You can configure many of the same OAM statements for VPLS and Layer 2 circuits:

- To enable OAM for VPLS, configure the **oam** statement and substatements at the **[edit routing-instances *routing-instance-name* protocols vpls]** hierarchy level and at the **[edit routing-instances *routing-instance-name* protocols vpls neighbor *address*]** hierarchy level. The **pwe3-control-word** statement configured at the **[edit routing-instances *routing-instance-name* protocols l2vpn oam control-channel]** hierarchy level is not applicable to VPLS configurations.
- To enable OAM for Layer 2 circuits, configure the **oam** statement and substatements at the **[edit protocols l2circuit neighbor *address* interface *interface-name*]** hierarchy level. The **control-channel** statement and sub-statements configured at the **[edit routing-instances routing-instance-name protocols l2vpn oam]** hierarchy level do not apply to Layer 2 circuit configurations.

You can use the **show ldp database extensive** command to display information about the VCCV control channel and the **show bfd session extensive** command to display information about BFD for Layer 2 VPNs, Layer 2 circuits, and VPLS.

Configuring Aggregate Labels for VPNs

Aggregate labels for VPNs allow a Juniper Networks routing platform to aggregate a set of incoming labels (labels received from a peer router) into a single forwarding label that is selected from the set of incoming labels. The single forwarding label corresponds to a single next hop for that set of labels. Label aggregation reduces the number of VPN labels that the router must examine.

For a set of labels to share an aggregate forwarding label, they must belong to the same forwarding equivalence class (FEC). The labeled packets must have the same destination egress interface.

Including the **community *community-name*** statement with the **aggregate-label** statement lets you specify prefixes with a common origin community. Set by policy on the peer PE, these prefixes represent an FEC on the peer PE router.

CAUTION: If the target community is set by mistake instead of the origin community, forwarding problems at the egress PE can result. All prefixes from the peer PE will appear to be in the same FEC, resulting in a single inner label for all CE routers behind a given PE in the same VPN.

To work with route reflectors in Layer 3 VPN networks, the Juniper Networks M10i router aggregates a set of incoming labels only when the routes:

- Are received from the same peer router
- Have the same site of origin community
- Have the same next hop

The next hop requirement is important because route reflectors forward routes originated from different BGP peers to another BGP peer without changing the next hop of those routes.

To configure aggregate labels for VPNs, include the aggregate-label statement:

```
aggregate-label {
  community community-name;
}
```

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

For information about how to configure a community, see *Understanding BGP Communities and Extended Communities as Routing Policy Match Conditions*.

Related Documentation

- *Understanding BGP Communities and Extended Communities as Routing Policy Match Conditions*

Configuring Path MTU Checks for VPNs

By default, the maximum transmission unit (MTU) check for VPN routing instances is disabled on M Series routers (except the M320 router) and enabled for the M320 router and T Series routers. On M Series routers, you can configure path MTU checks on the outgoing interfaces for unicast traffic routed on VRF routing instances and on virtual-router routing instances.

When you enable an MTU check, the routing platform sends an Internet Control Message Protocol (ICMP) message when a packet traversing the routing instance exceeds the MTU size and has the **do-not-fragment** bit set. The ICMP message uses the VRF local address as its source address.

For an MTU check to work in a routing instance, you must both include the **vrf-mtu-check** statement at the **[edit chassis]** hierarchy level and assign at least one interface containing an IP address to the routing instance.

For more information about the path MTU check, see the *Junos OS Administration Library for Routing Devices*.

To configure path MTU checks, do the tasks described in the following sections:

- Enabling Path MTU Checks for a VPN Routing Instance on page 48
- Assigning an IP Address to the VPN Routing Instance on page 49

Enabling Path MTU Checks for a VPN Routing Instance

To enable path checks on the outgoing interface for unicast traffic routed on a VRF or virtual-router routing instance, include the **vrf-mtu-check** statement at the **[edit chassis]** hierarchy level:

```
[edit chassis]
vrf-mtu-check;
```

Assigning an IP Address to the VPN Routing Instance

To ensure that the path MTU check functions properly, at least one IP address must be associated with each VRF or virtual-router routing instance. If an IP address is not associated with the routing instance, ICMP reply messages cannot be sent.

Typically, the VRF or virtual-router routing instance IP address is drawn from among the IP addresses associated with interfaces configured for that routing instance. If none of the interfaces associated with a VRF or virtual-router routing instance is configured with an IP address, you need to explicitly configure a logical loopback interface with an IP address. This interface must then be associated with the routing instance. See *Configuring Logical Units on the Loopback Interface for Routing Instances in Layer 3 VPNs* for details.

Enabling Unicast Reverse-Path Forwarding Check for VPNs

IP spoofing may occur during a denial-of-service (DoS) attack. IP spoofing allows an intruder to pass IP packets to a destination as genuine traffic, when in fact the packets are not actually meant for the destination. This type of spoofing is harmful because it consumes the destination's resources.

Unicast reverse-path forwarding (RPF) check is a tool to reduce forwarding of IP packets that may be spoofing an address. A unicast RPF check performs a route table lookup on an IP packet's source address, and checks the incoming interface. The router determines whether the packet is arriving from a path that the sender would use to reach the destination. If the packet is from a valid path, the router forwards the packet to the destination address. If it is not from a valid path, the router discards the packet. Unicast RPF is supported for the IPv4 and IPv6 protocol families, as well as for the virtual private network (VPN) address family. You can also enable unicast RPF within a VPN routing instance.

To enable unicast RPF check, include the **unicast-reverse-path** statement:

```
unicast-reverse-path (active-paths | feasible-paths);
```

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

To consider only active paths during the unicast RPF check, include the **active-paths** option. To consider all feasible paths during the unicast RPF check, include the **feasible-paths** option.

For more information about how to configure the **unicast-reverse-path** statement, see *Example: Configuring Unicast Reverse-Path-Forwarding Check* and .

Related Documentation

- *Example: Configuring Unicast Reverse-Path-Forwarding Check*

CHAPTER 4

VPN Examples

Example: BGP Route Target Filtering for VPNs

BGP route target filtering is enabled by configuring the **family route-target** statement at the appropriate BGP hierarchy level. This statement enables the exchange of a new **route-target** address family, which is stored in the bgp.rtarget.0 routing table.

The following configuration illustrates how you could configure BGP route target filtering for a BGP group titled **to_vpn04**:

```
[edit]
protocols {
   bgp {
      group to_vpn04 {
         type internal;
         local-address 10.255.14.182;
         peer-as 200;
         neighbor 10.255.14.174 {
            family inet-vpn {
               unicast;
            }
            family route-target;
         }
      }
   }
}
```

The following configuration illustrates how you could configure a couple of local VPN routing and forwarding (VRF) routing instances to take advantage of the functionality provided by BGP route target filtering. Based on this configuration, BGP would automatically generate local routes corresponding to the route targets referenced in the VRF import policies (note the targets defined by the **vrf-target** statements).

```
[edit]
routing-instances {
   vpn1 {
      instance-type vrf;
      interface t1-0/1/2.0;
      vrf-target target:200:101;
      protocols {
         ospf {
            export bgp-routes;
            area 0.0.0.0 {
               interface t1-0/1/2.0;
            }
         }
      }
   }
   vpn2 {
      instance-type vrf;
      interface t1-0/1/2.1;
      vrf-target target:200:102;
      protocols {
         ospf {
            export bgp-routes;
            area 0.0.0.0 {
               interface t1-0/1/2.1;
            }
         }
      }
   }
}
```

Issue the **show route table bgp.rtarget.0** show command to verify the BGP route target filtering configuration:

```
user@host> show route table bgp.rtarget.0
 bgp.rtarget.0: 4 destinations, 6 routes (4 active, 0 holddown, 0 hidden)
 + = Active Route, - = Last Active, * = Both
200:200:101/96
                    *[RTarget/5] 00:10:00
                       Local
200:200:102/96
                    *[RTarget/5] 00:10:00
                       Local
200:200:103/96
                    *[BGP/170] 00:09:48, localpref 100, from 10.255.14.174
                       AS path: I
                     > t3-0/0/0.0
200:200:104/96
                    *[BGP/170] 00:09:48, localpref 100, from 10.255.14.174
                       AS path: I
                     > t3-0/0/0.0
```

The **show** command display format for route target prefixes is:

AS number:route target extended community/length

The first number represents the autonomous system (AS) of the router that sent this advertisement. The remainder of the display follows the Junos **show** command convention for extended communities.

The output from the **show route table bgp-rtarget.0** command displays the locally generated and remotely generated routes.

The first two entries correspond to the route targets configured for the two local VRF routing instances (**vpn1** and **vpn2**):

- **200:200:101/96**—Community **200:101** in the **vpn1** routing instance
- **200:200:102/96**—Community **200:102** in the **vpn2** routing instance

The last two entries are prefixes received from a BGP peer:

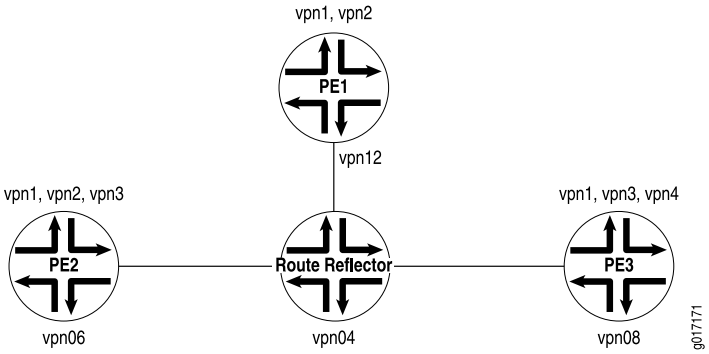
- **200:200:103/96**—Tells the local router that routes tagged with this community (**200:103**) should be advertised to peer **10.255.14.174** through **t3-0/0/0.0**
- **200:200:104/96**—Tells the local router that routes tagged with this community (**200:104**) should be advertised to peer **10.255.14.174** through **t3-0/0/0.0**

Example: Configuring BGP Route Target Filtering for VPNs

BGP route target filtering reduces the number of routers that receive VPN routes and route updates, helping to limit the amount of overhead associated with running a VPN. BGP route target filtering is most effective at reducing VPN-related administrative traffic in networks where there are many route reflectors or AS border routers that do not participate in the VPNs directly (do not act as PE routers for the CE devices).

Figure 2 on page 53 illustrates the topology for a network configured with BGP route target filtering for a group of VPNs.

Figure 2: BGP Route Target Filtering Enabled for a Group of VPNs



The following sections describe how to configure BGP route target filtering for a group of VPNs:

- Configure BGP Route Target Filtering on Router PE1 on page 54
- Configure BGP Route Target Filtering on Router PE2 on page 55
- Configure BGP Route Target Filtering on the Route Reflector on page 58
- Configure BGP Route Target Filtering on Router PE3 on page 59

Configure BGP Route Target Filtering on Router PE1

This section describes how to enable BGP route target filtering on Router PE1 for this example.

Configure the routing options on router PE1 as follows:

```
[edit]
routing-options {
   route-distinguisher-id 10.255.14.182;
   autonomous-system 200;
}
```

Configure the BGP protocol on Router PE1 as follows:

```
[edit]
protocols {
   bgp {
      group to_VPN_D {
         type internal;
         local-address 10.255.14.182;
         peer-as 200;
         neighbor 10.255.14.174 {
            family inet-vpn {
               unicast;
            }
            family route-target;
         }
      }
   }
}
```

Configure the **vpn1** routing instance as follows:

```
[edit]
routing-instances {
   vpn1 {
      instance-type vrf;
      interface t1-0/1/2.0;
      vrf-target target:200:101;
      protocols {
         ospf {
            export bgp-routes;
            area 0.0.0.0 {
               interface t1-0/1/2.0;
            }
         }
      }
   }
}
```

Configure the **vpn2** routing instance on Router PE1 as follows:

```
[edit]
routing-instances {
   vpn2 {
```

```
    instance-type vrf;
    interface t1-0/1/2.1;
    vrf-target target:200:102;
    protocols {
      ospf {
        export bgp-routes;
        area 0.0.0.0 {
          interface t1-0/1/2.1;
        }
      }
    }
  }
}
```

Once you have implemented this configuration, you should see the following when you issue a **show route table bgp.rtarget.0** command:

```
user@host> show route table bgp.rtarget.0
bgp.rtarget.0: 4 destinations, 6 routes (4 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

200:200:101/96
                   *[RTarget/5] 00:27:42
                      Local
                    [BGP/170] 00:27:30, localpref 100, from
10.255.14.174
                      AS path: I
                    > via t3-0/0/0.0
200:200:102/96
                   *[RTarget/5] 00:27:42
                      Local
                    [BGP/170] 00:27:30, localpref 100, from
10.255.14.174
                      AS path: I
                    > via t3-0/0/0.0
200:200:103/96
                   *[BGP/170] 00:27:30, localpref 100, from
10.255.14.174
                      AS path: I
                    > via t3-0/0/0.0
200:200:104/96
                   *[BGP/170] 00:27:30, localpref 100, from
10.255.14.174
                      AS path: I
                    > via t3-0/0/0.0
```

Configure BGP Route Target Filtering on Router PE2

This section describes how to enable BGP route target filtering on Router PE2 for this example.

Configure the routing options on Router PE2 as follows:

```
[edit]
routing-options {
  route-distinguisher-id 10.255.14.176;
  autonomous-system 200;
}
```

Configure the BGP protocol on Router PE2 as follows:

```
[edit]
protocols {
    bgp {
        group to_vpn04 {
            type internal;
            local-address 10.255.14.176;
            peer-as 200;
            neighbor 10.255.14.174 {
                family inet-vpn {
                    unicast;
                }
                family route-target;
            }
        }
    }
}
```

Configure the **vpn1** routing instance on Router PE2 as follows:

```
[edit]
routing-instances {
    vpn1 {
        instance-type vrf;
        interface t3-0/0/0.0;
        vrf-target target:200:101;
        protocols {
            bgp {
                group vpn1 {
                    type external;
                    peer-as 101;
                    as-override;
                    neighbor 10.49.11.2;
                }
            }
        }
    }
}
```

Configure the **vpn2** routing instance on Router PE2 as follows:

```
[edit]
routing-instances {
    vpn2 {
        instance-type vrf;
        interface t3-0/0/0.1;
        vrf-target target:200:102;
        protocols {
            bgp {
                group vpn2 {
                    type external;
                    peer-as 102;
                    as-override;
                    neighbor 10.49.21.2;
                }
            }
```

```
    }
  }
}
```

Configure the **vpn3** routing instance on Router PE2 as follows:

```
[edit]
routing-instances {
  vpn3 {
    instance-type vrf;
    interface t3-0/0/0.2;
    vrf-import vpn3-import;
    vrf-export vpn3-export;
    protocols {
      bgp {
        group vpn3 {
          type external;
          peer-as 103;
          as-override;
          neighbor 10.49.31.2;
        }
      }
    }
  }
}
```

Once you have configured router PE2 in this manner, you should see the following when you issue the **show route table bgp.rtarget.0** command:

```
user@host> show route table bgp.rtarget.0
 bgp.rtarget.0: 4 destinations, 7 routes (4 active, 0 holddown, 0 hidden)
 + = Active Route, - = Last Active, * = Both

 200:200:101/96
                    *[RTarget/5] 00:28:15
                       Local
                     [BGP/170] 00:28:03, localpref 100, from
 10.255.14.174
                       AS path: I
                     > via t1-0/1/0.0
 200:200:102/96
                    *[RTarget/5] 00:28:15
                       Local
                     [BGP/170] 00:28:03, localpref 100, from
 10.255.14.174
                       AS path: I
                     > via t1-0/1/0.0
 200:200:103/96
                    *[RTarget/5] 00:28:15
                       Local
                     [BGP/170] 00:28:03, localpref 100, from
 10.255.14.174
                       AS path: I
                     > via t1-0/1/0.0
 200:200:104/96
                    *[BGP/170] 00:28:03, localpref 100, from
 10.255.14.174
                       AS path: I
                     > via t1-0/1/0.0
```

Configure BGP Route Target Filtering on the Route Reflector

This section illustrates how to enable BGP route target filtering on the route reflector for this example.

Configure the routing options on the route reflector as follows:

```
[edit]
routing-options {
  route-distinguisher-id 10.255.14.174;
  autonomous-system 200;
}
```

Configure the BGP protocol on the route reflector as follows:

```
[edit]
protocols {
  bgp {
    group rr-group {
      type internal;
      local-address 10.255.14.174;
      cluster 10.255.14.174;
      peer-as 200;
      neighbor 10.255.14.182 {
        description to_PE1_vpn12;
        family inet-vpn {
          unicast;
        }
        family route-target;
      }
      neighbor 10.255.14.176 {
        description to_PE2_vpn06;
        family inet-vpn {
          unicast;
        }
        family route-target;
      }
      neighbor 10.255.14.178 {
        description to_PE3_vpn08;
        family inet-vpn {
          unicast;
        }
        family route-target;
      }
    }
  }
}
```

Once you have configured the route reflector in this manner, you should see the following when you issue the **show route table bgp.rtarget.0** command:

```
user@host> show route table bgp.rtarget.0
 bgp.rtarget.0: 4 destinations, 8 routes (4 active, 0 holddown, 0 hidden)
 + = Active Route, - = Last Active, * = Both

 200:200:101/96
```

```
                    *[BGP/170] 00:29:03, localpref 100, from
10.255.14.176
                      AS path: I
                    > via t1-0/2/0.0
                     [BGP/170] 00:29:03, localpref 100, from
10.255.14.178
                      AS path: I
                    > via t3-0/1/1.0
                     [BGP/170] 00:29:03, localpref 100, from
10.255.14.182
                      AS path: I
                    > via t3-0/1/3.0
200:200:102/96
                    *[BGP/170] 00:29:03, localpref 100, from
10.255.14.176
                      AS path: I
                    > via t1-0/2/0.0
                     [BGP/170] 00:29:03, localpref 100, from
10.255.14.182
                      AS path: I
                    > via t3-0/1/3.0
200:200:103/96
                    *[BGP/170] 00:29:03, localpref 100, from
10.255.14.176
                      AS path: I
                    > via t1-0/2/0.0
                     [BGP/170] 00:29:03, localpref 100, from
10.255.14.178
                      AS path: I
                    > via t3-0/1/1.0
200:200:104/96
                    *[BGP/170] 00:29:03, localpref 100, from
10.255.14.178
                      AS path: I
                    > via t3-0/1/1.0
```

Configure BGP Route Target Filtering on Router PE3

The following section describes how to enable BGP route target filtering on Router PE3 for this example.

Configure the routing options on Router PE3 as follows:

```
[edit]
routing-options {
   route-distinguisher-id 10.255.14.178;
   autonomous-system 200;
}
```

Configure the BGP protocol on Router PE3 as follows:

```
[edit]
protocols {
   bgp {
      group to_vpn04 {
         type internal;
         local-address 10.255.14.178;
         peer-as 200;
         neighbor 10.255.14.174 {
```

```
            family inet-vpn {
              unicast;
            }
            family route-target;
          }
        }
      }
    }
```

Configure the **vpn1** routing instance on Router PE3 as follows:

```
[edit]
routing-instances {
  vpn1 {
    instance-type vrf;
    interface t3-0/0/0.0;
    vrf-target target:200:101;
    protocols {
      rip {
        group vpn1 {
          export bgp-routes;
          neighbor t3-0/0/0.0;
        }
      }
    }
  }
}
```

Configure the **vpn3** routing instance on Router PE3 as follows:

```
[edit]
routing-instances {
  vpn3 {
    instance-type vrf;
    interface t3-0/0/0.1;
    vrf-target target:200:103;
    protocols {
      rip {
        group vpn3 {
          export bgp-routes;
          neighbor t3-0/0/0.1;
        }
      }
    }
  }
}
```

Configure the **vpn4** routing instance on Router PE3 as follows:

```
[edit]
routing-instances {
  vpn4 {
    instance-type vrf;
    interface t3-0/0/0.2;
    vrf-target target:200:104;
    protocols {
      rip {
```



```
            group vpn4 {
              export bgp-routes;
              neighbor t3-0/0/0.2;
            }
          }
        }
      }
    }
```

Once you have configured Router PE3 in this manner, you should see the following when you issue the **show route table bgp.rtarget.0** command:

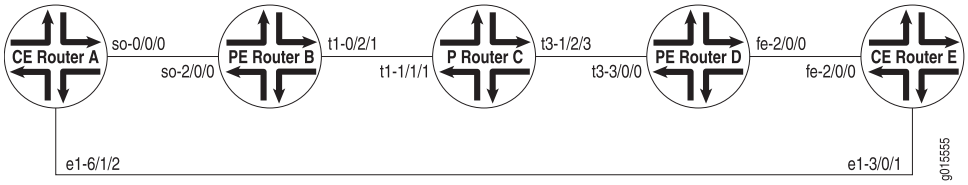
```
user@host> show route table bgp.rtarget.0
bgp.rtarget.0: 4 destinations, 7 routes (4 active, 0 holddown, 0 hidden)
 + = Active Route, - = Last Active, * = Both

 200:200:101/96
                    *[RTarget/5] 00:29:42
                       Local
                     [BGP/170] 00:29:30, localpref 100, from
 10.255.14.174
                       AS path: I
                     > via t3-0/0/1.0
 200:200:102/96
                    *[BGP/170] 00:29:29, localpref 100, from
 10.255.14.174
                       AS path: I
                     > via t3-0/0/1.0
 200:200:103/96
                    *[RTarget/5] 00:29:42
                       Local
                     [BGP/170] 00:29:30, localpref 100, from
 10.255.14.174
                       AS path: I
                     > via t3-0/0/1.0
 200:200:104/96
                    *[RTarget/5] 00:29:42
                       Local
                     [BGP/170] 00:29:30, localpref 100, from
 10.255.14.174
                       AS path: I
                     > via t3-0/0/1.0
```

Route Origin for VPNs

You can use route origin to prevent routes learned from one customer edge (CE) router marked with origin community from being advertised back to it from another CE router in the same AS.

In the example, the route origin is used to prevent routes learned from CE Router A that are marked with origin community from being advertised back to CE Router E by AS 200. The example topology is shown in Figure 3 on page 62.

Figure 3: Network Topology of Site of Origin Example



In this topology, CE Router A and CE Router E are in the same AS (AS200). They use EBGP to exchange routes with their respective provider edge (PE) routers, PE Router B and PE Router D. The two CE routers have a back connection.

The following sections describe how to configure the route origin for a group of VPNs:

- Configuring the Site of Origin Community on CE Router A on page 62
- Configuring the Community on CE Router A on page 62
- Applying the Policy Statement on CE Router A on page 63
- Configuring the Policy on PE Router D on page 63
- Configuring the Community on PE Router D on page 64
- Applying the Policy on PE Router D on page 64

Configuring the Site of Origin Community on CE Router A

The following section describes how to configure CE Router A to advertise routes with a site of origin community to PE Router B for this example.

NOTE: In this example, direct routes are configured to be advertised, but any route can be configured.

Configure a policy to advertise routes with **my-soo** community on CE Router A as follows:

```
[edit]
policy-options {
   policy-statement export-to-my-isp {
      term a {
         from {
            protocol direct;
         }
         then {
            community add my-soo;
            accept;
         }
      }
   }
}
```

Configuring the Community on CE Router A

Configure the **my-soo** community on CE Router A as follows:

```
[edit]
```

```
policy-options {
  community my-soo {
    members origin:100:1;
  }
}
```

Applying the Policy Statement on CE Router A

Apply the export-to-my-isp policy statement as an export policy to the EBGP peering on the CE Router A as follows:

```
[edit]
protocols {
  bgp {
    group my_isp {
      export export-to-my-isp;
    }
  }
}
```

When you issue the **show route receive-protocol bgp detail** command, you should see the following routes originated from PE Router B with **my-soo** community:

```
user@host> show route receive-protocol bgp 10.12.99.2 detail
inet.0: 16 destinations, 16 routes (15 active, 0 holddown, 1 hidden)
inet.3: 2 destinations, 2 routes (2 active, 0 holddown, 0 hidden)
vpn_blue.inet.0: 8 destinations, 10 routes (8 active, 0 holddown, 0 hidden)
* 10.12.33.0/30 (2 entries, 1 announced)
     Nexthop: 10.12.99.2
     AS path: 100 I
     Communities: origin:100:1
  10.12.99.0/30 (2 entries, 1 announced)
     Nexthop: 10.12.99.2
     AS path: 100 I
     Communities: origin:100:1
* 10.255.71.177/32 (1 entry, 1 announced)
     Nexthop: 10.12.99.2
     AS path: 100 I
     Communities: origin:100:1
* 192.168.64.0/21 (1 entry, 1 announced)
     Nexthop: 10.12.99.2
     AS path: 100 I
     Communities: origin:100:1
iso.0: 1 destinations, 1 routes (1 active, 0 holddown, 0 hidden)
mpls.0: 8 destinations, 8 routes (8 active, 0 holddown, 0 hidden)
bgp.l3vpn.0: 4 destinations, 4 routes (4 active, 0 holddown, 0 hidden)
inet6.0: 2 destinations, 2 routes (2 active, 0 holddown, 0 hidden)
__juniper_private1__.inet6.0: 1 destinations, 1 routes (1 active, 0 holddown, 0
hidden)
```

Configuring the Policy on PE Router D

Configure a policy on PE Router D that prevents routes with **my-soo** community tagged by CE Router A from being advertised to CE Router E as follows:

```
[edit]
policy-options {
  policy-statement soo-ce1-policy {
```

```
      term a {
        from {
          community my-soo;
          then {
            reject;
          }
        }
      }
    }
  }
```

Configuring the Community on PE Router D

Configure the community on PE Router D as follows:

```
[edit]
policy-options {
  community my-soo {
    members origin:100:1;
  }
}
```

Applying the Policy on PE Router D

To prevent routes learned from CE Router A from being advertised to CE Router E (the two routers can communicate these routes directly), apply the **soo-ce1-policy** policy statement as an export policy to the PE Router D and CE Router E EBGP session **vpn_blue**.

View the EBGP session on PE Router D using the **show routing-instances** command.

```
user@host# show routing-instances
vpn_blue {
    instance-type vrf;
    interface fe-2/0/0.0;
    vrf-target target:100:200;
    protocols {
        bgp {
            group ce2 {
                advertise-peer-as;
                peer-as 100;
                neighbor 10.12.99.6;
            }
        }
    }
}
```

Apply the **soo-ce1-policy** policy statement as an export policy to the PE Router D and CE Router E EBGP session **vpn_blue** as follows:

```
[edit routing-instances]
vpn_blue {
  protocols {
    bgp {
      group ce2{
        export soo-ce1-policy;
      }
    }
  }
```

}

Example: Configuring Proxy BGP Route Target Filtering

Understanding Proxy BGP Route Target Filtering

BGP route target filtering (also known as route target constrain, or RTC) allows you to distribute VPN routes to only the devices that need them. In VPN networks without BGP route target filtering configured, BGP distributes all VPN routes to all VPN peer devices, which can strain network resources. The route target filtering feature was introduced to reduce the number of devices receiving VPN routes and VPN routing updates, thereby limiting the amount of overhead associated with running a VPN. The Junos OS implementation for BGP route target filtering is based on RFC 4684, *Constrained Route Distribution for Border Gateway Protocol/MultiProtocol Label Switching (BGP/MPLS) Internet Protocol (IP) Virtual Private Networks (VPNs)*.

What if you have a network environment where route target filtering is not widely deployed, or what if some devices do not support route target filtering? For example, you might have a BGP speaker with route target filtering enabled that is peered with a BGP speaker that does not support or have route target filtering configured. In this case, the BGP speaker with route target filtering configured must advertise default route target membership (RT membership) on behalf of its peer. The route target filtering resource savings are unrealized because the device supporting the filtering must now send all VPN routes to the device that does not support the filter. Proxy BGP route target filtering (or Proxy RTC) permits the generation of RT membership for devices that do not support route target filtering. This eases the deployment of route target filtering in networks where it is incompletely deployed or not fully supported.

Proxy BGP route target filtering allows you to distribute proxy RT membership advertisements created from the received BGP VPN routes to other devices in the network that need them. These are known as proxy advertisements because the device creates the RT membership on behalf of its peers without the route target filtering functionality. Proxy BGP route target filtering uses BGP route target extended communities that are exported to a specific BGP speaker to generate the route targets. Generated proxy RTC routes are stored in the bgp.rtarget.0 routing table.

You can also configure a policy to control which VPN routes are used to generate the proxy RTC routes. This can help control which RT membership is generated by the proxying device. In addition, you can configure a policy to reduce the memory overhead associated with proxy RTC. Proxy RTC only uses additional memory on a per-VPN route basis when it is permitted by a policy to be used for generating RT membership.

Example: Configuring Proxy BGP Route Target Filtering

This example shows how to configure proxy BGP route target filtering (also known as proxy route target constrain, or proxy RTC).

Requirements

This example uses the following hardware and software components:

- Four Juniper Networks devices that can be a combination of M Series, MX Series, or T Series routers.
- Junos OS Release 12.2 or later on one or more devices configured for proxy BGP route filtering. In this example, you explicitly configure proxy BGP route filtering on the route reflectors.

Before configuring proxy BGP route target filtering, make sure that you are familiar with and understand the following concepts:

- *Layer 2 VPNs*
- *Layer 3 VPNs*
- *Route distinguishers*
- Route targets on page 32
- BGP route target filtering on page 38
- *BGP extended communities*

Overview

Route target filtering decreases the number of devices in a network that receive VPN routes that are not needed. Proxy BGP route target filtering allows networks to take advantage of route target filtering in locations where the feature is not currently supported. By configuring this feature, you can realize many of the same network resource savings that are available to you if your network fully supported BGP route target filtering.

To configure proxy BGP route target filtering, you include the **family route-target proxy-generate** statement on the devices that will distribute proxy route target membership (RT membership) advertisements for the devices that do not support BGP route target filtering. The proxy BGP route target filtering routes are then stored in the bgp.rtarget.0 routing table.

Proxy BGP route target filtering is intended to create RT membership advertisements for devices that do not support the BGP route target filtering feature. If the **proxy-generate** statement is present, but the route target family is negotiated with the BGP peer, the

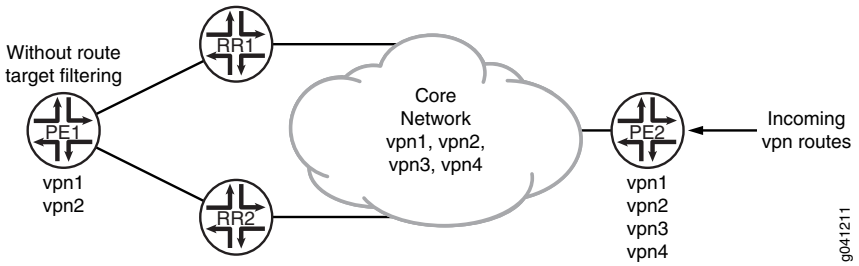
proxy-generate functionality is disabled. This allows simplified configuration of BGP peer groups where a portion of the peers in the group support route target filtering but others do not. In such an example case, the **family route-target proxy-generate** statement might be part of the BGP peer group configuration.

NOTE: When deploying proxy BGP route target filtering in your network, the **advertise-default** statement for BGP route target filtering causes the device to advertise the default route target route (0:0:0/0) and suppress all routes that are more specific. If you have proxy BGP route target filtering configured on one device and one or more peers have the **advertise-default** statement configured as part of their BGP route target filtering configuration, the advertise-default configuration is ignored.

Topology Diagram

Figure 4 on page 67 shows the topology used in this example.

Figure 4: Proxy BGP Route Target Filtering Topology



In this example, BGP route target filtering is configured on the route reflectors (Device RR1 and Device RR2) and the provider edge (PE) Device PE2, but the other PE, Device PE1, does not support the BGP route target filtering functionality. Device PE2 has four VPNs configured (vpn1, vpn2, vpn3, and vpn4). Device PE1 has two VPNs configured (vpn1 and vpn2), so this device is only interested in receiving route updates for vpn1 and vpn2. Currently, this is impossible because both route reflectors (Device RR1 and Device RR2) learn and share information about all of the incoming VPN routes (vpn1 through vpn4) with Device PE1. In the sample topology, all devices participate in autonomous system (AS) 200, OSPF is the configured interior gateway protocol (IGP), and LDP is the signaling protocol used by the VPNs. In this example, we use static routes in the VPN routing and forwarding (VRF) instances to generate VPN routes. This is done in place of using a PE to customer edge (CE) protocol such as OSPF or BGP.

To minimize the number of VPN route updates being processed by Device PE1, you include the **family route-target proxy-generate** statement to configure proxy BGP route target filtering on each route reflector. Each route reflector has a peering session with Device PE1 and supports route target filtering to the core. However, Device PE1 does not support route target filtering, so the network resource savings are unrealized by Device PE1 since it receives all of the VPN updates. By configuring proxy BGP route target filtering on the peering sessions facing Device PE1, you limit the number of VPN updates processed by

Device PE1, and the route reflectors generate the proxy BGP route target routes for Device PE1 throughout the network.

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 PE1

```
set interfaces ge-1/0/0 unit 0 description PE1-to-RR1
set interfaces ge-1/0/0 unit 0 family inet address 10.49.0.1/30
set interfaces ge-1/0/0 unit 0 family mpls
set interfaces ge-1/0/1 unit 0 description PE1-to-RR2
set interfaces ge-1/0/1 unit 0 family inet address 10.49.10.1/30
set interfaces ge-1/0/1 unit 0 family mpls
set protocols ldp interface ge-1/0/0
set protocols ldp interface ge-1/0/1
set protocols bgp group internal type internal
set protocols bgp group internal local-address 10.255.163.58
set protocols bgp group internal neighbor 10.255.165.220 family inet-vpn unicast
set protocols bgp group internal neighbor 10.255.165.28 family inet-vpn unicast
set protocols ospf area 0.0.0.0 interface ge-1/0/0
set protocols ospf area 0.0.0.0 interface ge-1/0/1
set protocols ospf area 0.0.0.0 interface lo0.0 passive
set routing-options route-distinguisher-id 10.255.163.58
set routing-options autonomous-system 200
set routing-instances vpn1 instance-type vrf
set routing-instances vpn1 vrf-target target:200:100
set routing-instances vpn1 routing-options static route 223.1.1.2/32 discard
set routing-instances vpn2 instance-type vrf
set routing-instances vpn2 vrf-target target:200:101
set routing-instances vpn2 routing-options static route 223.2.2.2/32 discard
```

Device RR1

```
set interfaces ge-1/0/0 unit 0 description RR1-to-PE1
set interfaces ge-1/0/0 unit 0 family inet address 10.49.0.2/30
set interfaces ge-1/0/0 unit 0 family mpls
set interfaces ge-1/0/1 unit 0 description RR1-to-PE2
set interfaces ge-1/0/1 unit 0 family inet address 10.50.0.2/30
set interfaces ge-1/0/1 unit 0 family mpls
set protocols ldp interface ge-1/0/0
set protocols ldp interface ge-1/0/1
set protocols bgp group internal type internal
set protocols bgp group internal local-address 110.255.165.220
set protocols bgp group internal cluster 1.1.1.1
set protocols bgp group internal neighbor 10.255.163.58 description vpn1-to-pe1 family
    inet-vpn unicast
set protocols bgp group internal neighbor 10.255.163.58 family route-target proxy-generate
```

```
set protocols bgp group internal neighbor 10.255.168.42 description vpn1-to-pe2 family
  inet-vpn unicast
set protocols bgp group internal neighbor 10.255.168.42 family route-target
set protocols ospf area 0.0.0.0 interface ge-1/0/0
set protocols ospf area 0.0.0.0 interface ge-1/0/1
set protocols ospf area 0.0.0.0 interface lo0.0 passive
set routing-options route-distinguisher-id 10.255.165.220
set routing-options autonomous-system 200
```

Device RR2

```
set interfaces ge-1/0/0 unit 0 description RR2-to-PE1
set interfaces ge-1/0/0 unit 0 family inet address 10.49.10.2/30
set interfaces ge-1/0/0 unit 0 family mpls
set interfaces ge-1/0/1 unit 0 description RR2-to-PE2
set interfaces ge-1/0/1 unit 0 family inet address 10.50.10.2/30
set interfaces ge-1/0/1 unit 0 family mpls
set protocols ldp interface ge-1/0/0
set protocols ldp interface ge-1/0/1
set protocols bgp group internal type internal
set protocols bgp group internal local-address 10.255.165.28
set protocols bgp group internal cluster 1.1.1.1
set protocols bgp group internal neighbor 10.255.163.58 description vpn2-to-pe1 family
  inet-vpn unicast
set protocols bgp group internal neighbor 10.255.163.58 family route-target proxy-generate
set protocols bgp group internal neighbor 10.255.168.42 description vpn2-to-pe2 family
  inet-vpn unicast
set protocols bgp group internal neighbor 10.255.168.42 family route-target
set protocols ospf area 0.0.0.0 interface ge-1/0/0
set protocols ospf area 0.0.0.0 interface ge-1/0/1
set protocols ospf area 0.0.0.0 interface lo0.0 passive
set routing-options route-distinguisher-id 10.255.165.28
set routing-options autonomous-system 200
```

Device PE2

```
set interfaces ge-1/0/0 unit 0 description PE2-to-RR1
set interfaces ge-1/0/0 unit 0 family inet address 10.50.0.1/30
set interfaces ge-1/0/0 unit 0 family mpls
set interfaces ge-1/0/1 unit 0 description PE2-to-RR2
set interfaces ge-1/0/1 unit 0 family inet address 10.50.10.1/30
set interfaces ge-1/0/1 unit 0 family mpls
set protocols ldp interface ge-1/0/0
set protocols ldp interface ge-1/0/1
set protocols bgp group internal type internal
set protocols bgp group internal local-address 10.255.168.42
set protocols bgp group internal family inet-vpn unicast
set protocols bgp group internal family route-target
set protocols bgp group internal neighbor 10.255.165.220
set protocols bgp group internal neighbor 10.255.165.28
set protocols ospf area 0.0.0.0 interface ge-1/0/0
set protocols ospf area 0.0.0.0 interface ge-1/0/1
set protocols ospf area 0.0.0.0 interface lo0.0 passive
set routing-options route-distinguisher-id 10.255.168.42
set routing-options autonomous-system 200
set routing-instances vpn1 instance-type vrf
set routing-instances vpn1 vrf-target target:200:100
set routing-instances vpn1 routing-options static route 223.1.1.2/32 discard
set routing-instances vpn2 instance-type vrf
```

```
set routing-instances vpn2 vrf-target target:200:101
set routing-instances vpn2 routing-options static route 223.2.2.2/32 discard
set routing-instances vpn3 instance-type vrf
set routing-instances vpn3 vrf-target target:200:103
set routing-instances vpn3 routing-options static route 223.3.3.3/32 discard
set routing-instances vpn4 instance-type vrf
set routing-instances vpn4 vrf-target target:200:104
set routing-instances vpn4 routing-options static route 223.4.4.4/32 discard
```

Configuring Device PE1

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 *CLI User Guide*.

To configure Device PE1:

1. Configure the interfaces.

```
[edit interfaces]
user@PE1# set ge-1/0/0 unit 0 description PE1-to-RR1
user@PE1# set ge-1/0/0 unit 0 family inet address 10.49.0.1/30
user@PE1# set ge-1/0/0 unit 0 family mpls

user@PE1# set ge-1/0/1 unit 0 description PE1-to-RR2
user@PE1# set ge-1/0/1 unit 0 family inet address 10.49.10.1/30
user@PE1# set ge-1/0/1 unit 0 family mpls
```

2. Configure the route distinguisher and the AS number.

```
[edit routing-options]
user@PE1# set route-distinguisher-id 10.255.163.58
user@PE1# set autonomous-system 200
```

3. Configure LDP as the signaling protocol used by the VPN.

```
[edit protocols ldp]
user@PE1# set interface ge-1/0/0
user@PE1# set interface ge-1/0/1
```

4. Configure BGP.

```
[edit protocols bgp group internal]
user@PE1# set type internal
user@PE1# set local-address 10.255.163.58
user@PE1# set neighbor 10.255.165.220 family inet-vpn unicast
user@PE1# set neighbor 10.255.165.28 family inet-vpn unicast
```

5. Configure OSPF.

```
[edit protocols ospf area 0.0.0.0]
user@PE1# set interface ge-1/0/0
user@PE1# set interface ge-1/0/1
user@PE1# set interface lo0.0 passive
```

6. Configure the VPN routing instances.

```
[edit routing-instances vpn1]
user@PE1# set instance-type vrf
```

```
user@PE1# set vrf-target target:200:100
user@PE1# set routing-options static route 223.1.1.2/32 discard

[edit routing-instances vpn2]
user@PE1# set instance-type vrf
user@PE1# set vrf-target target:200:101
user@PE1# set routing-options static route 223.2.2.2/32 discard
```

7. If you are done configuring the device, commit the configuration.

```
[edit]
user@PE1# commit
```

Results From configuration mode, confirm your configuration by entering the **show interfaces**, **show protocols**, **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@PE1# show interfaces
ge-1/0/0 {
  unit 0 {
    description PE1-to-RR1;
    family inet {
      address 10.49.0.1/30;
    }
    family mpls;
  }
}
ge-1/0/1 {
  unit 0 {
    description PE1-to-RR2;
    family inet {
      address 10.49.10.1/30;
    }
    family mpls;
  }
}
```

```
user@PE1# show protocols
bgp {
  group internal {
    type internal;
    local-address 10.255.163.58;
    neighbor 10.255.165.220 {
      family inet-vpn {
        unicast;
      }
    }
    neighbor 10.255.165.28 {
      family inet-vpn {
        unicast;
      }
    }
  }
}
ospf {
```



```
    area 0.0.0.0 {
      interface ge-1/0/0.0;
      interface ge-1/0/1.0;
      interface lo0.0 {
        passive;
      }
    }
  }
  ldp {
    interface ge-1/0/0.0;
    interface ge-1/0/1.0;
  }
```

```
user@PE1# show routing-options
route-distinguisher-id 10.255.14.182;
autonomous-system 200;
```

```
user@PE1# show routing-instances
vpn1 {
  instance-type vrf;
  vrf-target target:200:100;
  routing-options {
    static {
      route 223.1.1.2/32 discard;
    }
  }
}
vpn2 {
  instance-type vrf;
  vrf-target target:200:101;
  routing-options {
    static {
      route 223.2.2.2/32 discard;
    }
  }
}
```

Configuring Device RR1

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 *CLI User Guide*.

To configure Device RR1:

1. Configure the interfaces.

```
[edit interfaces]
user@RR1# set ge-1/0/0 unit 0 description RR1-to-PE1
user@RR1# set ge-1/0/0 unit 0 family inet address 10.49.0.2/30
user@RR1# set ge-1/0/0 unit 0 family mpls


user@RR1# set ge-1/0/1 unit 0 description RR1-to-PE2
user@RR1# set ge-1/0/1 unit 0 family inet address 10.50.0.2/30
user@RR1# set ge-1/0/1 unit 0 family mpls
```

2. Configure the route distinguisher and the AS number.

```
[edit routing-options]
user@RR1# set route-distinguisher-id 10.255.165.220
user@RR1# set autonomous-system 200
```

3. Configure LDP as the signaling protocol used by the VPN.

```
[edit protocols ldp]
user@RR1# set interface ge-1/0/0
user@RR1# set interface ge-1/0/1
```

4. Configure BGP.

```
[edit protocols bgp group internal]
user@RR1# set type internal
user@RR1# set local-address 10.255.165.220
user@RR1# set cluster 1.1.1.1
user@RR1# set neighbor 10.255.163.58 description vpn1-to-pe1 family inet-vpn
  unicast
user@RR1# set neighbor 10.255.168.42 description vpn1-to-pe2 family inet-vpn
  unicast
```

5. Configure BGP route target filtering on the peering session with Device PE2.

```
[edit protocols bgp group internal]
user@RR1# set neighbor 10.255.168.42 family route-target
```

6. Configure proxy BGP route target filtering on the peering session with Device PE1.

```
[edit protocols bgp group internal]
user@RR1# set neighbor 10.255.163.58 family route-target proxy-generate
```

7. Configure OSPF.

```
[edit protocols ospf area 0.0.0.0]
user@RR1# set interface ge-1/0/0
user@RR1# set interface ge-1/0/1
user@RR1# set interface lo0.0 passive
```

8. If you are done configuring the device, commit the configuration.

```
[edit]
user@RR1# commit
```

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

```
user@RR1# show interfaces
ge-1/0/0 {
   unit 0 {
      description RR1-to-PE1;
      family inet {
         address 10.49.0.2/30;
      }
      family mpls;
   }
}
ge-1/0/1 {
   unit 0 {
      description RR1-to-PE2;
```

```
      family inet {
        address 10.50.0.2/30;
      }
      family mpls;
    }
}
user@RR1# show protocols
bgp {
  group internal {
    type internal;
    local-address 110.255.165.220;
    cluster 1.1.1.1;
    neighbor 10.255.163.58 {
      description vpn1-to-pe1;
      family inet-vpn {
        unicast;
      }
      family route-target {
        proxy-generate;
      }
    }
    neighbor 10.255.168.42 {
      description vpn1-to-pe2;
      family inet-vpn {
        unicast;
      }
      family route-target;
    }
  }
}
ospf {
  area 0.0.0.0 {
    interface ge-1/0/0.0;
    interface ge-1/0/1.0;
    interface lo0.0 {
      passive;
    }
  }
}
ldp {
  interface ge-1/0/0.0;
  interface ge-1/0/1.0;
}
user@RR1# show routing-options
route-distinguisher-id 10.255.165.220;
autonomous-system 200;
```

Configuring Device RR2

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 *CLI User Guide*.

To configure Device RR2:

1. Configure the interfaces.

```
[edit interfaces]
user@RR2# set ge-1/0/0 unit 0 description RR2-to-PE1
user@RR2# set ge-1/0/0 unit 0 family inet address 10.49.10.2/30
user@RR2# set ge-1/0/0 unit 0 family mpls


user@RR2# set ge-1/0/1 unit 0 description RR2-to-PE2
user@RR2# set ge-1/0/1 unit 0 family inet address 10.50.10.2/30
user@RR2# set ge-1/0/1 unit 0 family mpls
```

2. Configure the route distinguisher and the AS number.

```
[edit routing-options]
user@RR2# set route-distinguisher-id 10.255.165.28
user@RR2# set autonomous-system 200
```

3. Configure LDP as the signaling protocol used by the VPN.

```
[edit protocols ldp]
user@RR2# set interface ge-1/0/0
user@RR2# set interface ge-1/0/1
```

4. Configure BGP.

```
[edit protocols bgp group internal]
user@RR2# set type internal
user@RR2# set local-address 10.255.165.28
user@RR2# set cluster 1.1.1.1
user@RR2# set neighbor 10.255.163.58 description vpn2-to-pe1 family inet-vpn
  unicast
user@RR2# set neighbor 10.255.168.42 description vpn2-to-pe2 family inet-vpn
  unicast
```

5. Configure BGP route target filtering on the peering session with Device PE2.

```
[edit protocols bgp group internal]
user@RR2# set neighbor 10.255.168.42 family route-target
```

6. Configure proxy BGP route target filtering on the peering session with Device PE1.

```
[edit protocols bgp group internal]
user@RR2# set neighbor 10.255.163.58 family route-target proxy-generate
```

7. Configure OSPF.

```
[edit protocols ospf area 0.0.0.0]
user@RR2# set interface ge-1/0/0
user@RR2# set interface ge-1/0/1
user@RR2# set interface lo0.0 passive
```

8. If you are done configuring the device, commit the configuration.

```
[edit]
user@RR2# commit
```

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

```
user@RR2# show interfaces
ge-1/0/0 {
  unit 0 {
    description RR2-to-PE1;
    family inet {
      address 10.49.10.2/30;
    }
    family mpls;
  }
}
ge-1/0/1 {
  unit 0 {
    description RR2-to-PE2;
    family inet {
      address 10.50.10.2/30;
    }
    family mpls;
  }
}
```

```
user@RR2# show protocols
bgp {
  group internal {
    local-address 10.255.165.28;
    cluster 1.1.1.1;
    neighbor 10.255.163.58 {
      description vpn2-to-pe1;
      family inet-vpn {
        unicast;
      }
      family route-target {
        proxy-generate;
      }
    }
    neighbor 10.255.168.42 {
      description vpn2-to-pe2;
      family inet-vpn {
        unicast;
      }
      family route-target;
    }
  }
}
ospf {
  area 0.0.0.0 {
    interface ge-1/0/0.0;
    interface ge-1/0/1.0;
    interface lo0.0 {
```



```
        passive;
      }
    }
  }
  ldp {
    interface ge-1/0/0.0;
    interface ge-1/0/1.0;
  }
```

```
user@RR2# show routing-options
route-distinguisher-id 10.255.165.28;
autonomous-system 200;
```

Configuring Device PE2

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 *CLI User Guide*.

To configure Device PE2:

1. Configure the interfaces.

```
[edit interfaces]
user@PE2# set ge-1/0/0 unit 0 description PE2-to-RR1
user@PE2# set ge-1/0/0 unit 0 family inet address 10.50.0.1/30
user@PE2# set ge-1/0/0 unit 0 family mpls

user@PE2# set ge-1/0/1 unit 0 description PE2-to-RR2
user@PE2# set ge-1/0/1 unit 0 family inet address 10.50.10.1/30
user@PE2# set ge-1/0/1 unit 0 family mpls
```

2. Configure the route distinguisher and the AS number.

```
[edit routing-options]
user@PE2# set route-distinguisher-id 10.255.168.42
user@PE2# set autonomous-system 200
```

3. Configure LDP as the signaling protocol used by the VPN.

```
[edit protocols ldp]
user@PE2# set interface ge-1/0/0
user@PE2# set interface ge-1/0/1
```

4. Configure BGP.

```
[edit protocols bgp group internal]
user@PE2# set type internal
user@PE2# set local-address 10.255.168.42
user@PE2# set family inet-vpn unicast
user@PE2# set family route-target
user@PE2# set neighbor 10.255.165.220
user@PE2# set neighbor 10.255.165.28
```

5. Configure OSPF.

```
[edit protocols ospf area 0.0.0.0]
user@PE2# set interface ge-1/0/0
user@PE2# set interface ge-1/0/1
```

```
user@PE2# set interface lo0.0 passive
```

6. Configure the VPN routing instances.

```
[edit routing-instances vpn1]
user@PE2# set instance-type vrf
user@PE2# set vrf-target target:200:100
user@PE2# set routing-options static route 223.1.1.2/32 discard

[edit routing-instances vpn2]
user@PE2# set instance-type vrf
user@PE2# set vrf-target target:200:101
user@PE2# set routing-options static route 223.2.2.2/32 discard

[edit routing-instances vpn3]
user@PE2# set instance-type vrf
user@PE2# set vrf-target target:200:103
user@PE2# set routing-options static route 223.3.3.3/32 discard

[edit routing-instances vpn4]
user@PE2# set instance-type vrf
user@PE2# set vrf-target target:200:104
user@PE2# set routing-options static route 223.4.4.4/32 discard
```

7. If you are done configuring the device, commit the configuration.

```
[edit]
user@PE2# commit
```

Results From configuration mode, confirm your configuration by entering the **show interfaces**, **show protocols**, **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@PE2# show interfaces
ge-1/0/0 {
  unit 0 {
    description PE2-to-RR1;
    family inet {
      address 10.50.0.1/30;
    }
    family mpls;
  }
}
ge-1/0/1 {
  unit 0 {
    description PE2-to-RR2;
    family inet {
      address 10.50.10.1/30;
    }
    family mpls;
  }
}

user@PE2# show protocols
bgp {
  group internal {
    type internal;
```

```
      local-address 10.255.168.42;
      family inet-vpn {
        unicast;
      }
      family route-target;
      neighbor 10.255.165.220;
      neighbor 10.255.165.28;
    }
  }
  ospf {
    area 0.0.0.0 {
      interface ge-1/0/0.0;
      interface ge-1/0/1.0;
      interface lo0.0 {
        passive;
      }
    }
  }
  ldp {
    interface ge-1/0/0.0;
    interface ge-1/0/1.0;
  }
```

```
user@PE2# show routing-options
route-distinguisher-id 10.255.168.42;
autonomous-system 200;
```

```
user@PE2# show routing-instances
vpn1 {
  instance-type vrf;
  vrf-target target:200:100;
  routing-options {
    static {
      route 223.1.1.2/32 discard;
    }
  }
}
vpn2 {
  instance-type vrf;
  vrf-target target:200:101;
  routing-options {
    static {
      route 223.2.2.2/32 discard;
    }
  }
}
vpn3 {
  instance-type vrf;
  vrf-target target:200:103;
  routing-options {
    static {
      route 223.3.3.3/32 discard;
    }
  }
}
vpn4 {
  instance-type vrf;
```

```
    vrf-target target:200:104;
    routing-options {
      static {
        route 223.4.4.4/32 discard;
      }
    }
  }
```

Verification

Confirm that the configuration is working properly.

Verifying the Proxy BGP Route Target Routes

Purpose Verify that the proxy BGP route target routes are displayed in the bgp.rtarget.0 table on Device RR1.

Action From operational mode, enter the **show route table bgp.rtartget.0** command to display the proxy BGP route targets.

```
user@RR1# show route table bgp.rtarget.0
4 destinations, 6 routes (4 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

200:200:100/96
                   *[RTarget/5] 00:01:22
                      Type Proxy
                        for 10.255.163.58
                         Local
                    [BGP/170] 00:04:55, localpref 100, from 10.255.168.42
                      AS path: I, validation-state: unverified
                    > to 10.50.0.1 via ge-1/0/1
200:200:101/96
                   *[RTarget/5] 00:01:22
                      Type Proxy
                        for 10.255.163.58
                         Local
                    [BGP/170] 00:04:55, localpref 100, from 10.255.168.42
                      AS path: I, validation-state: unverified
                    > to 10.50.0.1 via ge-1/0/1
200:200:103/96
                   *[BGP/170] 00:04:55, localpref 100, from 10.255.168.42
                      AS path: I, validation-state: unverified
                    > to 10.50.0.1 via ge-1/0/1
200:200:104/96
                   *[BGP/170] 00:04:55, localpref 100, from 10.255.168.42
                      AS path: I, validation-state: unverified
                    > to 10.50.0.1 via ge-1/0/1
```

Meaning Device RR1 is generating the proxy BGP route target routes on behalf of its peer Device PE1. The proxy BGP route target routes are identified with the protocol and preference **[RTarget/5]** and the route target type of **Proxy**.

Example: Configuring an Export Policy for BGP Route Target Filtering

This example shows how to configure an export routing policy for BGP route target filtering (also known as route target constrain, or RTC).

Requirements

This example uses the following hardware and software components:

- Four Juniper Networks devices that support BGP route target filtering.
- Junos OS Release 12.2 or later on one or more devices configured for proxy BGP route filtering. In this example, you explicitly configure proxy BGP route filtering on the route reflectors.

Before configuring an export policy for BGP route target filtering, make sure that you are familiar with and understand the following concepts:

- *Layer 2 VPNs*
- *Layer 3 VPNs*
- *Route distinguishers*
- Route targets on page 32
- BGP route target filtering on page 38
- *BGP extended communities*

Overview

BGP route target filtering allows you to reduce network resource consumption by distributing route target membership (RT membership) advertisements throughout the network. BGP uses the RT membership information to send VPN routes only to the devices that need them in the network. Similar to other types of BGP reachability, you can apply a routing policy to route target filtering routes to influence the network. When route target filtering is configured, restricting the flow of route target filtering routes also restricts the VPN routes that might be attracted by this RT membership. Configuring this policy involves:

- Creating a filter that defines the list of route target prefixes.
- Creating a policy to select a subset of the route target filters to use for BGP route target filtering.

To define the list of route target prefixes:

- You configure the **rtf-prefix-list** statement at the **[edit policy-options]** hierarchy level to specify the name of the route target prefix list and one or more route target prefixes to use. This configuration allows you to specify the incoming route target filtering routes that the device will use and then distribute them throughout the network.

To configure the routing policy and apply the route target prefix list to that policy, you can specify the following policy options:

- **family route-target**—(Optional) The route-target family match condition specifies matching BGP route target filtering routes. You define this criteria in the **from** statement. This example shows how to create an export policy using the **family route-target** match condition.
- **protocol route-target**—(Optional) The route-target protocol match condition defines the criteria that an incoming route must match. You define this criteria in the **from** statement. This statement is primarily useful for restricting the policy to locally generated route target filtering routes.

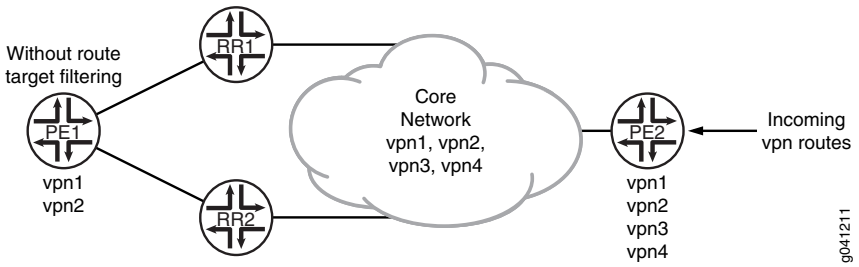
  NOTE: When you use the **show route table bgp.rtarget.0** command to view proxy BGP route target filtering routes, you will see the BGP protocol for received routes and the route target protocol routes for local route target filtering routes.

- **rtf-prefix-list** *name*—The rtf-prefix-list statement applies the list of route target prefixes that you already configured to the policy. You define this criteria in the **from** statement.

Topology Diagram

Figure 5 on page 82 shows the topology used in this example.

Figure 5: BGP Route Target Filtering Export Policy Topology



In this example, BGP route target filtering is configured on the route reflectors (Device RR1 and Device RR2) and provider edge (PE) Device PE2. The other PE, Device PE1, does not support BGP route target filtering. Proxy BGP route target filtering is also configured on the peering sessions between the route reflectors and Device PE1 to minimize the number of VPN route updates processed by Device PE1. Device PE2 has four VPNs configured (vpn1, vpn2, vpn3, and vpn4), and Device PE1 has two VPNs configured (vpn1 and vpn2). In the sample topology, all devices participate in autonomous system (AS) 200, OSPF is the configured interior gateway protocol (IGP), and LDP is the signaling

protocol used by the VPNs. In this example, we use static routes in the VPN routing and forwarding (VRF) instances to generate VPN routes. This is done in place of using a PE to customer edge (CE) protocol such as OSPF or BGP.

In this example, you further control the routes being advertised from Device PE2 to Device PE1 by configuring an export policy on Device PE2 to prevent vpn3 routes from being advertised to Device RR1. You create a policy that specifies the **family route-target** match condition, defines the list of route target prefixes, and applies the list of route target prefixes by defining the **rtf-prefix-list** criteria.

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 PE1

```
set interfaces ge-1/0/0 unit 0 description PE1-to-RR1
set interfaces ge-1/0/0 unit 0 family inet address 10.49.0.1/30
set interfaces ge-1/0/0 unit 0 family mpls
set interfaces ge-1/0/1 unit 0 description PE1-to-RR2
set interfaces ge-1/0/1 unit 0 family inet address 10.49.10.1/30
set interfaces ge-1/0/1 unit 0 family mpls
set protocols ldp interface ge-1/0/0
set protocols ldp interface ge-1/0/1
set protocols bgp group internal type internal
set protocols bgp group internal local-address 10.255.163.58
set protocols bgp group internal neighbor 10.255.165.220 family inet-vpn unicast
set protocols bgp group internal neighbor 10.255.165.28 family inet-vpn unicast
set protocols ospf area 0.0.0.0 interface ge-1/0/0
set protocols ospf area 0.0.0.0 interface ge-1/0/1
set protocols ospf area 0.0.0.0 interface lo0.0 passive
set routing-options route-distinguisher-id 10.255.163.58
set routing-options autonomous-system 200
set routing-instances vpn1 instance-type vrf
set routing-instances vpn1 vrf-target target:200:100
set routing-instances vpn1 routing-options static route 223.1.1.2/32 discard
set routing-instances vpn2 instance-type vrf
set routing-instances vpn2 vrf-target target:200:101
set routing-instances vpn2 routing-options static route 223.2.2.2/32 discard
```

Device RR1

```
set interfaces ge-1/0/0 unit 0 description RR1-to-PE1
set interfaces ge-1/0/0 unit 0 family inet address 10.49.0.2/30
set interfaces ge-1/0/0 unit 0 family mpls
set interfaces ge-1/0/1 unit 0 description RR1-to-PE2
set interfaces ge-1/0/1 unit 0 family inet address 10.50.0.2/30
set interfaces ge-1/0/1 unit 0 family mpls
set protocols ldp interface ge-1/0/0
set protocols ldp interface ge-1/0/1
```

```
set protocols bgp group internal type internal
set protocols bgp group internal local-address 110.255.165.220
set protocols bgp group internal cluster 1.1.1.1
set protocols bgp group internal neighbor 10.255.163.58 description vpn1-to-pe1 family
  inet-vpn unicast
set protocols bgp group internal neighbor 10.255.163.58 family route-target proxy-generate
set protocols bgp group internal neighbor 10.255.168.42 description vpn1-to-pe2 family
  inet-vpn unicast
set protocols ospf area 0.0.0.0 interface ge-1/0/0
set protocols ospf area 0.0.0.0 interface ge-1/0/1
set protocols ospf area 0.0.0.0 interface lo0.0 passive
set routing-options route-distinguisher-id 10.255.165.220
set routing-options autonomous-system 200
```

Device RR2

```
set interfaces ge-1/0/0 unit 0 description RR2-to-PE1
set interfaces ge-1/0/0 unit 0 family inet address 10.49.10.2/30
set interfaces ge-1/0/0 unit 0 family mpls
set interfaces ge-1/0/1 unit 0 description RR2-to-PE2
set interfaces ge-1/0/1 unit 0 family inet address 10.50.10.2/30
set interfaces ge-1/0/1 unit 0 family mpls
set protocols ldp interface ge-1/0/0
set protocols ldp interface ge-1/0/1
set protocols bgp group internal type internal
set protocols bgp group internal local-address 10.255.165.28
set protocols bgp group internal cluster 1.1.1.1
set protocols bgp group internal neighbor 10.255.163.58 description vpn2-to-pe1 family
  inet-vpn unicast
set protocols bgp group internal neighbor 10.255.163.58 family route-target proxy-generate
set protocols bgp group internal neighbor 10.255.168.42 description vpn2-to-pe2 family
  inet-vpn unicast
set protocols bgp group internal neighbor 10.255.163.58 family route-target
set protocols ospf area 0.0.0.0 interface ge-1/0/0
set protocols ospf area 0.0.0.0 interface ge-1/0/1
set protocols ospf area 0.0.0.0 interface lo0.0 passive
set routing-options route-distinguisher-id 10.255.165.28
set routing-options autonomous-system 200
```

Device PE2

```
set interfaces ge-1/0/0 unit 0 description PE2-to-RR1
set interfaces ge-1/0/0 unit 0 family inet address 10.50.0.1/30
set interfaces ge-1/0/0 unit 0 family mpls
set interfaces ge-1/0/1 unit 0 description PE2-to-RR2
set interfaces ge-1/0/1 unit 0 family inet address 10.50.10.2/30
set interfaces ge-1/0/1 unit 0 family mpls
set protocols ldp interface ge-1/0/0
set protocols ldp interface ge-1/0/1
set protocols bgp group internal type internal
set protocols bgp group internal local-address 10.255.168.42
set protocols bgp group internal family inet-vpn unicast
set protocols bgp group internal family route-target
set protocols bgp group internal neighbor 10.255.165.220 export filter-rtc
set protocols bgp group internal neighbor 10.255.165.28
set protocols ospf area 0.0.0.0 interface ge-1/0/0
set protocols ospf area 0.0.0.0 interface ge-1/0/1
set protocols ospf area 0.0.0.0 interface lo0.0 passive
set policy-options rtf-prefix-list exclude-103 200:200:103/96
```

```
set policy-options policy-statement filter-rtc from family route-target
set policy-options policy-statement filter-rtc from rtf-prefix-list exclude-103
set policy-options policy-statement filter-rtc then reject
set routing-options route-distinguisher-id 10.255.168.42
set routing-options autonomous-system 200
set routing-instances vpn1 instance-type vrf
set routing-instances vpn1 vrf-target target:200:100
set routing-instances vpn1 routing-options static route 223.1.1.2/32 discard
set routing-instances vpn2 instance-type vrf
set routing-instances vpn2 vrf-target target:200:101
set routing-instances vpn2 routing-options static route 223.2.2.2/32 discard
set routing-instances vpn3 instance-type vrf
set routing-instances vpn3 vrf-target target:200:103
set routing-instances vpn3 routing-options static route 223.3.3.3/32 discard
set routing-instances vpn4 instance-type vrf
set routing-instances vpn4 vrf-target target:200:104
set routing-instances vpn4 routing-options static route 223.4.4.4/32 discard
```

Configuring Device PE1

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 *CLI User Guide*.

To configure Device PE1:

1. Configure the interfaces.

```
[edit interfaces]
user@PE1# set ge-1/0/0 unit 0 description PE1-to-RR1
user@PE1# set ge-1/0/0 unit 0 family inet address 10.49.0.1/30
user@PE1# set ge-1/0/0 unit 0 family mpls


user@PE1#set ge-1/0/1 unit 0 description PE1-to-RR2
user@PE1#set ge-1/0/1 unit 0 family inet address 10.49.10.1/30
user@PE1# set ge-1/0/1 unit 0 family mpls
```

2. Configure the route distinguisher and the AS number.

```
[edit routing-options]
user@PE1# set route-distinguisher-id 10.255.163.58
user@PE1# set autonomous-system 200
```

3. Configure LDP as the signaling protocol used by the VPN.

```
[edit protocols ldp]
user@PE1# set interface ge-1/0/0
user@PE1# set interface ge-1/0/1
```

4. Configure BGP.

```
[edit protocols bgp group internal]
user@PE1# set type internal
user@PE1# set local-address 10.255.163.58
user@PE1# set neighbor 10.255.165.220 family inet-vpn unicast
user@PE1# set neighbor 10.255.165.28 family inet-vpn unicast
```

5. Configure OSPF.

```
[edit protocols ospf area 0.0.0.0]
user@PE1# set interface ge-1/0/0
user@PE1# set interface ge-1/0/1
user@PE1# set interface lo0.0 passive
```

6. Configure the VPN routing instances.

```
[edit routing-instances vpn1]
user@PE1# set instance-type vrf
user@PE1# set vrf-target target:200:100
user@PE1# set routing-options static route 223.1.1.2/32 discard

[edit routing-instances vpn2]
user@PE1# set instance-type vrf
user@PE1# set vrf-target target:200:101
user@PE1# set routing-options static route 223.2.2.2/32 discard
```

7. If you are done configuring the device, commit the configuration.

```
[edit]
user@PE1# commit
```

Results From configuration mode, confirm your configuration by entering the **show interfaces**, **show protocols**, **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@PE1# show interfaces
ge-1/0/0 {
  unit 0 {
    description PE1-to-RR1;
    family inet {
      address 10.49.0.1/30;
    }
    family mpls;
  }
}
ge-1/0/1 {
  unit 0 {
    description PE1-to-RR2;
    family inet {
      address 10.49.10.1/30;
    }
    family mpls;
  }
}
```

```
user@PE1# show protocols
bgp {
  group internal {
    type internal;
    local-address 10.255.163.58;
    neighbor 10.255.165.220 {
      family inet-vpn {
        unicast;
      }
    }
```

```
      neighbor 10.255.165.28 {
        family inet-vpn {
          unicast;
        }
      }
    }
  }
  ospf {
    area 0.0.0.0 {
      interface ge-1/0/0.0;
      interface ge-1/0/1.0;
      interface lo0.0 {
        passive;
      }
    }
  }
  ldp {
    interface ge-1/0/0.0;
    interface ge-1/0/1.0;
  }
```

```
user@PE1# show routing-options
route-distinguisher-id 10.255.14.182;
autonomous-system 200;
```

```
user@PE1# show routing-instances
vpn1 {
  instance-type vrf;
  vrf-target target:200:100;
  routing-options {
    static {
      route 223.1.1.2/32 discard;
    }
  }
}
vpn2 {
  instance-type vrf;
  vrf-target target:200:101;
  routing-options {
    static {
      route 223.2.2.2/32 discard;
    }
  }
}
```

Configuring Device RR1

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 *CLI User Guide*.

To configure Device RR1:

1. Configure the interfaces.

```
[edit interfaces]
user@RR1# set ge-1/0/0 unit 0 description RR1-to-PE1
```

```
user@RR1# set ge-1/0/0 unit 0 family inet address 10.49.0.2/30
user@RR1# set ge-1/0/0 unit 0 family mpls
user@RR1# set ge-1/0/1 unit 0 description RR1-to-PE2
user@RR1# set ge-1/0/1 unit 0 family inet address 10.50.0.2/30
user@RR1# set ge-1/0/1 unit 0 family mpls
```

2. Configure the route distinguisher and the AS number.

```
[edit routing-options]
user@RR1# set route-distinguisher-id 10.255.165.220
user@RR1# set autonomous-system 200
```

3. Configure LDP as the signaling protocol used by the VPN.

```
[edit protocols ldp]
user@RR1# set interface ge-1/0/0
user@RR1# set interface ge-1/0/1
```

4. Configure BGP.

```
[edit protocols bgp group internal]
user@RR1# set type internal
user@RR1# set local-address 10.255.165.220
user@RR1# set cluster 1.1.1.1
user@RR1# set neighbor 10.255.163.58 description vpn1-to-pe1 family inet-vpn
  unicast
user@RR1# set neighbor 10.255.168.42 description vpn1-to-pe2 family inet-vpn
  unicast
```

5. Configure BGP route target filtering on the peering session with Device PE2.

```
[edit protocols bgp group internal]
user@RR1# set neighbor 10.255.168.42 family route-target
```

6. Configure proxy BGP route target filtering on the peering session with Device PE1.

```
[edit protocols bgp group internal]
user@RR1# set neighbor 10.255.163.58 family route-target proxy-generate
```

7. Configure OSPF.

```
[edit protocols ospf area 0.0.0.0]
user@RR1# set interface ge-1/0/0
user@RR1# set interface ge-1/0/1
user@RR1# set interface lo0.0 passive
```

8. If you are done configuring the device, commit the configuration.

```
[edit]
user@RR1# commit
```

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

```
user@RR1# show interfaces
ge-1/0/0 {
  unit 0 {
    description RR1-to-PE1;
    family inet {
```

```
        address 10.49.0.2/30;
      }
      family mpls;
    }
  }
  ge-1/0/1 {
    unit 0 {
      description RR1-to-PE2;
      family inet {
        address 10.50.0.2/30;
      }
      family mpls;
    }
  }
```

```
user@RR1# show protocols
bgp {
  group internal {
    type internal;
    local-address 110.255.165.220;
    cluster 1.1.1.1;
    neighbor 10.255.163.58 {
      description vpn1-to-pe1;
      family inet-vpn {
        unicast;
      }
      family route-target {
        proxy-generate;
      }
    }
    neighbor 10.255.168.42 {
      description vpn1-to-pe2;
      family inet-vpn {
        unicast;
      }
      family route-target;
    }
  }
}
ospf {
  area 0.0.0.0 {
    interface ge-1/0/0.0;
    interface ge-1/0/1.0;
    interface lo0.0 {
      passive;
    }
  }
}
ldp {
  interface ge-1/0/0.0;
  interface ge-1/0/1.0;
}
ospf {
  area 0.0.0.0 {
    interface ge-1/0/0.0;
    interface ge-1/0/1.0;
```

```
      interface lo0.0 {
        passive;
      }
    }
  }
  ldp {
    interface ge-1/0/0.0;
    interface ge-1/0/1.0;
  }
```

```
user@RR1# show routing-options
route-distinguisher-id 10.255.165.220;
autonomous-system 200;
```

Configuring Device RR2

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 *CLI User Guide*.

To configure Device RR2:

1. Configure the interfaces.

```
[edit interfaces]
user@RR2# set ge-1/0/0 unit 0 description RR2-to-PE1
user@RR2# set ge-1/0/0 unit 0 family inet address 10.49.10.2/30
user@RR2# set ge-1/0/0 unit 0 family mpls


user@RR2# set ge-1/0/1 unit 0 description RR2-to-PE2
user@RR2# set ge-1/0/1 unit 0 family inet address 10.50.10.2/30
user@RR2# set ge-1/0/1 unit 0 family mpls
```

2. Configure the route distinguisher and the AS number.

```
[edit routing-options]
user@RR2# set route-distinguisher-id 10.255.165.28
user@RR2# set autonomous-system 200
```

3. Configure LDP as the signaling protocol used by the VPN.

```
[edit protocols ldp]
user@RR2# set interface ge-1/0/0
user@RR2# set interface ge-1/0/1
```

4. Configure BGP.

```
[edit protocols bgp group internal]
user@RR2# set type internal
user@RR2# set local-address 10.255.165.28
user@RR2# set cluster 1.1.1.1
user@RR2# set neighbor 10.255.163.58 description vpn2-to-pe1 family inet-vpn
    unicast
user@RR2# set neighbor 10.255.168.42 description vpn2-to-pe2 family inet-vpn
    unicast
```

5. Configure BGP route target filtering on the peering session with Device PE2.

```
[edit protocols bgp group internal]
```

```
user@RR2# set neighbor 10.255.168.42 family route-target
```

6. Configure proxy BGP route target filtering on the peering session with Device PE1.

```
[edit protocols bgp group internal]
user@RR2# set neighbor 10.255.163.58 family route-target proxy-generate
```

7. Configure OSPF.

```
[edit protocols ospf area 0.0.0.0]
user@RR2# set interface ge-1/0/0
user@RR2# set interface ge-1/0/1
user@RR2# set interface lo0.0 passive
```

8. If you are done configuring the device, commit the configuration.

```
[edit]
user@RR2# commit
```

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

```
user@RR2# show interfaces
ge-1/0/0 {
  unit 0 {
    description RR2-to-PE1;
    family inet {
      address 10.49.10.2/30;
    }
    family mpls;
  }
}
ge-1/0/1 {
  unit 0 {
    description RR2-to-PE2;
    family inet {
      address 10.50.10.2/30;
    }
    family mpls;
  }
}
```

```
user@RR2# show protocols
bgp {
  group internal {
    local-address 10.255.165.28;
    cluster 1.1.1.1;
    neighbor 10.255.163.58 {
      description vpn2-to-pe1;
      family inet-vpn {
        unicast;
      }
      family route-target {
        proxy-generate;
      }
    }
    neighbor 10.255.168.42 {
```

```
        description vpn2-to-pe2;
        family inet-vpn {
          unicast;
        }
        family route-target;
      }
    }
  }
  ospf {
    area 0.0.0.0 {
      interface ge-1/0/0.0;
      interface ge-1/0/1.0;
      interface lo0.0 {
        passive;
      }
    }
  }
  ldp {
    interface ge-1/0/0.0;
    interface ge-1/0/1.0;
  }
```

```
user@RR2# show routing-options
route-distinguisher-id 10.255.165.28;
autonomous-system 200;
```

Configuring Device PE2

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 *CLI User Guide*.

To configure Device PE2:

1. Configure the interfaces.

```
[edit interfaces]
user@PE2# set ge-1/0/0 unit 0 description PE2-to-RR1
user@PE2# set ge-1/0/0 unit 0 family inet address 10.50.0.1/30
user@PE2# set ge-1/0/0 unit 0 family mpls


user@PE2#set ge-1/0/1 unit 0 description PE2-to-RR2
user@PE2#set ge-1/0/1 unit 0 family inet address 10.50.10.2/30
user@PE2# set ge-1/0/1 unit 0 family mpls
```

2. Configure the route distinguisher and the AS number.

```
[edit routing-options]
user@PE2# set route-distinguisher-id 10.255.168.42
user@PE2# set autonomous-system 200
```

3. Configure LDP as the signaling protocol used by the VPN.

```
[edit protocols ldp]
user@PE2# set interface ge-1/0/0
user@PE2# set interface ge-1/0/1
```

4. Configure BGP.

```
[edit protocols bgp group internal]
user@PE2# set type internal
user@PE2# set local-address 10.255.168.42
user@PE2# set family inet-vpn unicast
user@PE2# set family route-target
user@PE2# set neighbor 10.255.165.220
user@PE2# set neighbor 10.255.165.28
```

5. Configure OSPF.

```
[edit protocols ospf area 0.0.0.0]
user@PE2# set interface ge-1/0/0
user@PE2# set interface ge-1/0/1
user@PE2# set interface lo0.0 passive
```

6. Configure the VPN routing instances.

```
[edit routing-instances vpn1]
user@PE2# set instance-type vrf
user@PE2# set vrf-target target:200:100
user@PE2# set routing-options static route 223.1.1.2/32 discard

[edit routing-instances vpn2]
user@PE2# set instance-type vrf
user@PE2# set vrf-target target:200:101
user@PE2# set routing-options static route 223.2.2.2/32 discard

[edit routing-instances vpn3]
user@PE2# set instance-type vrf
user@PE2# set vrf-target target:200:103
user@PE2# set routing-options static route 223.3.3.3/32 discard

[edit routing-instances vpn4]
user@PE2# set instance-type vrf
user@PE2# set vrf-target target:200:104
user@PE2# set routing-options static route 223.4.4.4/32 discard
```

7. Configure and apply the export routing policy.

```
[edit policy-options]
user@PE2# set rtf-prefix-list exclude-103 200:200:103/96


[edit policy-options policy-statement filter-rtc]
user@PE2# set from family route-target
user@PE2# set from rtf-prefix-list exclude-103
user@PE2# set then reject


[edit protocols bgp group internal]
user@PE2# set neighbor 10.255.165.220 export filter-rtc
```

8. If you are done configuring the device, commit the configuration.

```
[edit]
user@PE2# commit
```

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@PE2# show interfaces
ge-1/0/0 {
    unit 0 {
        description PE2-to-RR1;
        family inet {
            address 10.50.0.1/30;
        }
        family mpls;
    }
}
ge-1/0/1 {
    unit 0 {
        description PE2-to-RR2;
        family inet {
            address 10.50.10.2/30;
        }
        family mpls;
    }
}
```

```
user@PE2# show protocols
    bgp {
        group internal {
            type internal;
            local-address 10.255.168.42;
            family inet-vpn {
                unicast;
            }
            family route-target;
            neighbor 10.255.165.220 {
                export filter-rtc;
            }
            neighbor 10.255.165.28;
        }
    }
ospf {
    area 0.0.0.0 {
        interface ge-1/0/0.0;
        interface ge-1/0/1.0;
        interface lo0.0 {
            passive;
        }
    }
}
ldp {
    interface ge-1/0/0.0;
    interface ge-1/0/1.0;
}
```

```
user@PE2# show routing-options
route-distinguisher-id 10.255.168.42;
autonomous-system 200;
```

```
user@PE2# show policy-options
```

```
policy-statement filter-rtc {
   from {
      family route-target;
      rtf-prefix-list exclude-103;
   }
   then reject;
}
rtf-prefix-list exclude-103 {
   200:200:103/96;
}
```

```
user@PE2# show routing-instances
vpn1 {
   instance-type vrf;
   vrf-target target:200:100;
   routing-options {
      static {
         route 223.1.1.2/32 discard;
      }
   }
}
vpn2 {
   instance-type vrf;
   vrf-target target:200:101;
   routing-options {
      static {
         route 223.2.2.2/32 discard;
      }
   }
}
vpn3 {
   instance-type vrf;
   vrf-target target:200:103;
   routing-options {
      static {
         route 223.3.3.3/32 discard;
      }
   }
}
vpn4 {
   instance-type vrf;
   vrf-target target:200:104;
   routing-options {
      static {
         route 223.4.4.4/32 discard;
      }
   }
}
```

Verification

Confirm that the configuration is working properly.

- Verifying the Route Target Filtering Routes in the bgp.rtarget.0 Routing Table for Device RR1 on page 96
- Verifying the Route Target Filtering Routes in the bgp.rtarget.0 Routing Table for Device RR2 on page 96

Verifying the Route Target Filtering Routes in the bgp.rtarget.0 Routing Table for Device RR1

Purpose Verify that the route prefix for vpn3 is not in Device RR1's bgp.rtarget.0 table. Since an export policy on Device PE2 was applied to prevent the advertisement of vpn3 routes to Device RR1, Device RR1 should not receive those advertisements.

Action From operational mode, enter the **show route advertising-protocol bgp 10.255.165.220 table bgp.rtarget.0** command.

```
user@PE2# show route advertising-protocol bgp 10.255.165.220 table bgp.rtarget.0
bgp.rtarget.0: 4 destinations, 11 routes
(4 active, 0 holddown, 0 hidden)
  Prefix                  Nexthop              MED     Lclpref    AS path
  200:200:100/96          *                    Self      100        I
  200:200:101/96          *                    Self      100        I
  200:200:104/96          *                    Self      100        I
```

Meaning The bgp.rtartget.0 table does not display 200:200:103/96, which is the route prefix for vpn3. That means the export policy was applied correctly.

Verifying the Route Target Filtering Routes in the bgp.rtarget.0 Routing Table for Device RR2

Purpose Verify that the route prefix for vpn3 is in Device RR2's bgp.rtarget.0 table. Since an export policy was not applied on Device PE2 to prevent the advertisement of vpn3 routes to Device RR2, Device RR2 should receive advertisements from all of the VPNs.

Action From operational mode, enter the **show route advertising-protocol bgp 10.255.165.28 table bgp.rtarget.0** command.

```
user@PE2# show route advertising-protocol bgp 10.255.165.28 table bgp.rtarget.0
bgp.rtarget.0: 4 destinations, 11 routes (4 active, 0 holddown, 0 hidden)
(4 active, 0 holddown, 0 hidden)
  Prefix                  Nexthop              MED     Lclpref    AS path
  200:200:100/96          *                    Self      100        I
  200:200:101/96          *                    Self      100        I
  200:200:103/96          *                    Self      100        I
  200:200:104/96          *                    Self      100        I
```

Meaning The bgp.rtartget.0 table displays the route prefixes for all of the VPNs.

Example: Configuring Chained Composite Next Hops for Direct PE-PE Connections

This example shows how to enable back-to-back Provider Edge (PE) router Layer 3 Virtual Private Network (VPN) connections with chained composite next hops for MIC and MPC interfaces on MX Series and T4000 routers.

Requirements

This example uses the following hardware and software components:

- Six routers that can be a combination of MX240, MX480, MX960, or T4000 routers.
- Junos OS Release 13.3 running on all the devices.

Before you begin:

1. Configure the device interfaces.
2. Configure the following routing protocols on all the routers:
   a. MPLS
   b. BGP
   c. LDP LSPs as tunnels between the PE devices
   d. OSPF or any other IGP protocol

Overview

Prior to Junos OS Release 13.3, in a degenerated Layer 3 VPN case without the presence of an MPLS core router, previous behavior of flattened out indirect next hop and unicast next hop was utilized because there was no outer label available in the back-to-back PE-PE connection, and the ingress PE only pushed single VPN labels. In a Layer 3 VPN multipath scenario with mixed PE-PE and PE-P-PE paths, chained composite next hops could not be used either.

On platforms that support only MIC and MPC interfaces, chained composite next hops are enabled by default. On platforms that support both DPC and MPC interfaces, the Layer 3 VPN configuration required the **pe-pe-connection** statement to support chained composite next hops for PE-PE connections. However, the **pe-pe-connection** statement was not supported on platforms with MIC and FPC interfaces only.

As a solution to these limitations, starting with Junos OS Release 13.3, the support for chained composite next hops is enhanced to automatically identify the underlying platform capability on composite next hops at startup time, without relying on user configuration, and to decide the next hop type (composite or indirect) to embed in the

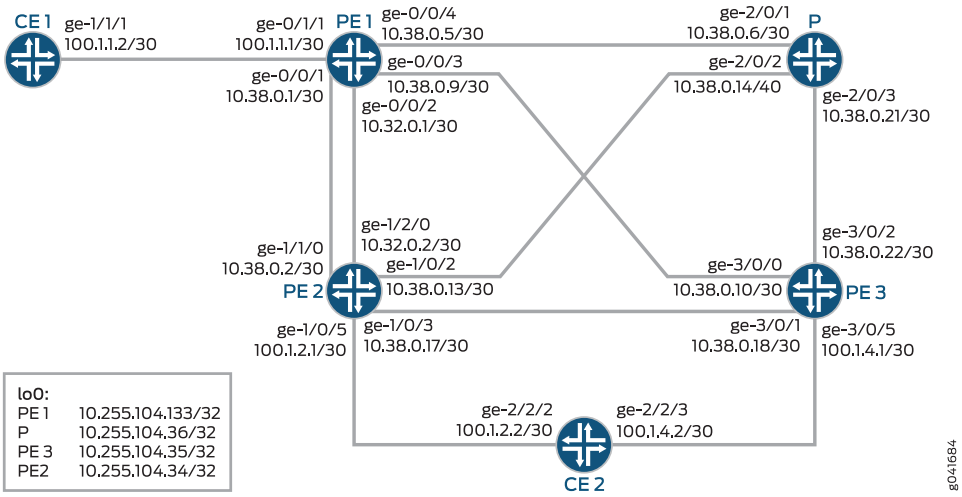
Layer 3 VPN label. This enhances the support for back-to-back PE-PE connections in Layer 3 VPN with composite next hops, and eliminates the need for the **pe-pe-connection** statement.

To enable chained composite next hops:

- On MX Series 3D Universal Edge Routers containing both DPC and MPC FPCs, chained composite next hops are disabled by default. To enable chained composite next hops on the MX240, MX480, and MX960, the chassis must be configured to use the **enhanced-ip** option in network services mode.
- On T4000 Core Routers containing MPC and FPCs, chained composite next hops are disabled by default. To enable chained composite next hops on a T4000 router, the chassis must be configured to use the **enhanced-mode** option in network services mode.

Topology

Figure 6: Chained Composite Next Hop for PE-PE Connections



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.

CE1

```
set interfaces ge-1/1/1 unit 0 family inet address 100.1.1.2/30
set interfaces ge-1/1/1 unit 0 family iso
set interfaces ge-1/1/1 unit 0 family mpls
set interfaces lo0 unit 0 family inet address 1.1.1.1/32
set protocols bgp group PE type external
set protocols bgp group PE peer-as 200
set protocols bgp group PE neighbor 100.1.1.1
set routing-options autonomous-system 100
```

PE1

```
set interfaces ge-0/0/1 unit 0 family inet address 10.38.0.1/30
set interfaces ge-0/0/1 unit 0 family mpls
```

```
set interfaces ge-0/0/2 unit 0 family inet address 10.38.0.5/30
set interfaces ge-0/0/2 unit 0 family mpls
set interfaces ge-0/0/3 unit 0 family inet address 10.38.0.9/30
set interfaces ge-0/0/3 unit 0 family mpls
set interfaces ge-0/0/4 unit 0 family inet address 10.32.0.1/30
set interfaces ge-0/0/4 unit 0 family mpls
set interfaces ge-0/1/1 unit 0 family inet address 100.1.1.1/30
set interfaces ge-0/1/1 unit 0 family mpls
set interfaces lo0 unit 0 family inet address 10.255.104.133/32
set chassis network-services enhanced-ip
set routing-options forwarding-table chained-composite-next-hop ingress l3vpn
set routing-options autonomous-system 200
set routing-options forwarding-table export lbpp
set protocols mpls interface 10.38.0.1/30
set protocols mpls interface 10.32.0.1/30
set protocols mpls interface 10.38.0.5/30
set protocols mpls interface 10.38.0.9/30
set protocols bgp group PEs type internal
set protocols bgp group PEs local-address 10.255.104.133
set protocols bgp group PEs family inet unicast
set protocols bgp group PEs family inet-vpn unicast
set protocols bgp group PEs neighbor 10.255.104.134 local-preference 200
set protocols bgp group PEs neighbor 10.255.104.135
set protocols ospf area 0.0.0.0 interface all
set protocols ospf area 0.0.0.0 interface fxp0.0 disable
set protocols ospf area 0.0.0.0 interface lo0.0 passive
set protocols ldp interface all
set protocols ldp interface fxp0.0 disable
set policy-options policy-statement lbpp then load-balance per-packet
set routing-instances vpn-a instance-type vrf
set routing-instances vpn-a interface ge-0/1/1.0
set routing-instances vpn-a route-distinguisher 200:1
set routing-instances vpn-a vrf-target target:200:1
set routing-instances vpn-a vrf-table-label
set routing-instances vpn-a protocols bgp group CE type external
set routing-instances vpn-a protocols bgp group CE peer-as 100
set routing-instances vpn-a protocols bgp group CE neighbor 100.1.1.2
```

PE2

```
set interfaces ge-1/0/2 unit 0 family inet address 10.38.0.13/30
set interfaces ge-1/0/2 unit 0 family mpls
set interfaces ge-1/0/3 unit 0 family inet address 10.32.0.17/30
set interfaces ge-1/0/3 unit 0 family mpls
set interfaces ge-1/0/5 unit 0 family inet address 100.1.2.1/30
set interfaces ge-1/0/5 unit 0 family mpls
set interfaces ge-1/1/0 unit 0 family inet address 10.38.0.2/30
set interfaces ge-1/1/0 unit 0 family mpls
set interfaces ge-1/2/0 unit 0 family inet address 10.32.0.2/30
set interfaces ge-1/2/0 unit 0 family mpls
set interfaces lo0 unit 0 family inet address 10.255.104.134/32
set chassis network-services enhanced-ip
set routing-options forwarding-table chained-composite-next-hop ingress l3vpn
set routing-instances vpn-a instance-type vrf
set routing-instances vpn-a interface ge-1/0/5.0
set routing-instances vpn-a route-distinguisher 200:2
set routing-instances vpn-a vrf-target target:200:1
set routing-instances vpn-a protocols bgp group CE type external
```

```
set routing-instances vpn-a protocols bgp group CE peer-as 300
set routing-instances vpn-a protocols bgp group CE neighbor 100.1.2.1
set protocols mpls interface 10.38.0.2/30
set protocols mpls interface 10.32.0.2/30
set protocols mpls interface 10.38.0.13/30
set protocols mpls interface 10.38.0.17/30
set protocols bgp group PEs type internal
set protocols bgp group PEs local-address 10.255.104.134
set protocols bgp group PEs family inet unicast
set protocols bgp group PEs family inet-vpn unicast
set protocols bgp group PEs neighbor 10.255.104.133
set protocols bgp group PEs neighbor 10.255.104.135
set protocols ospf area 0.0.0.0 interface all
set protocols ospf area 0.0.0.0 interface fxp0.0 disable
set protocols ospf area 0.0.0.0 interface lo0.0 passive
set protocols ldp interface all
set protocols ldp interface fxp0.0 disable
set routing-options autonomous-system 200
```

P

```
set interfaces ge-2/0/1 unit 0 family inet address 10.38.0.6/30
set interfaces ge-2/0/1 unit 0 family mpls
set interfaces ge-2/0/2 unit 0 family inet address 10.38.0.14/30
set interfaces ge-2/0/2 unit 0 family mpls
set interfaces ge-2/0/3 unit 0 family inet address 10.38.0.21/30
set interfaces ge-2/0/3 unit 0 family mpls
set interfaces lo0 unit 0 family inet address 10.255.104.136/32
set protocols mpls interface 10.38.0.6/30
set protocols mpls interface 10.38.0.14/30
set protocols mpls interface 10.38.0.21/30
set protocols bgp group PEs type internal
set protocols bgp group PEs local-address 10.255.104.136
set protocols bgp group PEs family inet unicast
set protocols bgp group PEs family inet-vpn unicast
set protocols bgp group PEs neighbor 10.255.104.133
set protocols ospf area 0.0.0.0 interface all
set protocols ospf area 0.0.0.0 interface fxp0.0 disable
set protocols ospf area 0.0.0.0 interface lo0.0 passive
set protocols ldp interface all
set protocols ldp interface fxp0.0 disable
set routing-options autonomous-system 200
```

PE3

```
set interfaces ge-3/0/0 unit 0 family inet address 10.38.0.10/30r0-r3
set interfaces ge-3/0/0 unit 0 family mpls
set interfaces ge-3/0/1 unit 0 family inet address 10.38.0.18/30r0-r1-2
set interfaces ge-3/0/1 unit 0 family mpls
set interfaces ge-3/0/2 unit 0 family inet address 10.38.0.22/30
set interfaces ge-3/0/2 unit 0 family mpls
set interfaces ge-3/0/5 unit 0 family inet address 100.1.4.1/30r0-r1-1
set interfaces ge-3/0/5 unit 0 family mpls
set interfaces lo0 unit 0 family inet address 10.255.104.135/32
set chassis network-services enhanced-mode
set routing-options forwarding-table chained-composite-next-hop ingress l3vpn
set routing-options autonomous-system 200
set routing-instances vpn-a instance-type vrf
set routing-instances vpn-a interface ge-3/0/5.0
```

```
set routing-instances vpn-a route-distinguisher 200:3
set routing-instances vpn-a vrf-target target:200:1
set routing-instances vpn-a protocols bgp group CE type external
set routing-instances vpn-a protocols bgp group CE peer-as 300
set routing-instances vpn-a protocols bgp group CE neighbor 100.1.4.2
set protocols mpls interface 10.38.0.10/30
set protocols mpls interface 10.38.0.18/30
set protocols mpls interface 10.38.0.22/30
set protocols bgp group PEs type internal
set protocols bgp group PEs local-address 10.255.104.135
set protocols bgp group PEs family inet unicast
set protocols bgp group PEs family inet-vpn unicast
set protocols bgp group PEs neighbor 10.255.104.133
set protocols bgp group PEs neighbor 10.255.104.134
set protocols ospf area 0.0.0.0 interface all
set protocols ospf area 0.0.0.0 interface fxp0.0 disable
set protocols ospf area 0.0.0.0 interface lo0.0 passive
set protocols ldp interface all
set protocols ldp interface fxp0.0 disable
```

CE2

```
set interfaces ge-2/2/2 unit 0 family inet address 100.1.2.2/30
set interfaces ge-2/2/2 unit 0 family mpls
set interfaces ge-2/2/3 unit 0 family inet address 100.1.4.2/30
set interfaces ge-2/2/3 unit 0 family mpls
set interfaces lo0 unit 0 family inet address 2.2.2.2/32
set protocols bgp group PE type external
set protocols bgp group PE metric-out 50
set protocols bgp group PE peer-as 200
set protocols bgp group PE export s2b
set protocols bgp group PE neighbor 100.1.2.2
set protocols bgp group PE neighbor 100.1.4.2
set policy-options policy-statement s2b from protocol direct
set policy-options policy-statement s2b then accept
set routing-options autonomous-system 300
```

Configuring Multipath Layer 3 VPN with Chained Composite Next Hop

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 basic Layer 3 VPN with chained composite next hop on the PE1 router:

NOTE: Repeat this procedure for the PE2 and PE3 routers in the MPLS domain, after modifying the appropriate interface names, addresses, and any other parameters for each router.

1. Configure the interfaces on the PE1 router.

   PE1 to CE1

   [edit interfaces]

```
user@PE1 # set ge-0/1/1 unit 0 family inet address 100.1.1.1/30
user@PE1 # set ge-0/1/1 unit 0 family mpls
```

PE1 to PE2

```
[edit interfaces]
user@PE1 # set ge-0/0/1 unit 0 family inet address 10.38.0.1/30
user@PE1 # set ge-0/0/1 unit 0 family mpls

user@PE1 # set ge-0/0/2 unit 0 family inet address 10.38.0.5/30
user@PE1 # set ge-0/0/2 unit 0 family mpls
```

PE1 to P

```
[edit interfaces]
user@PE1 # set ge-0/0/4 unit 0 family inet address 10.32.0.1/30
user@PE1 # set ge-0/0/4 unit 0 family mpls
```

PE1 to PE3

```
[edit interfaces]
user@PE1 # set ge-0/0/3 unit 0 family inet address 10.38.0.9/30
user@PE1 # set ge-0/0/3 unit 0 family mpls
```

Loopback interface

```
[edit interfaces]
user@PE1 # set lo0 unit 0 family inet address 10.255.104.133/32
```

2. Enable chained composite next hop on the PE1 chassis.

```
[edit chassis]
use@PE1# set network-services enhanced-ip
```

3. Enable chained composite next hop on the global Layer 3 VPN.

```
[edit routing-options]
use@PE1# set forwarding-table chained-composite-next-hop ingress l3vpn
```

4. Configure the autonomous system for PE1.

```
[edit routing-options]
user@PE1# set autonomous-system 200
```

5. Export the policy configured for load balancing.

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

6. Configure MPLS on the PE1 interfaces connecting to the P router and other PE routers.

```
[edit protocols]
user@PE1# set mpls interface 10.38.0.1/30
user@PE1# set mpls interface 10.32.0.1/30
user@PE1# set mpls interface 10.38.0.5/30
user@PE1# set mpls interface 10.38.0.9/30
```

7. Configure the IBGP group for PE1 to peer with the PE2 and PE3 routers.

```
[edit protocols]
```



```
user@PE1# set bgp group PEs type internal
user@PE1# set bgp group PEs local-address 10.255.104.133
user@PE1# set bgp group PEs family inet unicast
user@PE1# set bgp group PEs family inet-vpn unicast
user@PE1# set bgp group PEs neighbor 10.255.104.134 local-preference 200
user@PE1# set bgp group PEs neighbor 10.255.104.135
```

8. Configure OSPF with traffic engineering capability on all the interfaces of PE1, excluding the management interface.

```
[edit protocols]
user@PE1# set ospf area 0.0.0.0 interface all
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 passive
```

9. Configure LDP on all the interfaces of PE1, excluding the management interface.

```
[edit protocols]
user@PE1# set ldp interface all
user@PE1# set ldp interface fxp0.0 disable
```

10. Configure a policy to load-balance traffic on a per packet basis.

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

11. Configure a VRF routing instance on the CE1-facing interface of PE1.

```
[edit routing-instances]
user@PE1# set vpn-a instance-type vrf
user@PE1# set vpn-a interface ge-0/1/1.0
```

12. Configure the routing instance parameters.

```
[edit routing-instances]
user@PE1# set vpn-a route-distinguisher 200:1
user@PE1# set vpn-a vrf-target target:200:1
user@PE1# set vpn-a vrf-table-label
```

13. Configure an EBGP group for the routing instance, so PE1 can peer with CE1.

```
[edit routing-instances]
user@PE1# set vpn-a protocols bgp group CE type external
user@PE1# set vpn-a protocols bgp group CE peer-as 100
user@PE1# set vpn-a protocols bgp group CE neighbor 100.1.1.2
```

Results

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

PE1

```
user@PE1# show chassis
network-services enhanced-ip;

user@PE1# show interfaces
ge-0/0/1 {
   unit 0 {
```

```
      family inet {
        address 10.38.0.1/30;
      }
      family mpls;
    }
  }
  ge-0/0/2 {
    unit 0 {
      family inet {
        address 10.38.0.5/30;
      }
      family mpls;
    }
  }
  ge-0/0/3 {
    unit 0 {
      family inet {
        address 10.38.0.9/30;
      }
      family mpls;
    }
  }
  ge-0/0/4 {
    unit 0 {
      family inet {
        address 10.32.0.1/30;
      }
      family mpls;
    }
  }
  ge-0/1/1 {
    unit 0 {
      family inet {
        address 100.1.1.1/30;
      }
      family mpls;
    }
  }
  lo0 {
    unit 0 {
      family inet {
        address 10.255.104.133/32;
      }
    }
  }

user@PE1# show protocols
  mpls {
    interface 10.38.0.1/30;
    interface 10.32.0.1/30;
    interface 10.38.0.5/30;
    interface 10.38.0.9/30;
  }
  bgp {
    group PEs {
      type internal;
```

```
      local-address 10.255.104.133;
      family inet {
        unicast;
      }
      family inet-vpn {
        unicast;
      }
      neighbor 10.255.104.134 {
        local-preference 200;
      }
      neighbor 10.255.104.135;
    }
  }
  ospf {
    area 0.0.0.0 {
      interface all;
      interface fxp0.0 {
        disable;
      }
      interface lo0.0 {
        passive;
      }
    }
  }
  ldp {
    interface all;
    interface fxp0.0 {
      disable;
    }
  }
```

```
user@PE1# show routing-options
autonomous-system 200;
forwarding-table {
  export lbpp;
  chained-composite-next-hop {
    ingress {
      l3vpn;
    }
  }
}
```

```
user@PE1# show routing-instances
vpn-a {
  instance-type vrf;
  interface ge-0/1/1.0;
  route-distinguisher 200:1;
  vrf-target target:200:1;
  vrf-table-label;
  protocols {
    bgp {
      group CE {
        type external;
        peer-as 100;
        neighbor 100.1.1.2;
      }
    }
```

```
    }
}

user@PE1# show policy-options
policy-statement lbpp {
    then {
        load-balance per-packet;
    }
}
```

Verification

Confirm that the configuration is working properly.

- Verifying the Routes on page 106
- Verifying Chained Next Hop on Direct PE-PE Connection on page 108

Verifying the Routes

Purpose Verify that the Layer 3 VPN prefixes toward PE1-PE2 point to composite next hops.

Action From operational mode, run the **show route 2.2.2.2 table *vpn-a* extensive** command.

```
user@PE1> show route 2.2.2.2 table vpn-a extensive


vpn-a.inet.0: 7 destinations, 10 routes (7 active, 0 holddown, 0 hidden)
2.2.2.2/32 (2 entries, 1 announced)
TSI:
KRT in-kernel 2.2.2.2/3 -> {composite(720)}
Page 0 idx 0, (group CE type External) Type 1 val 938eaa8 (adv_entry)
   Advertised metrics:
     Nexthop: Self
     AS path: [200] 300 I
     Communities: target:200:1
Path 2.2.2.2 from 10.255.104.133 Vector len 4.  Val: 0
        *BGP    Preference: 170/-101
                Route Distinguisher: 200:2
                Next hop type: Indirect
                Address: 0x9391654
                Next-hop reference count: 12
                Source: 10.255.104.133
                Next hop type: Router, Next hop index: 1048580
                Next hop: 10.32.0.2 via ge-0/0/2.0
                Session Id: 0x1
                Next hop: 10.38.0.2 via ge-0/0/1.0, selected
                Session Id: 0x3
                Protocol next hop: 10.255.104.133
                Push 300192
                Composite next hop: 0x93918a4 718 INH Session ID: 0x9
                Indirect next hop: 0x941c000 1048581 INH Session ID: 0x9
                State: <Secondary Active Int Ext ProtectionCand>
                Local AS:   200 Peer AS:   200
                Age: 28   Metric: 50      Metric2: 1
                Validation State: unverified
                Task: BGP_200.10.255.104.133+57173
                Announcement bits (2): 0-KRT 1-BGP_RT_Background
                AS path: 300 I
                Communities: target:200:1
                Import Accepted
                VPN Label: 300192
                Localpref: 100
                Router ID: 10.255.104.133
                Primary Routing Table bgp.l3vpn.0
                Composite next hops: 1
                        Protocol next hop: 10.255.104.133 Metric: 1
                        Push 300192
                        Composite next hop: 0x93918a4 718 INH Session ID: 0x9
                        Indirect next hop: 0x941c000 1048581 INH Session ID:
0x9
                        Indirect path forwarding next hops: 2
                                Next hop type: Router
                                Next hop: 10.32.0.2 via ge-1/0/0.0
                                Session Id: 0x1
                                Next hop: 10.38.0.2 via ge-1/1/2.0
                                Session Id: 0x3
                  10.255.104.133/32 Originating RIB: inet.3
                    Metric: 1                       Node path count: 1
                    Forwarding nexthops: 2
                          Nexthop: 10.32.0.2 via ge-0/0/2.0
          BGP    Preference: 170/-101
```

```
                        Route Distinguisher: 200:3
                        Next hop type: Indirect
                        Address: 0x9391608
                        Next-hop reference count: 9
                        Source: 10.255.104.131
                        Next hop type: Router, Next hop index: 722
                        Next hop: 10.38.0.10 via ge-0/0/1.0, selected
                        Session Id: 0x4
                        Protocol next hop: 10.255.104.131
                        Push 299936
                        Composite next hop: 0x9391690 723 INH Session ID: 0xb
                        Indirect next hop: 0x941c0fc 1048583 INH Session ID: 0xb
                        State: <Secondary NotBest Int Ext ProtectionCand>
                        Inactive reason: Not Best in its group - Router ID
                        Local AS:   200 Peer AS:   200
                        Age: 28   Metric: 50      Metric2: 1
                        Validation State: unverified
                        Task: BGP_200.10.255.104.131+63797
                        AS path: 300 I
                        Communities: target:200:1
                        Import Accepted
                        VPN Label: 299936
                        Localpref: 100
                        Router ID: 10.255.104.131
                        Primary Routing Table bgp.l3vpn.0
                        Composite next hops: 1
                                Protocol next hop: 10.255.104.131 Metric: 1
                                Push 299936
                                Composite next hop: 0x9391690 723 INH Session ID: 0xb
                                Indirect next hop: 0x941c0fc 1048583 INH Session ID:
0xb
                                Indirect path forwarding next hops: 1
                                        Next hop type: Router
                                        Next hop: 10.38.0.10 via ge-1/0/2.0
                                        Session Id: 0x4
                          10.255.104.131/32 Originating RIB: inet.3
                            Metric: 1                       Node path count: 1
                            Forwarding nexthops: 1
                                  Nexthop: 10.38.0.10 via ge-1/0/2.0
```

Meaning The PE2 router is the composite next hop for PE1 to reach CE2.

Verifying Chained Next Hop on Direct PE-PE Connection

Purpose Verify that chained next hop is generated for direct PE-PE connection on CE1.

Action From operational mode, run the **ping** command.

```
user@CE1> ping 100.1.2.2
!!!!!
--- lsping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
```

Meaning Chained composite next hop is enabled for the PE1 to PE2 connection.

Related Documentation

- Chained Composite Next Hops for Transit Devices on page 12

CHAPTER 5

VPN Configuration Statements

aggregate-label

Syntax

```
aggregate-label {
  community community-name;
}
```

Hierarchy Level

[edit logical-systems *logical-system-name* protocols bgp family inet labeled-unicast],
[edit logical-systems *logical-system-name* protocols bgp family inet6 labeled-unicast],
[edit logical-systems *logical-system-name* protocols bgp family inet-vpn unicast],
[edit logical-systems *logical-system-name* protocols bgp family inet-vpn6 unicast],
[edit protocols bgp family inet labeled-unicast],
[edit protocols bgp family inet6 labeled-unicast],
[edit protocols bgp family inet-vpn unicast],
[edit protocols bgp family inet6-vpn unicast]

Release Information

Statement introduced before Junos OS Release 7.4.
Statement introduced in Junos OS Release 9.0 for EX Series switches.

Description

Specify matching criteria (in the form of a community) such that all routes which match are assigned the same VPN label, selected from one of the several routes in the set defined by this criteria. This reduces the number of VPN labels that the router must consider, and aggregates the received labels.

Options

community *community-name*—Specify the name of the community to which to apply the aggregate label.

Required Privilege Level

routing—To view this statement in the configuration.
routing-control—To add this statement to the configuration.

Related Documentation

- Configuring Aggregate Labels for VPNs on page 47

backup-neighbor

Syntax

```
backup-neighbor address {
  community name;
  mtu number;
  hot-standby;
  psn-tunnel-endpoint address;
  standby;
  static;
  virtual-circuit-id number;
}
```

Hierarchy Level

[edit logical-systems *logical-system-name* protocols l2circuit local-switching interface *interface-name*],
[edit logical-systems *logical-system-name* protocols l2circuit neighbor *address* interface *interface-name*],
[edit logical-systems *logical-system-name* routing-instances *routing-instance-name* protocols vpls neighbor *address*],
[edit protocols l2circuit local-switching interface *interface-name*],
[edit protocols l2circuit neighbor *address* interface *interface-name*],
[edit routing-instances *routing-instance-name* protocols vpls neighbor *address*]

Release Information

Statement introduced in Junos OS Release 9.2.
Statement introduced in Junos OS Release 12.2 for ACX Series routers.
Statement introduced in Junos OS Release 12.3 at the **[edit protocols l2circuit local-switching interface *interface-name*]** hierarchy level.

Description

Configure pseudowire redundancy for Layer 2 circuits and VPLS. A redundant pseudowire can act as a backup connection and can be configured between a PE router and a CE device or between PE routers, maintaining Layer 2 circuit and VPLS services after certain types of failures. This feature can help improve the reliability of certain types of networks where a single point of failure could interrupt service for customers.

NOTE: VPLS is not supported on ACX Series routers. The **psn-tunnel-endpoint** statement is not supported at the **[edit protocols l2circuit local-switching interface *interface-name* end-interface interface *interface-name*]** hierarchy level.

Options

address—Specifies the address for the backup neighbor.

The remaining 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

- Configuring Pseudowire Redundancy on the PE Router on page 44
- *Example: Configuring Layer 2 Circuit Switching Protection*
- *community (Protocols Layer 2 Circuit)*

- *psn-tunnel-endpoint*
- *virtual-circuit-id*

description (Routing Instances)

Syntax description *text*;

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.
Statement introduced in Junos OS Release 11.3 for the QFX Series.
Statement introduced in Junos OS Release 12.3 for ACX Series routers.

Description Provide a text description for the routing instance. If the text includes one or more spaces, enclose it in quotation marks (" "). Any descriptive text you include is displayed in the output of the **show route instance detail** command and has no effect on the operation of the routing instance.

Required Privilege Level routing—To view this statement in the configuration.
routing-control—To add this statement to the configuration.

Related Documentation
- Configuring the Description on page 28
- *show route instance*

family route-target

Syntax

```
family route-target {
  advertise-default;
  external-paths number;
  prefix-limit number;
  proxy-generate <route-target-policy route-target-policy-name>;
}
```

Hierarchy Level

[edit logical-systems *logical-system-name* protocols bgp group *group-name*],
[edit logical-systems *logical-system-name* protocols bgp group *group-name* neighbor *address*],
[edit protocols bgp group *group-name*],
[edit protocols bgp group *group-name* neighbor *address*]

Release Information

Statement introduced before Junos OS Release 7.4.

Description

Enable BGP route target filtering on the VPN.

The **family route-target** statement is useful for filtering VPN routes before they are sent. Provider edge (PE) routers inform the route reflector (RR) which routes to send, using **family route-target** to provide the route-target-interest information. The RR then sends to the PE router only the advertisements containing the specified route target.

Options

advertise-default—Cause the router to advertise the default route target route (**0:0:0/0**) and suppress all routes that are more specific. This can be used by a route reflector on BGP groups consisting of neighbors that act as provider edge (PE) routers only. PE routers often need to advertise all routes to the route reflector. Suppressing all route target advertisements other than the default route reduces the amount of information exchanged between the route reflector and the PE routers. The Junos OS further helps to reduce route target advertisement overhead by not maintaining dependency information unless a nondefault route is received.

external-paths *number*—Cause the router to advertise the VPN routes that reference a given route target. The number you specify with the **external-paths** statement determines the number of external peer routers (currently advertising that route target) that receive the VPN routes. The default value is **1**.

prefix-limit *number*—The number of prefixes that can be received from a peer router.

The remaining statement is described separately.

Required Privilege Level

routing—To view this statement in the configuration.
routing-control—To add this statement to the configuration.

Related Documentation

- Configuring BGP Route Target Filtering for VPNs on page 39
- Example: Configuring Proxy BGP Route Target Filtering on page 65
- Example: Configuring an Export Policy for BGP Route Target Filtering on page 81

graceful-restart (Enabling Globally)

Syntax

```
graceful-restart {
  disable;
  helper-disable;
  maximum-helper-recovery-time seconds;
  maximum-helper-restart-time seconds;
  notify-duration seconds;
  recovery-time seconds;
  restart-duration seconds;
  stale-routes-time seconds;
}
```

Hierarchy Level

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

Release Information

Statement introduced before Junos OS Release 7.4.
Statement introduced in Junos OS Release 9.0 for EX Series switches.
Statement introduced in Junos OS Release 12.1 for the QFX Series.

Description

Configure graceful restart globally to enable the feature. You cannot enable graceful restart for specific protocols unless graceful restart is also enabled globally. You can, optionally, modify the global settings at the individual protocol level.

NOTE:

- For VPNs, the **graceful-restart** statement allows a router whose VPN control plane is undergoing a restart to continue to forward traffic while recovering its state from neighboring routers.
- For BGP, if you configure graceful restart after a BGP session has been established, the BGP session restarts and the peers negotiate graceful restart capabilities.
- LDP sessions flap when **graceful-restart** configurations change.

Default

Graceful restart is disabled by default.

Options

The remaining 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

- *Enabling Graceful Restart*
- *Configuring Routing Protocols Graceful Restart*

- *Configuring Graceful Restart for MPLS-Related Protocols*
- *Configuring VPN Graceful Restart*
- *Configuring Logical System Graceful Restart*
- *Graceful Restart Configuration Statements*
- *Configuring Graceful Restart for QFabric Systems*

instance-type

Syntax instance-type *type*;

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.
virtual-switch and **layer2-control** options introduced in Junos OS Release 8.4.
Statement introduced in Junos OS Release 9.2 for EX Series switches.
Statement introduced in Junos OS Release 11.3 for the QFX Series.
Statement introduced in Junos OS Release 12.3 for ACX Series routers.
evpn option introduced in Junos OS Release 13.2 for MX 3D Series routers.

Description Define the type of routing instance.

Options

NOTE: On ACX Series routers, you can configure only the forwarding, virtual router, and VRF routing instances.

type—Can be one of the following:

- **evpn**—(MX 3D Series routers only) Enable an Ethernet VPN (EVPN) on the routing instance. You cannot configure the **evpn** option under the **[edit logical-systems *logical-system-name* routing-instances *routing-instance-name* instance-type]** hierarchy level.
- **forwarding**—Provide support for filter-based forwarding, where interfaces are not associated with instances. All interfaces belong to the default instance. Other instances are used for populating RPD learned routes. For this instance type, there is no one-to-one mapping between an interface and a routing instance. All interfaces belong to the default instance inet.0.
- **l2backhaul-vpn**—Provide support for Layer 2 wholesale VLAN packets with no existing corresponding logical interface. When using this instance, the router learns both the outer tag and inner tag of the incoming packets, when the **instance-role** statement is defined as **access**, or the outer VLAN tag only, when the **instance-role** statement is defined as **nni**.
- **l2vpn**—Enable a Layer 2 VPN on the routing instance. You must configure the **interface**, **route-distinguisher**, **vrf-import**, and **vrf-export** statements for this type of routing instance.
- **layer2-control**—(MX Series routers only) Provide support for RSTP or MSTP in customer edge interfaces of a VPLS routing instance. This instance type cannot be used if the customer edge interface is multihomed to two provider edge interfaces. If the customer edge interface is multihomed to two provider edge interfaces, use the default BPDU tunneling.

- **no-forwarding**—This is the default routing instance. Do not create a corresponding forwarding instance. Use this routing instance type when a separation of routing table information is required. There is no corresponding forwarding table. All routes are installed into the default forwarding table. IS-IS instances are strictly nonforwarding instance types.
- **virtual-router**—Enable a virtual router routing instance. This instance type is similar to a VPN routing and forwarding instance type, but used for non-VPN-related applications. You must configure the **interface** statement for this type of routing instance. You do not need to configure the **route-distinguisher**, **vrf-import**, and **vrf-export** statements.
- **virtual-switch**—(MX Series routers only) Provide support for Layer 2 bridging. Use this routing instance type to isolate a LAN segment with its Spanning Tree Protocol (STP) instance and to separate its VLAN identifier space.
- **vpls**—Enable VPLS on the routing instance. Use this routing instance type for point-to-multipoint LAN implementations between a set of sites in a VPN. You must configure the **interface**, **route-distinguisher**, **vrf-import**, and **vrf-export** statements for this type of routing instance.
- **vrf**—VPN routing and forwarding (VRF) instance. Provides support for Layer 3 VPNs, where interface routes for each instance go into the corresponding forwarding table only. Required to create a Layer 3 VPN. Create a VRF table (***instance-name*.inet.0**) that contains the routes originating from and destined for a particular Layer 3 VPN. For this instance type, there is a one-to-one mapping between an interface and a routing instance. Each VRF instance corresponds with a forwarding table. Routes on an interface go into the corresponding forwarding table. You must configure the **interface**, **route-distinguisher**, **vrf-import**, and **vrf-export** statements for this type of routing instance.

Required Privilege Level

routing—To view this statement in the configuration.
routing-control—To add this statement to the configuration.

Related Documentation

- *Example: Using Virtual Routing Instances to Route Among VLANs on EX Series Switches*
- Configuring the Instance Type on page 28
- *Configuring EVPN Routing Instances*
- *Configuring Virtual Routing Instances (CLI Procedure)*
- *Configuring Virtual Router Routing Instances*
- *Example: Configuring Filter-Based Forwarding on the Source Address*
- *Example: Configuring Filter-Based Forwarding on Logical Systems*
- *Layer 2 Routing Instance Types*

interface (Routing Instances)

Syntax

```
interface interface-name {
  description text;
}
```

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 9.2 for EX Series switches.
Statement introduced in Junos OS Release 12.3 for ACX Series routers.
Statement introduced in Junos OS Release 13.2 for MX 3D Series routers.

Description Interface over which the VPN traffic travels between the PE device and CE device. You configure the interface on the PE device. If the value **vrf** is specified for the **instance-type** statement included in the routing instance configuration, this statement is required.

Options ***interface-name***—Name of the interface.

Required Privilege Level routing—To view this statement in the configuration.
routing-control—To add this statement to the configuration.

Related Documentation

- Configuring Interfaces for VPN Routing on page 29
- *Configuring EVPN Routing Instances*
- *Example: Configuring MPLS-Based Layer 3 VPNs on EX Series Switches*
- *interface (VPLS Routing Instances)*

no-forwarding

Syntax no-forwarding;

Hierarchy Level
[edit logical-systems *logical-system-name* protocols ldp],
[edit logical-systems *logical-system-name* routing-instances *routing-instance-name* protocols ldp],
[edit protocols ldp],
[edit routing-instances *routing-instance-name* protocols ldp]

Release Information Statement introduced before Junos OS Release 7.4.

Description Do not add ingress routes to the inet.0 routing table even if **traffic-engineering bgp-igp** (configured at the **[edit protocols mpls]** hierarchy level) is enabled.

Default The **no-forwarding** statement is disabled. Ingress routes are added to the inet.0 routing table instead of the inet.3 routing table when **traffic-engineering bgp-igp** is enabled.

Required Privilege Level
routing—To view this statement in the configuration.
routing-control—To add this statement to the configuration.

Related Documentation
- *Configuring Miscellaneous LDP Properties*
- Configuring a Routing Protocol Between the Service Provider Routers on page 41

proxy-generate

Syntax proxy-generate <route-target-policy *route-target-policy-name*>;

Hierarchy Level [edit logical-systems *logical-system-name* protocols bgp group *group-name* family route-target],
[edit logical-systems *logical-system-name* protocols bgp group *group-name* neighbor *address* family route-target],
[edit protocols bgp group *group-name* family route-target],
[edit protocols bgp group *group-name* neighbor *address* family route-target]

Release Information Statement introduced in Junos OS Release 12.2.

Description Enable proxy BGP route target filtering (also known as proxy route target constrain, or proxy RTC). This feature is useful if you have a network environment where route target filtering is not widely deployed or fully supported. When configured for proxy BGP route target filtering, the device creates route target membership (RT membership) on behalf of its peers that do not have the route target filtering functionality. The device then distributes the RT membership advertisements from incoming BGP VPN routes to other devices in the network that need them.

Options route-target-policy *route-target-policy-name*—(Optional) Apply a routing policy that defines a subset of VPN routes to be used in your proxy BGP route target filter. The subset of VPN routes control which proxy BGP route targets are used to create the proxy BGP route target routes.

Required Privilege Level routing—To view this statement in the configuration.
routing-control—To add this statement to the configuration.

Related Documentation

- Example: Configuring Proxy BGP Route Target Filtering on page 66
- Example: Configuring an Export Policy for BGP Route Target Filtering on page 81
- Configuring BGP Route Target Filtering for VPNs on page 38
- family route-target on page 115
- rtf-prefix-list on page 129

revert-time (Protocols Layer 2 Circuits)

Syntax revert-time *seconds* maximum *seconds*;

Hierarchy Level [edit logical-systems *logical-system-name* protocols l2circuit neighbor *address* interface *interface-name*],
[edit logical-systems *logical-system-name* routing-instances *routing-instance-name* protocols vpls neighbor *address*],
[edit protocols l2circuit neighbor *address* interface *interface-name*],
[edit routing-instances *routing-instance-name* protocols vpls neighbor *address*]

Release Information Statement introduced in Junos OS Release 10.2.
maximum option introduced in Junos OS Release 13.2.

Description Specify a revert time for redundant Layer 2 circuits and VPLS pseudowires. When you have configured redundant pseudowires for Layer 2 circuits or VPLS, traffic is switched to the backup connection in the event that the primary connection fails. If you configure a revert time, when the configured time expires traffic is reverted to the primary path, assuming the primary path has been restored.

With the **maximum** option, specify a maximum reversion interval to add after the **revert-time** delay. If a revert-time delay is defined but a maximum timer is not defined, VCs are restored upon the revert-timer's expiration.

To reduce as much as possible the amount of traffic discarded, and potential data-path asymmetries observed during primary-to-backup transition periods, you can use this restoration timer. This restoration timer is activated when the backup path is performing as active, and then the primary path is restored. The goal is to avoid moving traffic back to the primary path right away, to make sure that the control plane's related tasks (such as IGP, LDP, RSVP, and internal BGP) have enough time to complete their updating cycle.

By enabling a gradual return of traffic to the primary path, you can ensure that the relatively-slow control-plane processing and updating does not have a negative impact on the restoration process.

The **maximum** option extends the revert timer's functionality to provide a jittered interval over which a certain number of circuits can be transitioned back to the primary path. By making use of this maximum value, you can define a time interval during which circuits are expected to switch over. As a consequence, circuits' effective transitions are scattered during restoration periods.

When making use of **revert-time *x* maximum *y*** statement, you can ensure that the corresponding circuit that is active is moved to the primary path within a time-slot (t1) such as that: x <= t1 <= y. In other words, by activating this statement, you can ensure the following:

- VCs stay in the backup path for at least x seconds after the primary path comes back up.
- VCs are moved back to the primary path before y seconds have elapsed.
- y maximum value = x maximum value * 2 = 1200 seconds.

The ideal values for x and y will are conditioned to internal aspects of your network. For this reason, there are no default values for these settings. If no revert-time is set, the default behavior is non-revertive. That is, circuits are not returned to the primary path upon restoration. They are kept on the backup path.

Default Without the **revert-time** statement, virtual circuit (VC) traffic is not transitioned to the primary path upon restoration of the primary path.

Options *seconds*—Revert time in seconds.

Range: 0 through 600 seconds

maximum *seconds*—Number of seconds to delay path restoration after the **revert-time** delay.

Range: 0 through 1200 seconds

Required Privilege Level routing—To view this statement in the configuration.
routing-control—To add this statement to the configuration.

Related Documentation

- Configuring Redundant Pseudowires for Layer 2 Circuits and VPLS on page 43

route-distinguisher

Syntax route-distinguisher (*as-number:id* | *ip-address:id*);

Hierarchy Level [edit logical-systems *logical-system-name* routing-instances *routing-instance-name*],
[edit logical-systems *logical-system-name* routing-instances *routing-instance-name* protocols l2vpn mesh-group *mesh-group-name*],
[edit logical-systems *logical-system-name* routing-instances *routing-instance-name* protocols vpls mesh-group *mesh-group-name*],
[edit routing-instances *routing-instance-name*],
[edit routing-instances *routing-instance-name* protocols l2vpn mesh-group *mesh-group-name*],
[edit routing-instances *routing-instance-name* protocols vpls mesh-group *mesh-group-name*]

Release Information Statement introduced before Junos OS Release 7.4.
Statement introduced in Junos OS Release 11.1 for EX Series switches.
Support at **[edit routing-instances *routing-instance-name* protocols vpls mesh-group *mesh-group-name*]** hierarchy level introduced in Junos OS Release 11.2.
Statement introduced in Junos OS Release 12.3 for ACX Series routers.
Support at **[edit routing-instances *routing-instance-name* protocols l2vpn mesh-group *mesh-group-name*]** hierarchy level introduced in Junos OS Release 13.2.

Description Specify an identifier attached to a route, enabling you to distinguish to which VPN or VPLS the route belongs. Each routing instance must have a unique route distinguisher associated with it. The route distinguisher is used to place bounds around a VPN so that the same IP address prefixes can be used in different VPNs without having them overlap. If the instance type is **vrf**, the **route-distinguisher** statement is required.

For Layer 2 VPNs and VPLS, if you configure the **l2vpn-use-bgp-rules** statement, you must configure a unique route distinguisher for each PE router participating in the routing instance.

For other types of VPNs, we recommend that you use a unique route distinguisher for each PE router participating in specific routing instance. Although you can use the same route distinguisher on all PE routers for the same VPN routing instance, if you use a unique route distinguisher, you can determine the CE router from which a route originated within the VPN.

For Layer 2 VPNs and VPLS, if you configure mesh groups, the route distinguisher in each mesh group must be unique.

CAUTION: We strongly recommend that if you change a route distinguisher that has already been configured, make the change during a maintenance window, as follows:

1. Deactivate the routing instance.
2. Change the route distinguisher.
3. Activate the routing instance.

This is not required if you are configuring the route distinguisher for the first time.

Options

as-number:number*—*as-number is an assigned AS number, and ***number*** is any 2-byte or 4-byte value. The AS number can be from 1 through 4,294,967,295. If the AS number is a 2-byte value, the administrative number is a 4-byte value. If the AS number is 4-byte value, the administrative number is a 2-byte value. A route distinguisher consisting of a 4-byte AS number and a 2-byte administrative number is defined as a type 2 route distinguisher in RFC 4364 *BGP/MPLS IP Virtual Private Networks (VPNs)*.

NOTE: In Junos OS Release 9.1 and later, the numeric range for AS numbers is extended to provide BGP support for 4-byte AS numbers, as defined in RFC 4893, *BGP Support for Four-octet AS Number Space*. All releases of Junos OS support 2-byte AS numbers. To configure a route distinguisher that includes a 4-byte AS number, append the letter "L" to the end of the AS number. For example, a route distinguisher with the 4-byte AS number 7,765,000 and an administrative number of 1,000 is represented as 77765000L:1000.

In Junos OS Release 9.2 and later, you can also configure a 4-byte AS number using the AS dot notation format of two integer values joined by a period: <*16-bit high-order value in decimal*>.<*16-bit low-order value in decimal*>. For example, the 4-byte AS number of 65,546 in the plain-number format is represented as 1.10 in AS dot notation format.

ip-address*:*id—IP address (***ip-address*** is a 4-byte value) within your assigned prefix range and a 2-byte value for the ***id***. The IP address can be any globally unique unicast address.

Range: 0 through 4,294,967,295 ($2^{32} - 1$). If the router you are configuring is a BGP peer of a router that does not support 4-byte AS numbers, you need to configure a local AS number. For more information, see *Using 4-Byte Autonomous System Numbers in BGP Networks Technology Overview*.

Required Privilege Level

routing—To view this statement in the configuration.
routing-control—To add this statement to the configuration.

Related Documentation

- Example: Configuring BGP Route Target Filtering for VPNs on page 53
- *Example: Configuring FEC 129 BGP Autodiscovery for VPWS*
- *Configuring EVPN Routing Instances*
- Configuring the Route Distinguisher on page 30
- *Configuring an MPLS-Based Layer 2 VPN (CLI Procedure)*
- *Configuring an MPLS-Based Layer 3 VPN (CLI Procedure)*
- *l2vpn-use-bgp-rules*

route-distinguisher-id

Syntax route-distinguisher-id *ip-address*;

Hierarchy Level [edit logical-systems *logical-system-name* routing-options],
[edit routing-options]

Release Information Statement introduced before Junos OS Release 7.4.
Statement introduced in Junos OS Release 9.0 for EX Series switches.

Description Automatically assign a route distinguisher to the routing instance.

If you configure the **route-distinguisher** statement in addition to the **route-distinguisher-id** statement, the value configured for **route-distinguisher** supersedes the value generated from **route-distinguisher-id**.

NOTE: To avoid a conflict in the two route distinguisher values, it is recommended to ensure that the first half of the route distinguisher obtained by configuring the **route-distinguisher** statement is different from the first half of the route distinguisher obtained by configuring the **route-distinguisher-id** statement.

Options ***ip-address***—Address for routing instance.

Required Privilege Level routing—To view this statement in the configuration.
routing-control—To add this statement to the configuration.

Related Documentation

- Example: Configuring BGP Route Target Filtering for VPNs on page 53
- Configuring the Route Distinguisher on page 30

route-target-filter

Syntax

```
route-target-filter destination {
  group group-name;
  local;
  neighbor address;
}
```

Hierarchy Level [edit logical-systems *logical-system-name* routing-options rib bgp.rtarget.0 static], [edit routing-options rib bgp.rtarget.0 static]

Release Information Statement introduced in Junos OS Release 12.2.
Statement introduced in Junos OS Release 12.3 for ACX Series routers.

Description Statically configure route target filtering. Route target filtering allows you to distribute VPN routes to only the routers that need them. In VPN networks without route target filtering configured, BGP distributes all static VPN routes to all of the VPN peer routers. You can add static routes to the bgp.rtarget.0 routing table with specific NLRI-imposed constraints.

Options

destination—Allows you to specify the static route destination. This value must be in the format x:y/len. The x value is either an IP address or an AS number followed by an optional L to indicate a 4 byte AS number, and y is a number (for example, 123456L:100/64).

group *group-name(s)*—Installs an RT-Constrain filter for the destination for all peers in the specified BGP group. The route and corresponding BGP group are displayed in the output of the **show bgp group rtf detail** command.

local—Causes the router to originate the route target constrain NLRI, but does not install any filtering state for the prefix. This behavior can be useful when the router should always receive VPN routes with this route-target regardless of the state of a given BGP peering session or group status. The route is not displayed in the output of the **show bgp group rtf detail** command unless it is also included in either the BGP neighbor or BGP group configuration.

neighbor *address*—Installs an RT-Constrain filter for the destination for this BGP neighbor. The route is displayed in the output of the **show bgp group rtf detail** command. Specify the BGP neighbor using the router's 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 Static Route Target Filtering for VPNs on page 40
- Reducing Network Resource Use with Static Route Target Filtering on page 8

rtf-prefix-list

Syntax rtf-prefix-list *name route-targets*

Hierarchy Level [edit logical-systems *logical-system-name* policy-options],
[edit logical-systems *logical-system-name* policy-options policy-statement *policy-name* term *term-name*],
[edit policy-options],
[edit policy-options policy-statement *policy-name* term *term-name*]

Release Information Statement introduced in Junos OS Release 12.2.

Description Define a list of route target prefixes for use in a routing policy statement. These prefixes are only useful for filtering routes in the bpg.rtarget.0 table.

The route target filtering prefix is in the format: *AS number:route target extended community/length*. The first number represents the autonomous system (AS) of the device that sent the advertisement. The second group of numbers represent the route target extended community. The format of the extended community is the same as the extended community type **target:**. For more information about extended communities, see *Understanding How to Define BGP Communities and Extended Communities* .

In this route target prefix example 200:200:101/96, 200 is the AS number, 200:101 is the BGP extended community used for the route target, and 96 is the prefix length.

For more information about the route target community, see RFC 4360, *BGP Extended Communities Attribute*.

For more information about the route target filtering prefix format, see RFC 4684, *Constrained Route Distribution for Border Gateway Protocol/MultiProtocol Label Switching (BGP/MPLS) Internet Protocol (IP) Virtual Private Networks (VPNs)*.

Options

name—Name that identifies the list of route target filtering prefixes. The name can contain letters, numbers, and hyphens (-) and can be up to 255 characters long. To include spaces in the name, enclose the entire name in quotation marks (“ “).

route-targets—List of route target filtering prefixes, one route target filter per line in the configuration. When you use the **rtf-prefix-list** statement as a match condition, you do not have the option of configuring the list of route target filtering prefixes. You must first define and configure the route target filtering prefixes with the **policy-options** statement.

Required Privilege Level routing—To view this statement in the configuration.
routing-control—To add this statement to the configuration.

Related Documentation

- Example: Configuring an Export Policy for BGP Route Target Filtering on page 81
- Configuring BGP Route Target Filtering for VPNs on page 38
- Understanding Proxy BGP Route Target Filtering on page 65

- *Understanding How to Define BGP Communities and Extended Communities* in the *Routing Policy Feature Guide for Routing Devices*
- family route-target on page 115

switchover-delay

Syntax switchover-delay *milliseconds*;

Hierarchy Level [edit logical-systems *logical-system-name* protocols l2circuit neighbor *address* interface *interface-name*],
[edit logical-systems *logical-system-name* routing-instances *routing-instance-name* protocols vpls neighbor *address*],
[edit protocols l2circuit neighbor *address* interface *interface-name*],
[edit routing-instances *routing-instance-name* protocols vpls neighbor *address*]

Release Information Statement introduced in Junos OS Release 9.2.

Description After the primary pseudowire goes down, specifies the delay (in milliseconds) to wait before the backup pseudowire takes over. You configure this statement for each backup neighbor configuration to adjust the switchover time after a failure is detected.

Options ***milliseconds***—Specify the time to wait before switching to the backup pseudowire after the primary pseudowire fails.

Default: 10,000 milliseconds

Range: 0 through 180,000 milliseconds

Required Privilege Level routing—To view this statement in the configuration.
routing-control—To add this statement to the configuration.

Related Documentation
- Configuring the Switchover Delay for the Pseudowires on page 44

unicast-reverse-path

Syntax unicast-reverse-path (active-paths | feasible-paths);

Hierarchy Level [edit logical-systems *logical-system-name* routing-options forwarding-table],
[edit routing-instances *routing-instance-name* instance-type *name* routing-options forwarding-table],
[edit routing-options forwarding-table]

Release Information Statement introduced before Junos OS Release 7.4.
Support for routing instances added in Junos OS Release 8.3.
Statement introduced in Junos OS Release 12.3 for ACX Series routers.

Description Control the operation of unicast reverse-path-forwarding check. This statement enables the RPF check to be used when routing is asymmetrical.

Options **active-paths**—Consider only active paths during the unicast reverse-path check.

feasible-paths—Consider all feasible paths during the unicast reverse-path check.

Default: If you omit the **unicast-reverse-path** statement, only the active paths to a particular destination are considered.

Required Privilege Level routing—To view this statement in the configuration.
routing-control—To add this statement to the configuration.

Related Documentation

- *Example: Configuring Unicast Reverse-Path-Forwarding Check*
- Enabling Unicast Reverse-Path Forwarding Check for VPNs on page 49

vpn-apply-export

Syntax vpn-apply-export;

Hierarchy Level [edit logical-systems *logical-system-name* protocols bgp],
[edit logical-systems *logical-system-name* protocols bgp group *group-name*],
[edit logical-systems *logical-system-name* protocols bgp group *group-name* neighbor *neighbor*],
[edit protocols bgp],
[edit protocols bgp group *group-name*],
[edit protocols bgp group *group-name* neighbor *neighbor*]

Release Information Statement introduced before Junos OS Release 7.4.

Description Apply both the VRF export and BGP group or neighbor export policies (VRF first, then BGP) before routes from the **vrf** or **l2vpn** routing tables are advertised to other PE routers.

Required Privilege Level routing—To view this statement in the configuration.
routing-control—To add this statement to the configuration.

Related Documentation

- Applying Both the VRF Export and the BGP Export Policies on page 37

vrf-export

Syntax vrf-export [*policy-names*];

Hierarchy Level
[edit logical-systems *logical-system-name* routing-instances *routing-instance-name*],
[edit logical-systems *logical-system-name* routing-instances *routing-instance-name* protocols vpls mesh-group *mesh-group-name*]
[edit routing-instances *routing-instance-name*]
[edit routing-instances *routing-instance-name* protocols vpls mesh-group *mesh-group-name*]

Release Information
Statement introduced before Junos OS Release 7.4.
Statement introduced in Junos OS Release 11.1 for EX Series switches.
Statement introduced in Junos OS Release 12.3 for ACX Series routers.

Description Specify how routes are exported from the local PE router's VRF table (*routing-instance-name*.inet.0) to the remote PE router. If the value **vrf** is specified for the **instance-type** statement included in the routing instance configuration, this statement is required.

You can configure multiple export policies on the PE router or PE switch (EX8200 switch only).

Default If the instance-type is **vrf**, **vrf-export** is a required statement. The default action is to reject.

Options ***policy-names***—Names for the export policies.

Required Privilege Level
routing—To view this statement in the configuration.
routing-control—To add this statement to the configuration.

Related Documentation

- instance-type on page 118
- Configuring an Export Policy for the PE Router's VRF Table on page 35

vrf-import

Syntax vrf-import [*policy-names*];

Hierarchy Level [edit logical-systems *logical-system-name* routing-instances *routing-instance-name*],
[edit logical-systems *logical-system-name* routing-instances *routing-instance-name* protocols vpls mesh-group *mesh-group-name*]
[edit routing-instances *routing-instance-name*]
[edit routing-instances *routing-instance-name* protocols vpls mesh-group *mesh-group-name*]

Release Information Statement introduced before Junos OS Release 7.4.
Statement introduced in Junos OS Release 11.1 for EX Series switches.

Description Specify how routes are imported into the VRF table (***routing-instance-name***.inet.0) of the local provider edge (PE) router or switch (EX8200 only) from the remote PE. If the value **vrf** is specified for the **instance-type** statement included in the routing instance configuration, this statement is required.

You can configure multiple import policies on the PE router or PE switch (EX8200 switch only).

Default If the instance-type is **vrf**, **vrf-import** is a required statement. The default action is to accept.

Options ***policy-names***—Names for the import policies.

Required Privilege Level routing—To view this statement in the configuration.
routing-control—To add this statement to the configuration.

Related Documentation

- instance-type on page 118
- Configuring an Import Policy for the PE Router's VRF Table on page 34

vrf-mtu-check

Syntax vrf-mtu-check;

Hierarchy Level [edit chassis]

Release Information Statement introduced before Junos OS Release 7.4.
Statement introduced in Junos OS Release 11.1 for EX Series switches.

Description On M Series routers (except the M120 and M320 router), T Series routers, and on EX Series 8200 switches, configure path maximum transmission unit (MTU) checks on the outgoing interface for unicast traffic routed on a virtual private network (VPN) routing and forwarding (VRF) instance.

Default Disabled.

Required Privilege Level interface—To view this statement in the configuration.
interface-control—To add this statement to the configuration.

Related Documentation

- Configuring Path MTU Checks for VPNs on page 48
- *Configuring the Junos OS to Enable MTU Path Check for a Routing Instance on M Series Routers*

vrf-target

Syntax

```
vrf-target {
  community;
  import community-name;
  export community-name;
}
```

Hierarchy Level

[edit logical-systems *logical-system-name* routing-instances *routing-instance-name*],
[edit logical-systems *logical-system-name* routing-instances *routing-instance-name* protocols l2vpn mesh-group *mesh-group-name*]
[edit logical-systems *logical-system-name* routing-instances *routing-instance-name* protocols vpls mesh-group *mesh-group-name*]
[edit routing-instances *routing-instance-name*],
[edit routing-instances *routing-instance-name* protocols l2vpn mesh-group *mesh-group-name*]
[edit routing-instances *routing-instance-name* protocols vpls mesh-group *mesh-group-name*]

Release Information

Statement introduced before Junos OS Release 7.4.
Statement introduced in Junos OS Release 11.1 for EX Series switches.
Statement introduced in Junos OS Release 12.3 for ACX Series routers.

Description

Specify a VRF target community. If you configure the ***community*** option only, default VRF import and export policies are generated that accept and tag routes with the specified target community. The purpose of the **vrf-target** statement is to simplify the configuration by allowing you to configure most statements at the **[edit routing-instances]** hierarchy level. In effect, this statement configures a single policy for import and a single policy for export to replace the per-VRF policies for every community.

You can still create more complex policies by explicitly configuring VRF import and export policies using the **import** and **export** options.

Options

community—Community name.

import *community-name*—Allowed communities accepted from neighbors.

export *community-name*—Allowed communities sent to neighbors.

Required Privilege Level

routing—To view this statement in the configuration.
routing-control—To add this statement to the configuration.

Related Documentation

- Configuring a VRF Target on page 37
- *Example: Configuring FEC 129 BGP Autodiscovery for VPWS*

PART 3

Administration

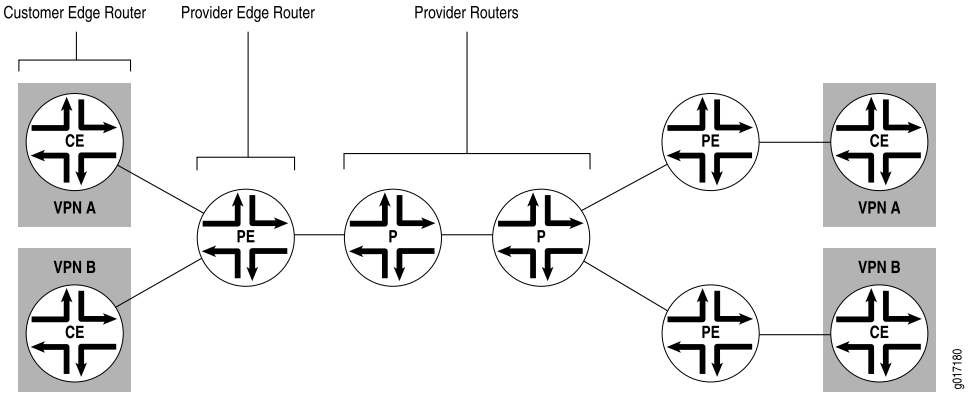
CHAPTER 6

VPN References

Routers in a VPN

Figure 7 on page 139 illustrates how VPN functionality is provided by the provider edge (PE) routers; the provider and customer edge (CE) routers have no special configuration requirements for VPNs.

Figure 7: Routers in a VPN



VPN Terminology

C

Customer edge (CE) devices — Routers or switches located at the customer site that connect to the provider's network. CE devices are typically IP routers, but could also be an Asynchronous Transfer Mode (ATM), Frame Relay, or Ethernet switch.

P

Provider (P) routers — Routers within the core of the provider's network that are not connected to any routers at a customer site but are part of the tunnel between pairs of PE routers. P routers support MPLS LSP or LDP functionality, but do not need to support VPN functionality.

Provider edge (PE) routers Routers in the provider's network that connect to customer edge devices located at customer sites. PE routers support VPN and label functionality. (The label functionality can be provided either by the Resource Reservation Protocol [RSVP] or Label Distribution Protocol [LDP].) Within a single VPN, pairs of PE routers are connected through a tunnel, which can be either an MPLS label-switched path (LSP) or an LDP tunnel.

PART 4

Troubleshooting

CHAPTER 7

Troubleshooting VPNs

Pinging a Layer 2 VPN

To ping a Layer 2 VPN, use one of the following commands:

- **ping mpls l2vpn interface** ***interface-name***

  You ping an interface configured for the Layer 2 VPN on the egress PE router.

- **ping mpls l2vpn instance** ***l2vpn-instance-name*** **local-site-id** ***local-site-id-number*** **remote-site-id** ***remote-site-id-number***

  You ping a combination of the Layer 2 VPN routing instance name, the local site identifier, and the remote site identifier to test the integrity of the Layer 2 VPN connection (specified by the identifiers) between the ingress and egress PE routers.

Related Documentation

- *Example: Configuring MPLS-Based Layer 2 VPNs*

Pinging a Layer 3 VPN

To ping a Layer 3 VPN, use the following command:

ping mpls l3vpn *l3vpn-name* prefix *prefix* <count *count*>

You ping a combination of an IPv4 destination prefix and a Layer 3 VPN name on the egress PE router to test the integrity of the VPN connection between the ingress and egress PE routers. The destination prefix corresponds to a prefix in the Layer 3 VPN. However, the ping tests only whether the prefix is present in a PE router's VRF table. It does not test the connection between a PE router and a CE router.

Pinging a Layer 2 Circuit

To ping a Layer 2 circuit, use one of the following commands:

- **ping mpls l2circuit interface** *interface-name*

  You ping an interface configured for the Layer 2 circuit on the egress PE router.

- **ping mpls l2circuit virtual-circuit neighbor** <*prefix*> <*virtual-circuit-id*>

  You ping a combination of the IPv4 prefix and the virtual circuit identifier on the egress PE router to test the integrity of the Layer 2 circuit between the ingress and egress PE routers.

PART 5

Index

Index

Symbols

A

B

C

D

E

F

G

I

L

T

U

V