JUNIPER NETWORKS

Junos® OS

EVPN Control Plane and VXLAN Data Plane Feature Guide for QFX5100 Switches

Release
14.1X53-D40

Modified: 2016-11-07

Juniper Networks, Inc.
1133 Innovation Way
Sunnyvale, California 94089
USA
408-745-2000
www.juniper.net

Junos® OS EVPN Control Plane and VXLAN Data Plane Feature Guide for QFX5100 Switches
14.1X53-D40

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:

- QFX 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 CLI Explorer.

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 xiii 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**

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

Convention	Description	Examples
`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		
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**.

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

Convention	Description	Examples
> (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 provide feedback by using either of the following methods:

- Online feedback rating system—On any page of the Juniper Networks TechLibrary site at http://www.juniper.net/techpubs/index.html, simply click the stars to rate the content, and use the pop-up form to provide us with information about your experience. Alternately, you can use the online feedback form at http://www.juniper.net/techpubs/feedback/.
- E-mail—Send your comments to **techpubs-comments@juniper.net**. Include the document or topic name, URL or page number, and software 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 Partner Support Service 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

EVPN and VXLAN Overview

Understanding VXLANs

Virtual eXtensible LAN protocol (VXLAN) technology allows networks to support more VLANs. According to the IEEE 802.1Q standard, traditional VLAN identifiers are 12-bits long—this naming limits networks to 4094 VLANs. The VXLAN protocol overcomes this limitation by using a longer logical network identifier that allows more VLANs and, therefore, more logical network isolation for large networks such as clouds that typically include many virtual machines.

VXLAN Benefits

VXLAN technology allows you to segment your networks (as VLANs do) but it provides benefits that VLANs cannot. Here are the most important benefits of using VXLANs:

- You can theoretically create as many as 16 million VXLANs in an administrative domain (as opposed to 4094 VLANs on a Juniper Networks device).
  - MX Series routers and EX9200 switches support as many as 32K VXLANs, 32K multicast groups, and 8K virtual tunnel endpoints (VTEPs). This means that VXLANs based on MX Series routers provide network segmentation at the scale required by cloud builders to support very large numbers of tenants.
  - QFX 5100 switches support 4K VXLANs, 4K multicast groups, and 2K VTEPs.
- You can enable migration of virtual machines between servers that exist in separate Layer 2 domains by tunneling the traffic over Layer 3 networks. This functionality allows you to dynamically allocate resources within or between data centers without being constrained by Layer 2 boundaries or being forced to create large or geographically stretched Layer 2 domains.

Using VXLANs to create smaller Layer 2 domains that are connected over a Layer 3 network means that you don't need to use STP to converge the topology but can use more-robust routing protocols in the Layer 3 network instead. In the absence of STP, none of your links are blocked, which means you can get full value from all the ports that you purchase. Using routing protocols to connect your Layer 2 domains also allows you to load balance the traffic to ensure that you get the best use of your available bandwidth. Given the amount of east-west traffic that often flows within or between data centers, maximizing your network performance for that traffic is very important.

The video *Why Use an Overlay Network in a Data Center?* presents a brief overview of the advantages of using VXLANs.

Video: Why Use an Overlay Network in a Data Center?

How Does VXLAN Work?

VXLAN is often described as an overlay technology because it allows you to stretch Layer 2 connections over an intervening Layer 3 network by encapsulating (tunneling) Ethernet frames in a VXLAN packet that includes IP addresses. Devices that support VXLANs are called virtual tunnel endpoints (VTEPs)—they can be end hosts or network switches or routers. VTEPs encapsulate VXLAN traffic and de-encapsulate that traffic when it leaves the VXLAN tunnel. To encapsulate an Ethernet frame, VTEPs add a number of fields, including the following:

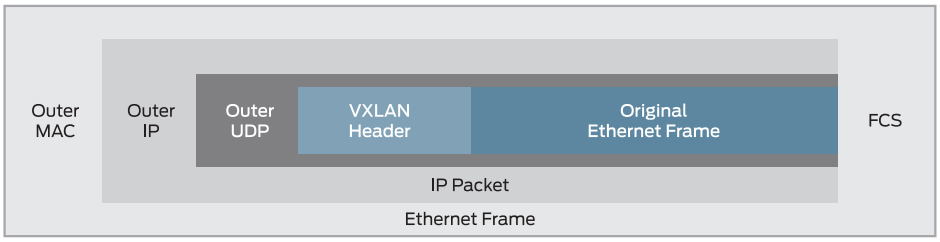
- Outer MAC destination address (MAC address of the tunnel endpoint VTEP)
- Outer MAC source address (MAC address of the tunnel source VTEP)
- Outer IP destination address (IP address of the tunnel endpoint VTEP)

- Outer IP source address (IP address of the tunnel source VTEP)
- Outer UDP header
- A VXLAN header that includes a 24-bit field—called the VXLAN network identifier (VNI)—that is used to uniquely identify the VXLAN. The VNI is similar to a VLAN ID, but having 24 bits allows you to create many more VXLANs than VLANs.

NOTE: Because VXLAN adds 50 to 54 bytes of additional header information to the original Ethernet frame, you might want to increase the MTU of the underlying network. In this case, configure the MTU of the physical interfaces that participate in the VXLAN network, not the MTU of the logical VTEP source interface, which is ignored.

Figure 1 on page 5 shows the VXLAN packet format.

Figure 1: VXLAN Packet Format



VXLAN Implementation Methods

Junos OS supports implementing VXLANs in the following environments:

- Without SDN controllers. VXLANs implemented in this type of environment are known as manual VXLANs.
- With SDN controllers. In this environment, SDN controllers use the Open vSwitch Database (OVSDB) management protocol to provide a means through which controllers (such as a VMware NSX or Juniper Contrail controller) and Juniper Networks devices that support OVSDB can communicate.

Using a QFX5100 Switch with VXLANs

You can configure a QFX5100 switch to perform all of the following roles:

- In an environment without an SDN controller, act as a transit Layer 3 switch for downstream hosts acting as VTEPs. In this configuration, you do not need to configure any VXLAN functionality on the switch. You do need to configure IGMP and PIM so that the switch can form the multicast trees for the VXLAN multicast groups. (See Manual VXLANs Require PIM on page 7 for more information.)
- In an environment with or without an SDN controller, act as a Layer 2 gateway between virtualized and non-virtualized networks in the same data center or between data

centers. For example, you can use a QFX5100 switch to connect a network that uses VXLANs to one that uses VLANs.

- Act as a Layer 2 gateway between virtualized networks in the same or different data centers and allow virtual machines to move (VMotion) between those networks and data centers. For example, if you want to allow VMotion between devices in two different networks, you can create the same VLAN in both networks and put both devices on that VLAN. The QFX5100 switches connected to these devices, acting as VTEPs, can map that VLAN to the same VXLAN, and the VXLAN traffic can then be routed between the two networks.

NOTE: A QFX 5100 switch cannot route traffic between different VXLANs. To connect devices in different VXLANs you need a VXLAN-capable Layer 3 gateway, such as a Juniper Networks MX Series router.

Because the additional headers add 50 to 54 bytes, you might need to increase the MTU on a QFX5100 VTEP to accommodate larger packets. For example, if the switch is using the default MTU value of 1514 bytes and you want to forward 1500-byte packets over the VXLAN, you need to increase the MTU to allow for the increased packet size caused by the additional headers.

Changing the UDP Port on a QFX5100 Switch

Starting with Junos OS 14.1X53-D25, you can configure the UDP port used as the destination port for VXLAN traffic on a QFX5100 switch. To configure the VXLAN destination port to be something other than the default UDP port of 4789, enter

set protocols l2-learning destination-udp-port *port-number*

The port you configure will be used for all VXLANs configured on the switch.

NOTE: If you make this change on one switch in a VXLAN, you must make the same change on all the devices that terminate the VXLANs configured on your switch. If you do not do so, traffic will be disrupted for all the VXLANs configured on your switch. When you change the UDP port, the previously learned remote VTEPs and remote MACs are lost and VXLAN traffic is disrupted until the switch relearns the remote VTEPs and remote MACs.

Controlling Transit Multicast Traffic on a QFX5100 Switch

When a QFX5100 switch acting as a VXLAN VTEP receives a broadcast, unknown unicast, or multicast packet, it performs the following actions on the packet:

1. It de-encapsulates the packet and delivers it to the locally attached hosts.
2. It then adds the VXLAN encapsulation again and sends the packet to the other VTEPs in the VXLAN.

These actions are performed by the loopback interface used as the VXLAN tunnel address and can therefore negatively impact the bandwidth available to the VTEP. With Junos OS 14.1X53-D30 and later, if you know that there are no multicast receivers attached to other VTEPs in the VXLAN who want traffic for a specific multicast group, you can reduce the processing load on the loopback interface by entering the following statement:

set protocols l2-learning disable-vxlan-multicast-transit vxlan-multicast-group *multicast-group*

In this case, no traffic will be forwarded for the specified group but all other multicast traffic will be forwarded. If you do not want to forward any multicast traffic to other VTEPs in the VXLAN, enter the following statement:

set protocols l2-learning disable-vxlan-multicast-transit vxlan-multicast-group all

Using an MX Series Router or EX9200 Switch as a VTEP

You can configure an MX Series router or EX9200 switch to act as a VTEP and perform all of the following roles:

- Act as a Layer 2 gateway between virtualized and non-virtualized networks in the same data center or between data centers. For example, you can use an MX Series router to connect a network that uses VXLANs to one that uses VLANs.
- Act as a Layer 2 gateway between virtualized networks in the same or different data centers and allow virtual machines to move (VMotion) between those networks and data centers.
- Act as a Layer 3 gateway to route traffic between different VXLANs in the same data center.
- Act as a Layer 3 gateway to route traffic between different VXLANs in different data centers over a WAN or the Internet using standard routing protocols or VPLS tunnels.

NOTE: If you want an MX Series router or EX9200 switch to be a VXLAN Layer 3 gateway, you must configure integrated routing and bridging (IRB) interfaces to connect the VXLANs, just as you do if you want to route traffic between VLANs.

Manual VXLANs Require PIM

In an environment with a controller (such as VMware's NSX), you can provision VXLANs on a Juniper Networks device. A controller also provides a control plane that VTEPs use to advertise their reachability and learn about the reachability of other VTEPs. You can also manually create VXLANs on Juniper Networks devices instead of using a controller. If you use this approach, you must also configure PIM on the VTEPs so that they can create VXLAN tunnels between themselves.

You must also configure each VTEP in a given VXLAN to be a member of the same multicast group. (If possible, you should assign a different multicast group address to each VXLAN, though this is not required. Multiple VXLANs can share the same multicast group.) The VTEPs can then forward ARP requests they receive from their connected hosts to the multicast group. The other VTEPs in the group de-encapsulate the VXLAN

information, and (assuming they are members of the same VXLAN) they forward the ARP request to their connected hosts. When the target host receives the ARP request, it responds with its MAC address, and its VTEP forwards this ARP reply back to the source VTEP. Through this process, the VTEPs learn the IP addresses of the other VTEPs in the VXLAN and the MAC addresses of the hosts connected to the other VTEPs.

The multicast groups and trees are also used to forward broadcast, unknown unicast, and multicast (BUM) traffic between VTEPs. This prevents BUM traffic from being unnecessarily flooded outside the VXLAN.

NOTE: Multicast traffic that is forwarded through a VXLAN tunnel is sent only to the remote VTEPs in the VXLAN. That is, the encapsulating VTEP does not copy and send copies of the packets according to the multicast tree—it only forwards the received multicast packets to the remote VTEPs. The remote VTEPs de-encapsulate the encapsulated multicast packets and forward them the appropriate Layer 2 interfaces. The remote VTEPs also do not copy and send copies of the packets according to the multicast tree.

Load Balancing VXLAN Traffic

On QFX5100 switches, the Layer 3 routes that form VXLAN tunnels use per-packet load balancing by default, which means that load balancing is implemented if there are ECMP paths to the remote VTEP. This is different from normal routing behavior in which per-packet load balancing is not used by default. (Normal routing uses per-prefix load balancing by default.)

The source port field in the UDP header is used to enable ECMP load balancing of the VXLAN traffic in the Layer 3 network. This field is set to a hash of the inner packet fields, which results in a variable that ECMP can use to distinguish between tunnels (flows). (None of the other fields that flow-based ECMP normally uses are suitable for use with VXLANs. All tunnels between the same two VTEPs have the same outer source and destination IP addresses, and the UDP destination port is set to port 4789 by definition. Therefore, none of these fields provide a sufficient way for ECMP to differentiate flows.)

Using ping and traceroute With a VXLAN

On a QFX5100 switch, you can use the **ping** and **traceroute** commands to troubleshoot traffic flow through a VXLAN tunnel by including the **overlay** parameter and various options. You use these options to force the **ping** or **traceroute** packets to follow the same path as data packets through the VXLAN tunnel. In other words, you make the underlay packets (**ping** and **traceroute**) take the same route as the overlay packets (data traffic). See *ping overlay* and *traceroute overlay* for more information.

VXLAN Constraints on QFX5100 Switches

When configuring VXLANs on QFX5100 switches, be aware of the constraints in the following list. In this list, “Layer 3 side” refers to a network-facing interface that performs VXLAN encapsulation and de-encapsulation, and “Layer 2 side” refers to a server-facing interface that is a member of a VLAN that is mapped to a VXLAN.

- You can use VXLANs on a Virtual Chassis or Virtual Chassis Fabric if all of the members are QFX5100 switches. You cannot use VXLANs if any of the members is not a QFX5100 switch.
- VXLAN configuration is supported only in the default routing instance.
- A QFX5100 switch cannot route traffic between different VXLANs.
- A physical interface cannot be a member of a VLAN and a VXLAN. That is, an interface that performs VXLAN encapsulation and de-encapsulation cannot also be a member of a VLAN. For example, if a VLAN that is a mapped to a VXLAN is a member of trunk port xe-0/0/0, any other VLAN that is a member of xe-0/0/0 must also be assigned to a VXLAN.
- Multichassis link aggregation groups (MC-LAGS) are not supported with VXLAN.
- IP fragmentation and defragmentation are not supported on the Layer 3 side.
- The following features are not supported on the Layer 2 side:
  - STP (any variant)
  - IGMP snooping
- Access port security features are not supported with VXLAN. For example, the following features are not supported:
  - DHCP snooping
  - dynamic ARP inspection
  - MAC limiting and MAC move limiting
- See the following to determine whether ingress node replication is supported on QFX switches:
  - PIM used for control plane: ingress node replication is not supported.
  - Control plane provided by a controller: ingress node replication is not supported.
  - EVPN used with VXLANs: ingress node replication is supported.
- PIM-BIDIR and PIM-SSM are not supported with VXLANs.
- Class of service (CoS) features are not supported with VXLANs.
- If you configure a port-mirroring instance to mirror traffic egressing from an interface that performs VXLAN encapsulation, the source and destination MAC addresses of the mirrored packets are invalid. The original VXLAN traffic is not affected.

Related Documentation

- *Examples: Manually Configuring VXLANs on QFX Series Switches*
- *Example: Manually Configuring VXLANs on MX Series Routers*
- *OVSDB Support on Juniper Networks Devices*
- *mtu*

VXLAN Constraints on QFX5100 Switches

When configuring VXLANs on QFX5100 switches, be aware of the constraints in the following list. In this list, “Layer 3 side” refers to a network-facing interface that performs VXLAN encapsulation and de-encapsulation, and “Layer 2 side” refers to a server-facing interface that is a member of a VLAN that is mapped to a VXLAN.

- You can use VXLANs on a Virtual Chassis or Virtual Chassis Fabric if all of the members are QFX5100 switches. You cannot use VXLANs if any of the members is not a QFX5100 switch.
- VXLAN configuration is supported only in the default routing instance.
- A QFX5100 switch cannot route traffic between different VXLANs.
- A physical interface cannot be a member of a VLAN and a VXLAN. That is, an interface that performs VXLAN encapsulation and de-encapsulation cannot also be a member of a VLAN. For example, if a VLAN that is a mapped to a VXLAN is a member of trunk port xe-0/0/0, any other VLAN that is a member of xe-0/0/0 must also be assigned to a VXLAN.
- Multichassis link aggregation groups (MC-LAGS) are not supported with VXLAN.
- IP fragmentation and defragmentation are not supported on the Layer 3 side.
- The following features are not supported on the Layer 2 side:
  - STP (any variant)
  - IGMP snooping
- Access port security features are not supported with VXLAN. For example, the following features are not supported:
  - DHCP snooping
  - dynamic ARP inspection
  - MAC limiting and MAC move limiting
- See the following to determine whether ingress node replication is supported on QFX switches:
  - PIM used for control plane: ingress node replication is not supported.
  - Control plane provided by a controller: ingress node replication is not supported.
  - EVPN used with VXLANs: ingress node replication is supported.
- PIM-BIDIR and PIM-SSM are not supported with VXLANs.
- Class of service (CoS) features are not supported with VXLANs.
- If you configure a port-mirroring instance to mirror traffic egressing from an interface that performs VXLAN encapsulation, the source and destination MAC addresses of the mirrored packets are invalid. The original VXLAN traffic is not affected.

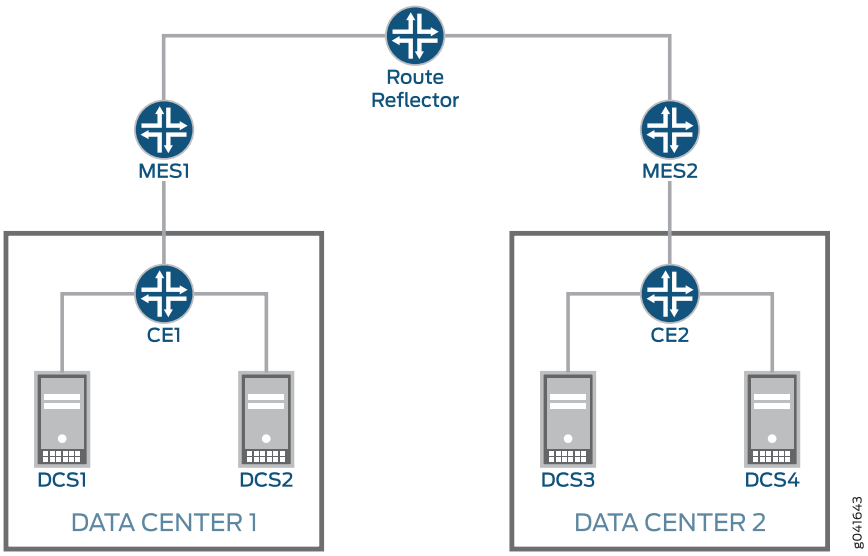
Related Documentation

- Understanding VXLANs on page 3
- *Examples: Manually Configuring VXLANs on QFX Series Switches*
- *Configuring VXLANs on a QFX5100 Switch*

EVPN Overview

An Ethernet VPN (EVPN) enables you to connect dispersed customer sites using a Layer 2 virtual bridge. As with other types of VPNs, an EVPN consists of customer edge (CE) devices (host, router, or switch) connected to provider edge (PE) routers. The PE routers can include an MPLS edge switch (MES) that acts at the edge of the MPLS infrastructure. Either an MX Series 3D Universal Edge Router or a standalone EX9200 switch (Junos OS Release 14.2) can be configured to act as an MES. You can deploy multiple EVPNs within a service provider network, each providing network connectivity to a customer while ensuring that the traffic sharing on that network remains private. Figure 2 on page 11 illustrates a typical EVPN deployment. Traffic from Data Center 1 is transported over the service provider's network through MES1 to MES2 and then onto Data Center 2. DCS1, DCS2, DCS3, and DCS4 are the data center switches.

Figure 2: EVPN Connecting Data Center 1 and Data Center 2



The MESs are interconnected within the service provider's network using label-switched paths (LSPs). The MPLS infrastructure allows you to take advantage of the MPLS functionality provided by the Junos OS, including fast reroute, node and link protection, and standby secondary paths. For EVPNs, learning between MESs takes place in the control plane rather than in the data plane (as is the case with traditional network bridging). The control plane provides greater control over the learning process, allowing you to restrict which devices discover information about the network. You can also apply policies on the MESs, allowing you to carefully control how network information is distributed and processed. EVPNs utilize the BGP control plane infrastructure, providing greater scale and the ability to isolate groups of devices (hosts, servers, virtual machines, and so on) from each other.

The MESs attach an MPLS label to each MAC address learned from the CE devices. This label and MAC address combination is advertised to the other MESs in the control plane. Control plane learning enables load balancing and improves convergence times in the event of certain types of network failures. The learning process between the MESs and the CE devices is completed using the method best suited to each CE device (data plane learning, IEEE 802.1, LLDP, 802.1aq, and so on).

The policy attributes of an EVPN are similar to an IP VPN (for example, Layer 3 VPNs). Each EVPN routing instance requires that you configure a route distinguisher (RD) and one or more route targets (RTs). A CE device attaches to an EVPN routing instance on an MES through an Ethernet interface that might be configured for one or more VLANs.

The following features are available for EVPNs:

- Ethernet connectivity between data centers spanning metropolitan area networks (MANs) and WANs
- One VLAN for each MAC VPN
- Automatic RDs
- Dual-homed EVPN connection with active/standby multihoming

The following features are not supported for EVPNs:

- Graceful restart, graceful Routing Engine switchover (GRES), and nonstop active routing (NSR)
- Active/active multihoming

Related Documentation

- Supported EVPN Standards on page 12
- EVPN Overview on page 11

Supported EVPN Standards

Junos OS supports the following RFCs and Internet drafts that define standards for EVPNs:

- RFC 4364, *BGP/MPLS IP Virtual Private Networks (VPNs)*
- RFC 4761, *Virtual Private LAN Service (VPLS) Using BGP for Auto-Discovery and Signaling*
- RFC 7432, *BGP MPLS-Based Ethernet VPN*

  The following features are not supported:

  - Automatic derivation of Ethernet segment (ES) values. Only static ES configurations are supported.
  - Host proxy ARP.
  - MAC mobility extended community.
  - VLAN bundle service interface.

Related Documentation

- EVPN Overview on page 11
- *Accessing Standards Documents on the Internet*

EVPN with VXLAN Data Plane Encapsulation

Ethernet VPNs (EVPNs) enable you to connect groups of dispersed customer sites using Layer 2 virtual bridges, and Virtual Extensible LANs (VXLANs) allow you to stretch Layer 2 connection over an intervening Layer 3 network, while providing network segmentation like a VLAN, but without the scaling limitation of traditional VLANs. EVPN with VXLAN encapsulation handles Layer 2 connectivity at the scale required by cloud server providers and replaces limiting protocols like Spanning Tree Protocol (STP), freeing up your Layer 3 network to use more robust routing protocols. EVPN with VXLAN data plane encapsulation can be used with and without Juniper Networks Contrail virtualization software—use Contrail with EVPN VXLAN data plane encapsulation when you have an environment that includes both virtual and bare-metal devices.

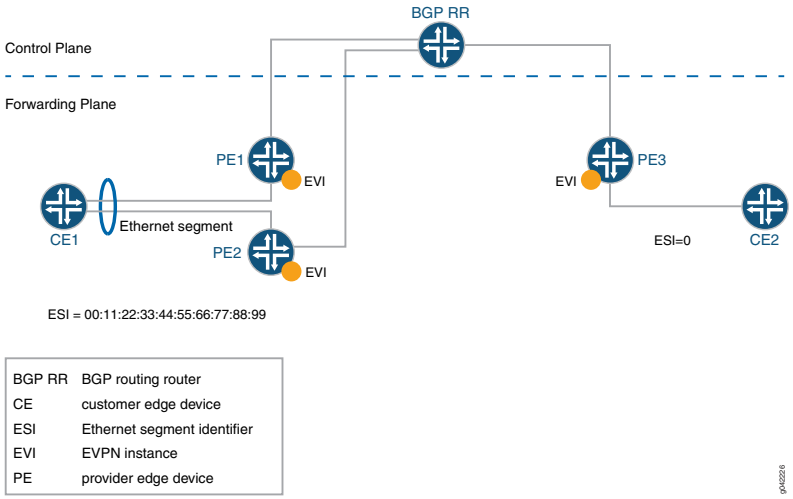
NOTE: **MX Series Routers interoperate with QFX5100 Series Switches and QFX5100 Virtual Chassis to provide full EVPN with VXLAN functionality.**

- Understanding EVPN on page 13
- Understanding VXLAN on page 15
- VXLAN-EVPN Integration Overview on page 16
- Understanding Contrail Virtual Networks Use with EVPN VXLAN on page 17
- VXLAN-EVPN Packet Format on page 18
- Understanding Contrail Virtual Networks Use with EVPN-VXLAN on page 18
- Q & A on page 19

Understanding EVPN

Ethernet VPN (EVPN) is a new standards-based technology that provides virtual multipoint bridged connectivity between different domains over an IP or IP/MPLS backbone network. Like other VPN technologies, such as IP VPN and virtual private LAN service (VPLS), EVPN instances (EVIs) are configured on provider edge (PE) routers to maintain logical service separation between customers. The PE routers connect to customer edge (CE) devices, which can be routers, switches, or hosts. The PE routers then exchange reachability information using Multiprotocol BGP (MP-BGP) and encapsulated traffic is forwarded between PE routers. Because elements of the architecture are common with other VPN technologies, EVPN can be seamlessly introduced and integrated into existing service environments as shown in Figure 3 on page 14.

Figure 3: EVPN Overview



EVPN technology offers multi-tenancy, flexible services that can be extended on demand, frequently using compute resources of different physical data centers. In addition, EVPN uses these techniques:

- *Multihoming* provides redundancy in the event that an access link or one of the PE routers fails. In either case traffic flows from the CE device toward the PE router, using the remaining active links. For traffic in the other direction, the remote PE router updates its forwarding table to send traffic to the remaining active PE routers connected to the multihomed Ethernet segment. EVPN provides a fast convergence mechanism, which reduces traffic restoration time so that the time it takes to make this adjustment is independent of the number of MAC addresses learned by the PE router. All-active multihoming allows a CE device to connect to two or more PE routers such that traffic is forwarded using all of the links between the devices. This enables the CE device to load-balance traffic to multiple PE routers. More importantly it allows a remote PE router to load-balance traffic to the multihomed PE routers across the core network. This load balancing of traffic flows between data centers is known as aliasing, which is an effect that causes different signals to become indistinguishable—they become aliases of one another. Aliasing is used with digital audio and digital images.
- *Split horizon* prevents the looping of broadcast, unknown unicast, and multicast (BUM) traffic in a network. The split horizon basic principle is simple: Information about the routing for a particular packet is never sent back in the direction from which it was received.
- *Local link bias* conserves bandwidth by using local links to forward unicast traffic exiting a Virtual Chassis or Virtual Chassis Fabric (VCF) that has a Link Aggregation group (LAG) bundle composed of member links on different member switches in the same Virtual Chassis or VCF. A local link is a member link in the LAG bundle that is on the member switch that received the traffic.
- *VM motion*, part of EVPN's MP-BGP control plane, allows live virtual machines to be dynamically moved from one data center to another.

The EVPN technology, similar to Layer 3 MPLS VPN, is a technology that introduces the concept of routing MAC addresses using MP-BGP over MPLS core. Some of the important benefits of using EVPNs include:

- Ability to have an active multihomed edge device
- Load balancing across dual-active links
- MAC address mobility
- Multitenancy
- Aliasing
- Fast convergence

NOTE: MPLS core is not supported on QFX5100 switches in Junos OS Release 14.1X_53-D30. Only MX Series routers support this feature.

Understanding VXLAN

Network overlays are created by encapsulating traffic and tunneling the traffic over a physical network. A number of tunneling protocols can be used in the data center to create network overlays—the most common protocol is Virtual Extensible LAN (VXLAN). VXLAN tunneling protocol encapsulates Layer 2 Ethernet frames in Layer 3 UDP packets. This encapsulation enables you to create virtual Layer 2 subnets or segments that can span physical Layer 3 networks.

In a VXLAN overlay network, each Layer 2 subnet or segment is uniquely identified by a VXLAN virtual network identifier (VNI). A VNI segments traffic the same way that an IEEE 802.1Q VLAN ID segments traffic. As is the case with VLAN, virtual machines on the same VNI can communicate directly with each other, whereas virtual machines on different VNIs need a router to communicate with each other.

The entity that performs the encapsulation and decapsulation is called a VXLAN tunnel endpoint, or VTEP for short.

VTEPs typically reside in hypervisor hosts, such as ESXi or KVM hosts. Each VTEP has two interfaces. One is a switching interface that faces the virtual machines in the host and provides communication between VMs on the local LAN segment. The other is an IP interface that faces the Layer 3 network. Each VTEP has a unique IP address which is used for routing the UDP packets between VTEPs. In Figure 2, when VTEP1 receives an Ethernet frame from VM1 addressed to VM3, it uses the VNI and the destination MAC to look up in its forwarding table which VTEP to send the packet to. It then adds a VXLAN header that contains the VNI to the Ethernet frame and encapsulates the frame in a Layer 3 UDP packet and routes the packet to VTEP2 over the Layer 3 network. VTEP2 de-encapsulates the original Ethernet frame and forwards it to VM3. VM1 and VM3 cannot detect the VXLAN tunnel and the Layer 3 network between them.

VXLAN-EVPN Integration Overview

VXLAN defines a tunneling scheme to overlay Layer 2 networks on top of Layer 3 networks. It allows for optimal forwarding of Ethernet frames with support for multipathing of unicast and multicast traffic with the use of UDP/IP encapsulation for tunneling, and is mainly used for the intra-data center site connectivity.

A unique characteristic of EVPN is that MAC address learning between PE routers occurs in the control plane. A new MAC address detected from a CE device is advertised by the local PE, using MP-BGP, to all the remote PE routers. This method differs from existing Layer 2 VPN solutions such as VPLS, which learn by flooding unknown unicast in the data plane. This control plane based MAC learning method is the key enabler of the many useful features provided by EVPN.

Because MAC learning is handled in the control plane, EVPN has the flexibility to support different data plane encapsulation technologies between PE routers. This is important because not every backbone network might be running MPLS, especially in Enterprise networks.

There is a lot of interest in EVPN today because it addresses many of the challenges faced by network operators building data centers to offer cloud and virtualization services. The main application of EVPN is Data Center Interconnect (DCI), the ability to extend Layer 2 connectivity between different data centers that are deployed to improve the performance of delivering application traffic to end users and for disaster recovery.

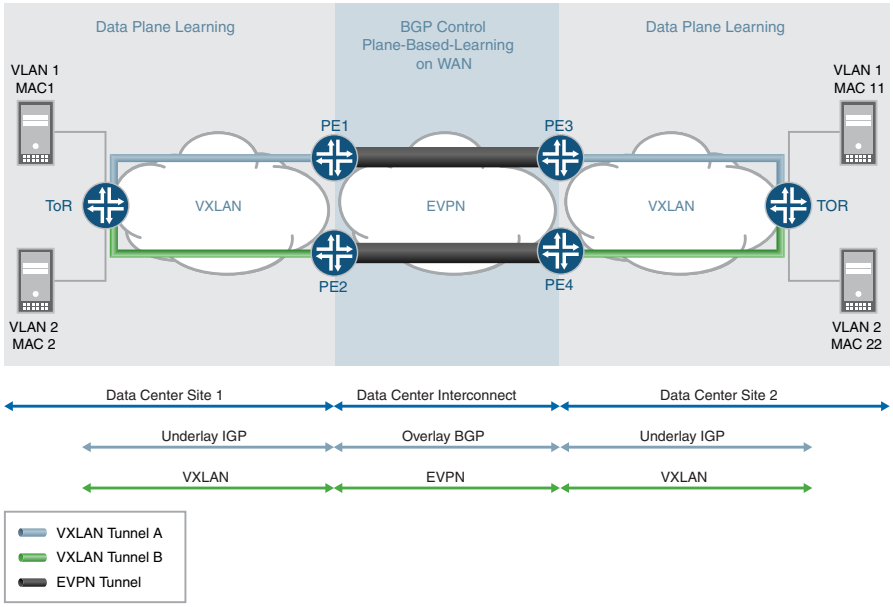
Although there are various DCI technologies available, EVPN has an added advantage over the other MPLS technologies because of its unique features, such as active/active redundancy, aliasing, and mass MAC withdrawal. As a result, to provide a solution for DCI, VXLAN is integrated with EVPN.

Every VXLAN network, which is connected to the MPLS or IP core, runs an independent instance of the IGP control plane. Each PE router participates in the IGP control plane instance of its VXLAN network. Here, each customer is a data center, so each has its own virtual router for VXLAN underlay.

Each PE node can terminate the VXLAN data plane encapsulation where the VXLAN virtual network identifier (VNI) or VSID is mapped to a bridge domain. The PE router performs data plane learning on the traffic received from the VXLAN network.

Each PE node implements EVPN to distribute the client MAC addresses learned over the VXLAN tunnel into BGP. Each PE node encapsulates the VXLAN or Ethernet frames with MPLS when sending the packets over the MPLS core and with the VXLAN tunnel header when sending the packets over the VXLAN network.

Figure 4: VXLAN-EVPN Integration Overview



Understanding Contrail Virtual Networks Use with EVPN VXLAN

Juniper Networks Contrail virtualization software is an SDN solution that automates and orchestrates the creation of highly scalable virtual networks. These virtual networks enable you to harness the power of the cloud—for new services, increased business agility, and revenue growth. MX Series routers can use EVPN with VXLAN encapsulation to provide both Layer 2 and Layer 3 connectivity for end stations within a Contrail virtual network (VN).

Contrail network's software for virtual networks provides both Layer 2 and Layer 3 connectivity. With Contrail, Layer 3 routing is preferred over Layer 2 bridging whenever possible, hence the saying "route first then bridge." Layer 3 routing is used through VRFs between Contrail vRouters and physical MX Series routers. Juniper MX Series routers provide Layer 3 gateway functionality for inter-VN communication.

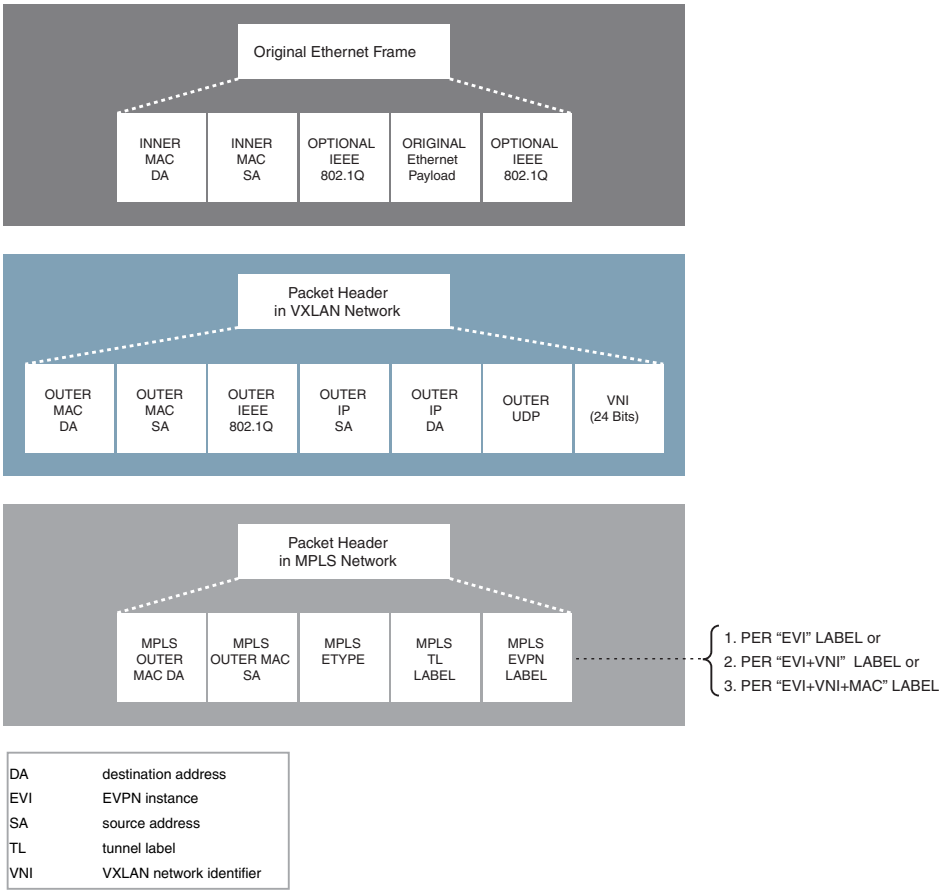
Contrail allows you to use EVPN VXLAN when your network includes both virtual and bare-metal devices. There are two types of encapsulation methods used in virtual networks. MPLS-over-GRE is used for Layer 3 routing between Contrail and MX Series routers. EVPN with VXLAN encapsulation is used for Layer 2 connectivity between virtual machines and QFX5100 within a Layer 2 domain where QFX5100 is physically connected to top-of-rack switches. For Layer 2 connectivity, traffic load balancing in the core is achieved using the multihoming all-active feature provided by EVPN. Each QFX5100 is also multihomed to top-of-rack switches.

For more information on using EVPN and VXLAN, see http://techwiki.juniper.net/Documentation/Contrail/Contrail_Controller_Feature_Guide/Configuration/30_Using_EVPN_and_VXLAN.

VXLAN-EVPN Packet Format

The VXLAN and EVPN packet format is shown in Figure 5 on page 18.

Figure 5: VXLAN-EVPN Packet Format



Understanding Contrail Virtual Networks Use with EVPN-VXLAN

Juniper Networks Contrail virtualization software is a software-defined networking (SDN) solution that automates and orchestrates the creation of highly scalable virtual networks. These virtual networks enable you to harness the power of the cloud—for new services, increased business agility, and revenue growth. MX Series routers can use EVPN-VXLAN to provide both Layer 2 and Layer 3 connectivity for end stations within a Contrail virtual network (VN).

NOTE: **QFX Series switches do not support Contrail.**

Contrail's software for virtual networks provides both Layer 2 and Layer 3 connectivity. With Contrail, Layer 3 routing is preferred over Layer 2 bridging whenever possible, hence the saying "route first then bridge." Layer 3 routing is used through VRFs between Contrail

vRouters and physical MX Series routers. MX Series routers provide Layer 3 gateway functionality for inter-VN communication.

Contrail allows you to use EVPN-VXLAN when your network includes both virtual and bare-metal devices. There are two types of encapsulation methods used in virtual networks. MPLS-over-GRE is used for Layer 3 routing between Contrail and MX Series routers. EVPN-VXLAN is used for Layer 2 connectivity between virtual machines and QFX Series switches within a Layer 2 domain where the QFX Series switch is physically connected to top-of-rack switches. For Layer 2 connectivity, traffic load balancing in the core is achieved using the multihoming all-active feature provided by EVPN. Each QFX switch is also multihomed to top-of-rack switches.

For details about Contrail support, see http://pathfinder.juniper.net/feature-explorer and then search on EVPN.

For more information on using EVPN-VXLAN, see Configuring EVPN and VXLAN.

Q & A

Q: What is the difference between EVPN/VXLAN and MPLS/EVPN?
A: With EVPN/VxLAN the host entries are created by L2 and there is no existing infrastructure method to export to L3. In the MPLS/EVPN solution host entries are created by RPD, are in the RIB and can be exported into a VRF or inet.0. Therefore, the ability to advertise /32 host routes is not there with EVPN/VxLAN , so the classic VMTO does not work with VxLAN. You can take advantage of the A/A L2 functionality with VxLAN.
Q: Can I simultaneously use EVPN-VXLAN with OVSDB-VXLAN on QFX switches?
A: No, you cannot mix EVPN-VXLAN with OVSDB-VXLAN. Once a switch is set to OVSDB-managed, the controller expects all ports to be OVSDB managed.

NOTE: **MPLS core is not supported on switches—only MX Series routers support this feature.**

Related Documentation

- EVPN Multihoming Overview on page 31
- Understanding VXLANs on page 3
- Implementing EVPN with VXLAN for Data Centers on page 19

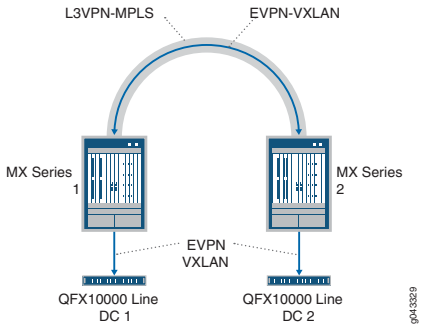
Implementing EVPN with VXLAN for Data Centers

Although there are various Data center interconnect (DCI) technologies available, EVPN has an added advantage over other MPLS technologies because of its unique features, such as active/active redundancy, aliasing, and mass MAC withdrawal. To provide a DCI solution, VXLAN is integrated with EVPN.

There are different options for using EVPN-VXLAN with DCI:

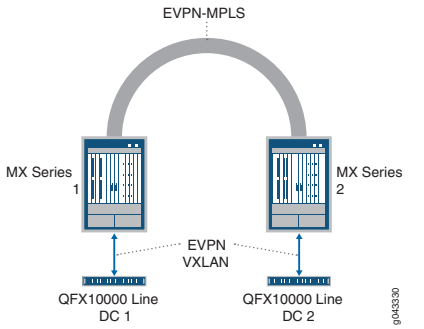
- Data center interconnect (DCI) can connect multiple data centers in your WAN using MX Series edge routers with a Layer 3 VPN MPLS network between them. QFX10000 switches start and stop the VXLAN tunnel. This option requires no changes to your WAN.

Figure 6: DCI Option: L3VPN-MPLS



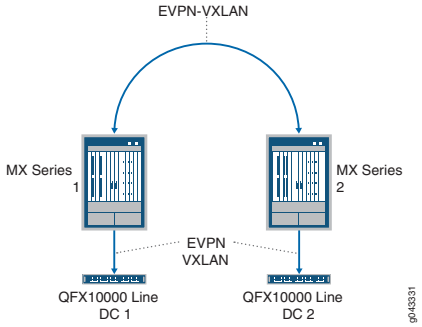
- A second option connects multiple data centers in your WAN using MX Series edge routers with an EVPN MPLS network between them. This option uses an EVPN dataplane and an MPLS control plane and requires changes to your WAN. You must change your LAN architecture to natively support EVPN and you must implement EVPN stitching between each MX router and the corresponding QFX10000 switch.

Figure 7: DCI Option: EVPN-MPLS



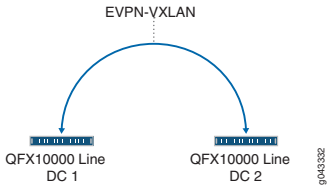
- You can also tunnel two branch locations across the Internet. In this case, implementation requires neither a traditional WAN nor MPLS. This method can use the Internet or an IP tunnel, where VXLAN rides on top of IP and EVPN is used throughout.

Figure 8: DCI Option: EVPN-VXLAN over the Internet



- If you do not have a branch router or a peering router, you can simply connect the data centers directly and EVPN is again used natively throughout. This implementation requires neither a traditional WAN nor MPLS, but you typically need a dark fiber connection.

Figure 9: DCI Option: L3VPN-MPLS Direct Connection



You can alternately create a EVPN VXLAN fabric internally in the data center using bare-metal servers and/or virtual servers and using OpenClos for management. Here you also use VXLAN L2 gateways and L3 gateways on switches such as QFX10000. The underlying fabric is built on BGP.

EVPN over VXLAN uses both routers and switches—the configurations are the same for both devices but they are located in different areas of the Junos OS CLI. MX routers are configured under a routing instance with the instance type **virtual switch**. QFX5100 switches are configured under global **switching-options** and global **protocol evpn**. See Table 3 on page 21 for a list of CLI commands used by EVPN over VXLAN.

Table 3: CLI Commands for EVPN Over VXLAN

Function	CLI Command
Specifies an identifier attached to a route. This enables you to distinguish to which VPN or VPLS the route belongs. Each routing instance must have a unique route distinguisher (RD) associated with it. The RD 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.	route-distinguisher
Specifies a VRF target community. 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. The options **import** and **export** apply to both routers and QFX5100 switches. The option **auto** applies to QFX5100 switches only.	vrf-target

Table 3: CLI Commands for EVPN Over VXLAN *(continued)*

Function	CLI Command
Specifies how routes are imported into the VRF table of the local PE router or switch from the remote PE router.	vrf-import
Specifies how routes are exported from the local PE router's VRF table to the remote PE router.	vrf-export
A designated forwarder (DF) is required when CEs are multihomed to more than one PE. Without a designated forwarder, multihomed hosts would receive duplicate packets. Designated forwarders are chosen for an Ethernet segment identifier (ESI) based on type-4 route advertisements.	designated-forwarder-election-hold-time
Configures a logical link-layer encapsulation type.	encapsulation
Establishes which VXLAN virtual network identifiers (VNIs) will be part of the EVPN/VXLAN MP-BGP domain. There are different BUM replication options available in EVPN—using **extended-vni-list** forgoes a multicast underlay in favor of EVPN and VXLAN ingress replication.	extended-vni-list
You configure different route targets (RTs) for each VNI instance under vni-options.	vni-options (QFX5100 only)
Displays both imported EVPN routes and export/import EVPN routes for the default switch routing instances.	show route table
Displays results of the configuration commands extended-vni-list and vni-options.	show configuration protocols evpn

Related Documentation

- EVPN with VXLAN Data Plane Encapsulation on page 13

EVPN-VXLAN Support of Virtual Chassis and Virtual Chassis Fabric

Ethernet VPN (EVPN) supports multihoming active-active mode, which enables a host to be connected to two leaf devices through a Layer 2 link aggregation group (LAG) interface. Before Junos OS Release 14.1X53-D40, the two leaf devices had to be QFX5100 standalone switches. Starting with 14.1X53-D40, the two leaf devices can be QFX5100 standalone switches, QFX5100 switches configured as a Virtual Chassis, QFX5100 switches configured as a Virtual Chassis Fabric (VCF), or a mix of the three options.

Regardless of whether a leaf device is a standalone QFX5100 switch or configured as a Virtual Chassis or VCF, on each leaf device, the LAG interface is configured with the same Ethernet segment identifier (ESI) for the host. The two leaf devices on which the same ESI is configured are peers to each other.

Implementing multihoming active-active mode in an EVPN-VXLAN topology with Virtual Chassis or VCFs has implications for the sending and receiving of Layer 2 broadcast, unknown unicast, and multicast (BUM) traffic.

Through the control of the LAG interface, only one copy of a BUM packet is sent from a host, for example, Host 1, to one of the leaf devices to which Host 1 is connected.

This topic includes the following use cases that describe how multihomed hosts receive Layer 2 BUM packets:

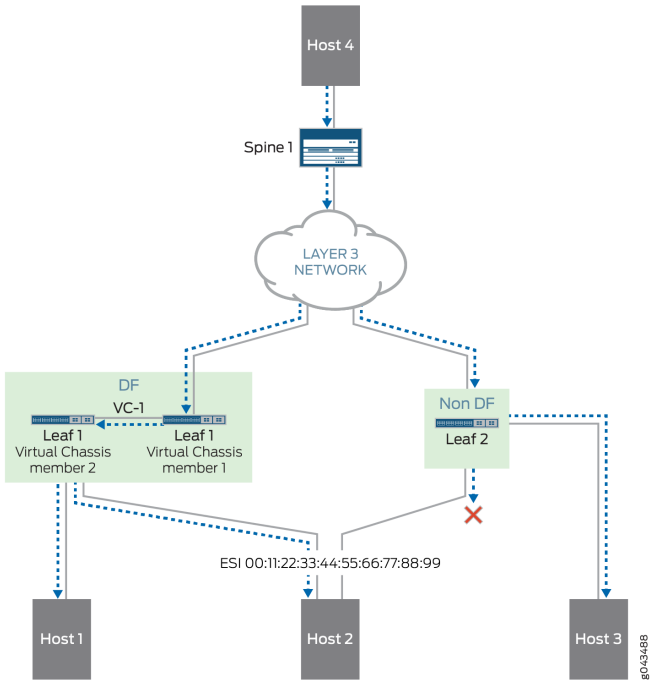
- Use Case 1: Multihomed Host Receives BUM Packets Only from Designated Forwarder (Virtual Chassis) on page 23
- Use Case 2: Multihomed Host Receives BUM Packets Only from Designated Forwarder (VCF) on page 25
- Use Case 3: Local Bias Prevents Multihomed Host from Receiving Same BUM Traffic that It Originates (Virtual Chassis and VCF) on page 27
- Use Case 4: Local Bias Prevents Multihomed Host from Receiving BUM Packets from Both Leaf Device (Virtual Chassis and VCF) on page 29

Use Case 1: Multihomed Host Receives BUM Packets Only from Designated Forwarder (Virtual Chassis)

In the EVPN-VXLAN topology shown in Figure 10 on page 24, there are four hosts connected to spine or leaf devices through a Layer 2 connection. All four hosts are in the same VXLAN. The spine and leaf devices function as virtual tunnel endpoints (VTEPs) that encapsulate and de-encapsulate all Layer 2 packets, including BUM packets, with a VXLAN header.

In this topology, there are two leaf devices. Leaf 1 is a Virtual Chassis that includes QFX5100 switches known as Virtual Chassis members 1 and 2, and Leaf 2 is a standalone QFX5100 switch. In this topology, Host 2 is connected to the two leaf devices through a Layer 2 LAG interface. The ESI for this interface is 00:11:22:33:44:55:66:77:88:99. Per multihoming active-active mode, the two leaf devices are peers, and one of the leaf devices, for example, Leaf 1, is elected as a designated forwarder (DF).

Figure 10: Multihomed Host 2 Receives BUM Packets Only from Leaf 1 (Virtual Chassis)

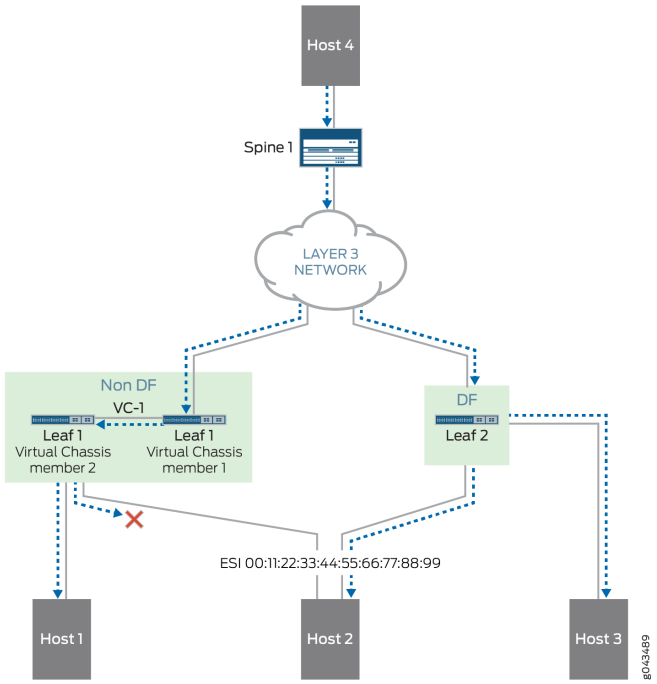


If Host 4 sends a BUM packet, the following packet flow occurs:

- Spine 1 encapsulates the Layer 2 packet with a VXLAN header and forwards the resulting Layer 3 packet to Leafs 1 and 2.
- Upon receipt of the Layer 3 packet, Leaf 1 de-encapsulates the packet and checks its MAC table to find the destination host. Since the packet is a BUM packet, the packet is to be forwarded to all hosts in the VXLAN. Hosts 1 and 2 are not included in the Virtual Chassis member 1, so it sends the BUM packet through the Virtual Chassis link to Virtual Chassis member 2. Virtual Chassis member 2 forwards the BUM packet to Host 1, and because Leaf 1 is the DF for ESI 00:11:22:33:44:55:66:77:88:99, Virtual Chassis member 2 forwards the packet to Host 2.
- Upon receipt of the Layer 3 packet, Leaf 2 de-encapsulates the packet and checks its MAC table to find the destination host. Since the packet is a BUM packet, the packet is to be forwarded to all hosts in the VXLAN. Therefore, Leaf 2 forwards the BUM packet to Host 3, and because Leaf 2 is not the DF for ESI 00:11:22:33:44:55:66:77:88:99, Leaf 2 drops the packet intended for Host 2.

Figure 11 on page 25 shows the same EVPN-VXLAN topology as in Figure 10 on page 24 except that in Figure 11 on page 25, Leaf 2 is the DF for ESI 00:11:22:33:44:55:66:77:88:99.

Figure 11: Multihomed Host 2 Receives BUM Packets Only from, Leaf 2 (Virtual Chassis)



Therefore, the packet flow for the topology in Figure 11 on page 25 is the same as that in Figure 10 on page 24 except for the following:

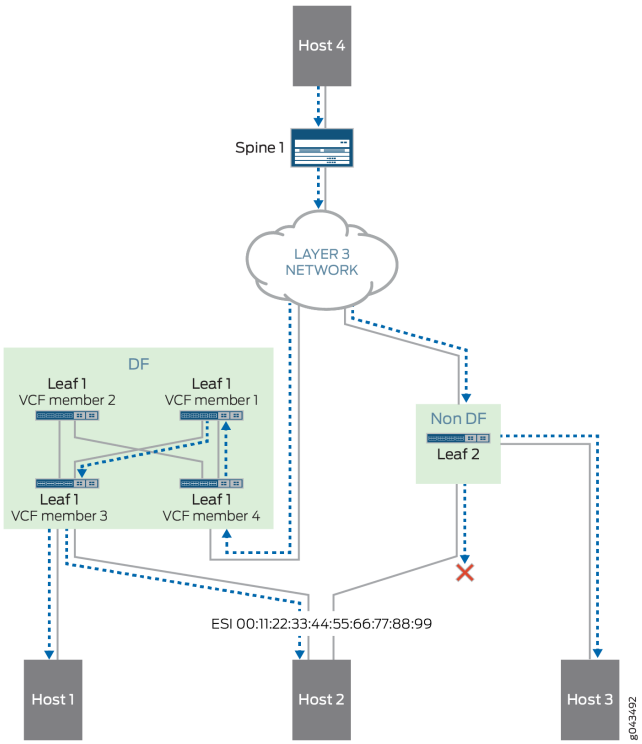
- Because Leaf 1 is not the DF for ESI 00:11:22:33:44:55:66:77:88:99, Leaf 1 drops the packet intended for Host 2.
- Because Leaf 2 is the DF for ESI 00:11:22:33:44:55:66:77:88:99, Leaf 2 forwards the packet to Host 2.

Use Case 2: Multihomed Host Receives BUM Packets Only from Designated Forwarder (VCF)

In the EVPN-VXLAN topology shown in Figure 12 on page 26, there are four hosts connected to spine or leaf devices through a Layer 2 connection. All four hosts are in the same VXLAN. The spine and leaf devices function as VTEPs that encapsulate and de-encapsulate all Layer 2 packets, including BUM packets, with a VXLAN header.

In this topology, there are two leaf devices. Leaf 1 is a VCF that includes QFX5100 switches known as VCF members 1 though 4; and Leaf 2 is a standalone QFX5100 switch. In this topology, Host 2 is connected to the two leaf devices through a Layer 2 LAG interface. The ESI for this interface is 00:11:22:33:44:55:66:77:88:99. Per multihoming active-active mode, the two leaf devices are peers, and one of the leaf devices, for example, Leaf 1, is elected as a DF.

Figure 12: Multihomed Host 2 Receives BUM Packets Only from Leaf 1 (VCF)

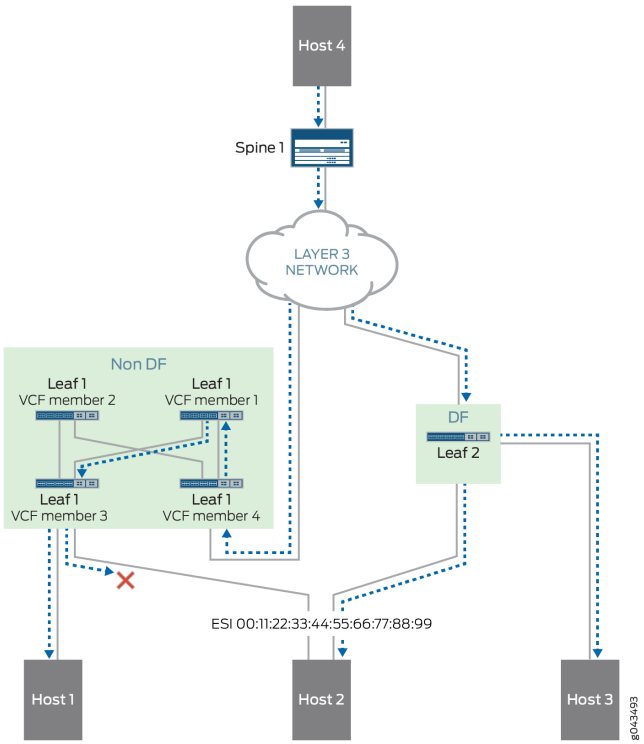


If Host 4 sends a BUM packet, the following packet flow occurs:

- Spine 1 encapsulates the Layer 2 packet with a VXLAN header and forwards the resulting Layer 3 packet to Leafs 1 and 2.
- Upon receipt of the Layer 3 packet, Leaf 1 de-encapsulates the packet and checks its MAC table to find the destination host. Since the packet is a BUM packet, the packet is to be forwarded to all hosts in the VXLAN. Hosts 1 and 2 are not included in VCF member 4, and although VCF member 4 can forward the BUM packet to either VCF member 1 or 2, in this case, it sends the packet through the VCF link to VCF member 1. VCF member 1 forwards the BUM packet to VCF member 3, which in turn forwards the packet to Host 1. Because Leaf 1 is the DF for ESI 00:11:22:33:44:55:66:77:88:99, Leaf 1 forwards the packet to Host 2.
- Upon receipt of the Layer 3 packet, Leaf 2 de-encapsulates the packet and checks its MAC table to find the destination host. Since the packet is a BUM packet, the packet is to be forwarded to all hosts in the VXLAN. Therefore, Leaf 2 forwards the BUM packet to Host 3, and because Leaf 2 is not the DF for ESI 00:11:22:33:44:55:66:77:88:99, Leaf 2 drops the packet intended for Host 2.

Figure 13 on page 27 shows the same EVPN-VXLAN topology as in Figure 12 on page 26 except that in Figure 13 on page 27, Leaf 2 is the DF for ESI 00:11:22:33:44:55:66:77:88:99.

Figure 13: Multihomed Host 2 Receives BUM Packets Only from Leaf 2 (Virtual Chassis Fabric)



The packet flow for the topology in Figure 13 on page 27 is the same as that in Figure 12 on page 26 except for the following:

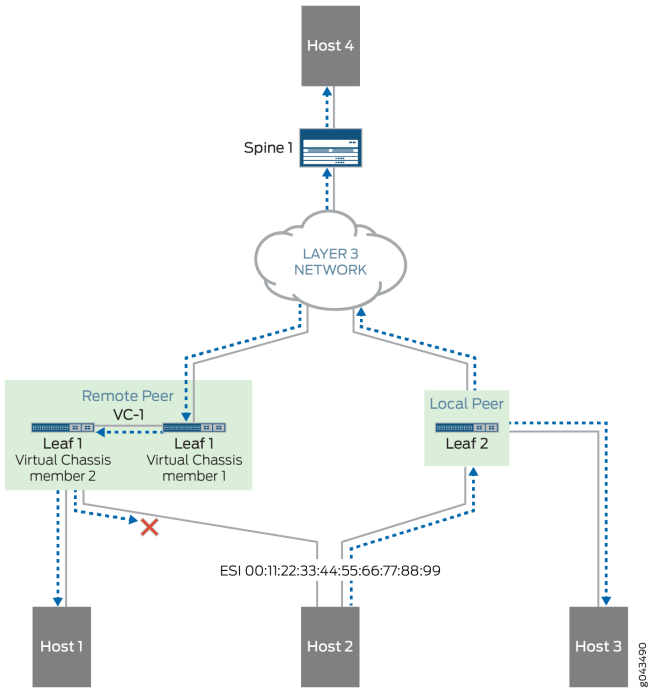
- Because Leaf 1 is not the DF for ESI 00:11:22:33:44:55:66:77:88:99, Leaf 1 drops the packet intended for Host 2.
- Because Leaf 2 is the DF for ESI 00:11:22:33:44:55:66:77:88:99, Leaf 2 forwards the packet to Host 2.

Use Case 3: Local Bias Prevents Multihomed Host from Receiving Same BUM Traffic that It Originates (Virtual Chassis and VCF)

In the EVPN-VXLAN topology shown in Figure 14 on page 28, there are four hosts connected to spine or leaf devices through a Layer 2 connection. All four hosts are in the same VXLAN. The spine and leaf devices function as VTEPs that encapsulate and de-encapsulate all Layer 2 packets, including BUM packets, with a VXLAN header.

In this topology, there are two leaf devices. Leaf 1 is a Virtual Chassis that includes QFX5100 switches known as Virtual Chassis members 1 and 2, and Leaf 2 is a standalone QFX5100 switch. In this topology, Host 2 is connected to the two leaf devices through a Layer 2 LAG interface. The ESI for this interface is 00:11:22:33:44:55:66:77:88:99. Per multihoming active-active mode, the two leaf devices are peers.

Figure 14: Local Bias Prevents Multihomed Host 2 from Receiving Same BUM Traffic that It Originates (Virtual Chassis)



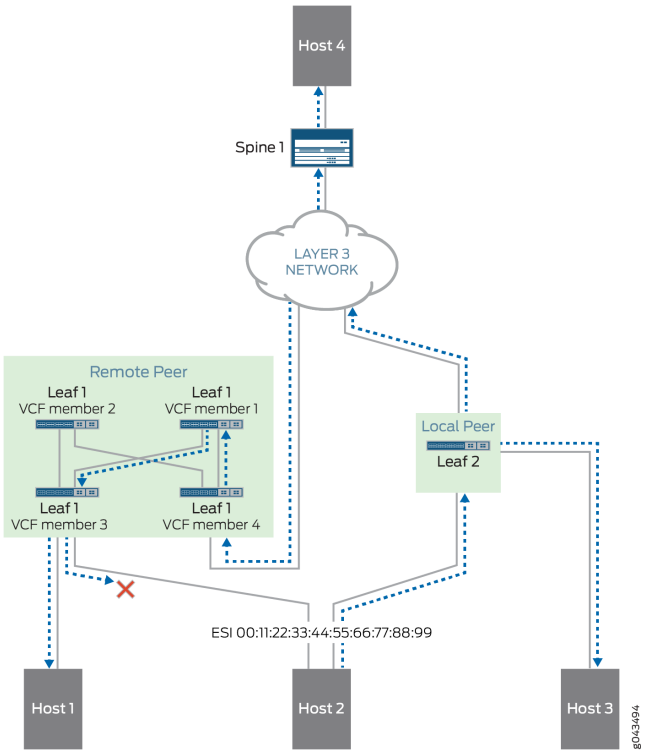
In this use case, Host 2 sends a BUM packet that must be forwarded to all hosts in the VXLAN. To prevent multihomed Host 2 from receiving the same BUM packet that it originated, there is a local bias, which determines that the local peer forwards the BUM packet and the remote peer drops it. This behavior of the local peer forwarding the packet occurs regardless of the DF status (DF or non-DF) of the local peer.

To illustrate the local bias, if Host 2 sends a BUM packet, the following packet flow occurs:

- Upon receipt of the Layer 2 BUM packet, Leaf 2 does the following:
  - Forwards the packet to Host 3.
  - Encapsulates the Layer 2 packet with a VXLAN header and forwards the resulting Layer 3 packet to Spine 1 and Leaf 1.
- Upon receipt of the Layer 3 packet, Leaf 1 de-encapsulates the packet and checks its MAC table to find the destination host. Since the packet is a BUM packet, the packet is to be forwarded to all hosts in the VXLAN. Hosts 1 and 2 are not included in Virtual Chassis member 1, so Virtual Chassis member 1 sends the BUM packet through the Virtual Chassis link to Virtual Chassis member 2. which in turn forwards the BUM packet to Host 1. However, since Leaf 1 is the remote peer to Leaf 2, which received the original BUM packet from Host 2, Leaf 1 drops the packet intended for Host 2.

Figure 15 on page 29 shows the same EVPN-VXLAN topology as in Figure 14 on page 28 except that in Figure 15 on page 29, Leaf 1 is configured as a VCF that includes QFX5100 switches that are known as VCF members 1 through 4.

Figure 15: Local Bias Prevents Multihomed Host 2 from Receiving Same BUM Traffic that It Originates (VCF)



As a result, the packet flow for the topology in Figure 15 on page 29 is the same as that for topology in Figure 14 on page 28 except for the following:

- Upon receipt of the Layer 3 packet, Leaf 1 de-encapsulates the packet and checks its MAC table to find the destination host. Hosts 1 and 2 are not included in VCF member 4, and although VCF member 4 can send the BUM packet to VCF member 1 or 2, in this case, it sends the packet to VCF member 1. VCF member 1 forwards the BUM packet to VCF member 3, which in turn forwards the packet to Host 1. However, since Leaf 1 is the remote peer to Leaf 2, which received the original BUM packet from Host 2, Leaf 1 drops the packet intended for Host 2.

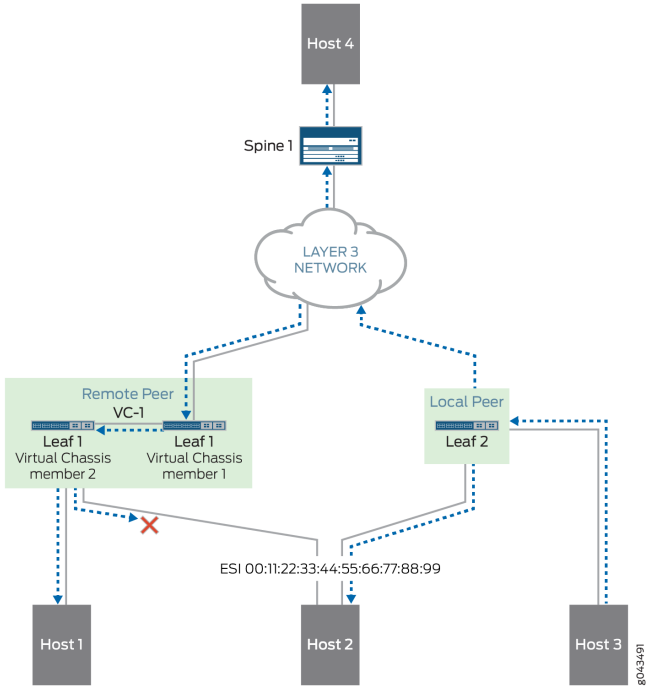
Use Case 4: Local Bias Prevents Multihomed Host from Receiving BUM Packets from Both Leaf Device (Virtual Chassis and VCF)

In the EVPN-VXLAN topology shown in Figure 16 on page 30, there are four hosts connected to spine or leaf devices through a Layer 2 connection. All four hosts are in the same VXLAN. The spine and leaf devices function as VTEPs that encapsulate and de-encapsulate all Layer 2 packets, including BUM packets, with a VXLAN header.

In this topology, there are two leaf devices. Leaf 1 is a Virtual Chassis that includes QFX5100 switches known as Virtual Chassis members 1 and 2, and Leaf 2 is a standalone QFX5100 switch. In this topology, Host 2 is connected to the two leaf devices through a

Layer 2 LAG interface. The ESI for this interface is 00:11:22:33:44:55:66:77:88:99. Per multihoming active-active mode, the two leaf devices are peers.

Figure 16: Local Bias Prevents Multihomed Host 2 from Receiving BUM Packets from Both Leaf Devices (Virtual Chassis)



In this use case, Host 3 sends a BUM packet that must be forwarded to all hosts in the VXLAN. To prevent multihomed Host 2 from receiving this BUM packet from both Leafs 1 and 2, there is a local bias, which determines that the local peer forwards the BUM packet and the remote peer drops it. This behavior of the local peer forwarding the packet occurs regardless of the DF status (DF or non-DF) of the local peer.

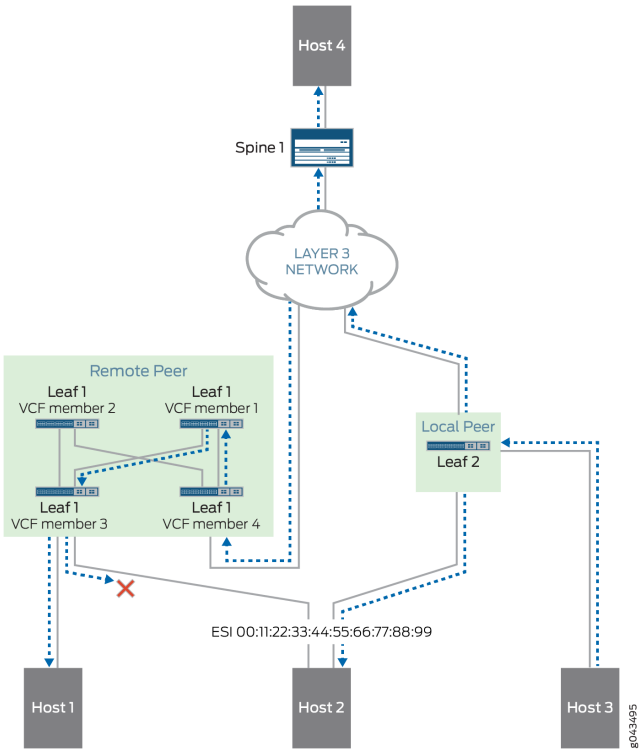
To illustrate the local bias, if Host 3 sends a BUM packet, the following packet flow occurs:

- Upon receipt of the BUM packet, Leaf 2 does the following:
  - Forwards the packet to Host 2. Since Host 3 is connected to the same leaf device (Leaf 2) as multihomed Host 2, Leaf 2 is considered the local peer, and according to the local bias, the local peer takes precedence over the remote peer, which in this case is Leaf 1.
  - Encapsulates the Layer 2 packet with a VXLAN header and forwards the resulting Layer 3 packet to Spine 1 and Leaf 1.
- Upon receipt of the Layer 3 packet, Leaf 1 de-encapsulates the packet and checks its MAC table to find the destination host. Hosts 1 and 2 are not included in Virtual Chassis member 1, so Virtual Chassis member 1 sends the BUM packet through the Virtual Chassis link to Virtual Chassis member 2. which in turn forwards the BUM packet to Host 1. However, when it comes to forwarding the packet to Host 2, Leaf 1, which is

considered to be the remote peer, drops the packet because the packet is forwarded by the local peer, which is Leaf 2.

Figure 17 on page 31 shows the same EVPN-VXLAN topology as in Figure 16 on page 30 except that in Figure 17 on page 31, Leaf 1 is configured as a VCF that includes QFX5100 switches that are known as VCF members 1 through 4.

Figure 17: Local Bias Prevents Multihomed Host 2 from Receiving BUM Packets from Both Leaf Devices (VCF)



As a result, the packet flow for the topology in Figure 17 on page 31 is the same as that for the topology in Figure 16 on page 30 except for the following:

- Upon receipt of the Layer 3 packet, Leaf 1 de-encapsulates the packet and checks its MAC table to find the destination host. Hosts 1 and 2 are not included in VCF member 4, and although VCF member 4 can send the BUM packet to either VCF members 1 or 2, in this case, VCF member 4 forwards the BUM packet to VCF member 1. VCF member 4 forwards the packet to VCF member 3, which in turn forwards the packet to Host 1. However, when it comes to forwarding the packet to Host 2, Leaf 1, which is considered to be the remote peer, drops the packet because the packet is forwarded by the local peer, which is Leaf 2.

EVPN Multihoming Overview

Introduction to EVPN Multihoming

An Ethernet VPN (EVPN) is comprised of customer edge (CE) devices that are connected to provider edge (PE) devices, which form the edge of the MPLS infrastructure. A CE device can be a host, a router, or a switch. The PE devices provide Layer 2 virtual bridge connectivity between the CE devices. There can be multiple EVPNs in the provider network. Learning between the PE routers occurs in the control plane using BGP, unlike traditional bridging, where learning occurs in the data plane.

The EVPN multihoming feature enables you to connect a customer site to two or more PE devices to provide redundant connectivity. A CE device can be multihomed to different PE devices or the same PE device. A redundant PE device can provide network service to the customer site as soon as a failure is detected. Thus, EVPN multihoming helps to maintain EVPN service and traffic forwarding to and from the multihomed site in the event of the following types of network failures:

- PE device to CE device link failure
- PE device failure
- MPLS-reachability failure between the local PE device and a remote PE device

Figure 18: CE Device Multihomed to Two PE Routers

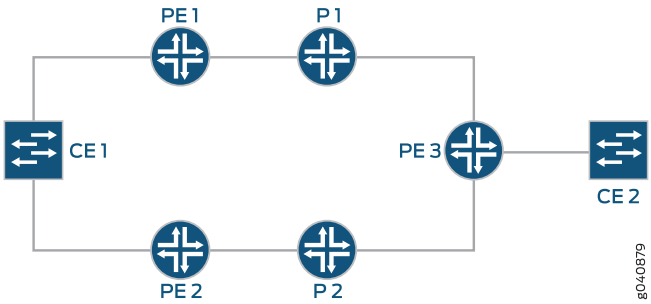


Figure 18 on page 32 illustrates how a CE device could be multihomed to two PE routers. Device CE1 is multihomed to Routers PE1 and PE2. Device CE2 has two potential paths to reach Device CE1, but only one path is active at any one time. If Router PE1 were the designated PE router to forward traffic to the CE device (also called a designated forwarder), PE1 forwards traffic to Device CE1 using MPLS LSP or GRE tunnels. If a failure occurs over this path, a new designated forwarder is elected to forward the traffic to Device CE1.

Understanding EVPN Multihoming Concepts

Figure 19 on page 34 explains the EVPN multihoming concepts using a simple EVPN network topology.

- **Ethernet segment**—When a CE device is multihomed to two or more PE routers, the set of Ethernet links constitutes an Ethernet segment. An Ethernet segment appears as a Link Aggregation Group (LAG) to the CE device .

  The links from Routers PE1 and PE2 to Device CE1 form an Ethernet segment.

- **ESI**—An Ethernet segment must have a unique nonzero identifier, called the Ethernet segment identifier (ESI). The ESI is encoded as a 10 octet integer. When a single-homed CE device is attached to a Ethernet segment, the ESI value is zero.

  The Ethernet segment of the multihomed Device CE1 has an ESI value of **00:11:22:33:44:55:66:77:88:99** assigned. The single-homed Device CE2 has an ESI value of 0.

- **EVI**—An EVPN instance (EVI) is an EVPN routing and forwarding instance spanning across all the PE routers participating in that VPN. An EVI is configured on the PE routers on a per-customer basis. Each EVI has a unique route distinguisher and one or more route targets.

  An EVI is configured on Routers PE1, PE2, and PE3.

- **Ethernet tag**—An Ethernet tag identifies a particular broadcast domain, such as a VLAN. An EVPN instance consists of one or more broadcast domains. Ethernet tags are assigned to the broadcast domains of a given EVPN instance by the provider of that EVPN. Each PE router in that EVPN instance performs a mapping between broadcast domain identifiers understood by each of its attached CE devices and the corresponding Ethernet tag.

- **Ethernet segment route**—The PE routers that are connected to a multihomed CE device use BGP Ethernet segment route messages to discover that each of the PE routers is connected to the same Ethernet segment. The PE routers advertise the Ethernet segment route, which consists of an ESI and ES-import extended community.

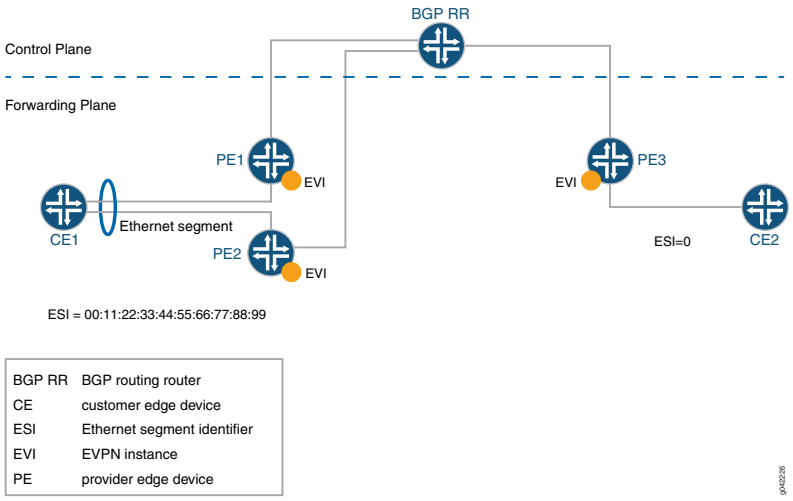
  Routers PE1 and PE2 advertise an ES route with an ES-import extended community (along with other extended communities like the route target). The PE routers also construct a filter based on an ES-import extended community, which results in only these PE routers importing the ES route and identifying that they are connected to the same Ethernet segment.

- **Extended community**— An extended community is similar in most ways to a regular community. EVPNs use extended communities because the 4-octet regular community value does not provide enough expansion and flexibility. An extended community is an eight-octet value divided into two main sections.

- **BUM traffic**—This type of traffic is sent to multiple destinations, including broadcast traffic, unknown unicast traffic that is broadcast in the Ethernet segment, and multicast traffic.

- **DF**—When a CE device is multihomed to two or more PE routers, one of the PE routers is used to reach the customer site. The PE router that assumes the primary role for forwarding BUM traffic to the CE device is called the designated forwarder (DF).
- **BDF**—Each router in the set of other PE routers advertising the autodiscovery route per Ethernet segment for the same ESI, and serving as the backup path in case the DF encounters a failure, is called a backup designated forwarder (BDF). A BDF is also called a non-DF router.
- **DF election**—On every Ethernet segment, the PE routers participate in a procedure called designated forwarder (DF) election to select the DF and the BDF PE routers.

Figure 19: Simple EVPN Topology



EVPN Multihoming Mode of Operation

The different modes of operation for EVPN multihoming include:

- Single—When a PE router is connected to a single-homed customer site, this mode is in operation. This is the default mode of operation, and does not require Ethernet segment values to be configured.
- Active-Standby—When only a single PE router, among a group of PE routers attached to an Ethernet segment, is allowed to forward traffic to and from that Ethernet segment, the Ethernet segment is defined to be operating in the active-standby redundancy mode.

  To configure the active-standby mode, include the ESI value and **single-active** mode under interface configuration.
- Active-Active—When all PE routers attached to an Ethernet segment are allowed to forward traffic to and from the Ethernet segment, the Ethernet segment is defined to be operating in an active-active redundancy mode.

NOTE: The EX9200 Series switch supports only the active-standby mode of operation for EVPN multihoming, and the MX Series router supports the active-active mode of operation.

NOTE: Starting with Junos OS Release 14.1x53-D30, QFX5100 switches support the active-active mode of operation for EVPN multihoming. In this scenario, QFX5100 switches function as top-of-rack (ToR) switches in the data center for virtual networks. EVPN multihoming active-active functionality is used to provide access to the bare-metal servers connected to the ToR switches.

Active-Standby Multihoming Implementation

The EVPN active-standby multihoming mode of operation provides redundancy for access link failures and PE node failure for the multihomed CE device. The active-standby multihoming feature is based on the EVPN *draft-ietf-l2vpn-evpn-03*.

The Junos OS implementation of the EVPN multihoming active-standby mode of operation includes the following:

New BGP NLRIs

To support active-standby EVPN multihoming, two new BGP network layer reachability information (NLRI) routes have been introduced:

- Autodiscovery route per Ethernet segment

  In active-standby mode, the designated forwarder (DF) advertises the autodiscovery route per Ethernet segment with an ESI MPLS label extended community that has the standby bit set to **1**. The autodiscovery route is advertised per ESI, and the ESI label is set to **0** when active-standby mode is in operation.

  The autodiscovery route is imported by all the multihomed and remote PE routers that are part of the EVI. On receiving the autodiscovery route, the PE routers in the network topology learn that active-standby multihoming mode is in operation for the ESI advertised.

The autodiscovery route NLRI features include:

- This is a Type 1 mandatory route, used for fast convergence and for advertising the split-horizon label. It is also known as the mass withdraw route.
- Type 1 route distinguishers are used with the IP address (loopback) of the originating PE router as the route distinguisher value.
- This route carries the ESI in the NLRI (nonzero when it is a multihomed PE, zero otherwise).
- The split-horizon label is per ESI only, and carries an explicit NULL (0).
- The bit in the active-standby flag field in the ESI label extended community is used for signaling the active-standby mode (bit set).
- The 3-byte label values in the NLRI and the Ethernet tag is zero.
- This route is advertised and imported by all multihomed and remote PE routers that share the same EVI on the advertising ESI.

- Ethernet segment route

  The Ethernet segment route is exchanged among all the PE routers within a data center with the ES-import extended community. The ES-import extended community is constructed based on the ESI PE routers that are multihomed, and the Ethernet segment route carries the ESI value related to the Ethernet segment on which the PE routers are multihomed.

  The Ethernet segment routes are filtered based on the ES-import extended community, such that only the PE routers that are multihomed on the same Ethernet segment import this route. Each PE router that is connected to a particular Ethernet segment constructs an import filtering rule to import a route that carries the ES-import extended community.

  The Ethernet segment route NRLI features include:

  - This is a Type 4 route. The purpose of this route is to enable the PE routers connected to the same Ethernet segment to automatically discover each other with minimal configuration on exchanging this route.
  - This route is associated with an ES-import extended community with an ESI value condensed to 6 bytes, similar to a route target.
  - This route is advertised and imported only by PE routers that are multihomed on the advertising Ethernet segment.

New Extended Communities

An extended community is similar in most ways to a regular community. Some networking implementations, such as virtual private networks (VPNs), use extended communities because the 4-octet regular community value does not provide enough expansion and flexibility. An extended community is an 8-octet value divided into two main sections.

To support active-standby multihoming, two new extended communities have been introduced:

- ESI-import extended community—This extended community is attached to the ES route, and is populated from the ESI-import value extracted from the configured ESI value under the interface. To solve the problem of a conflict with another regular route target, the type is set to **0x06**, which has been allocated by IANA.

  The ESI-import extended community route target populates the list of import route targets configured for the special instance from where the ES route using this community is advertised.

  Therefore, incoming ESI routes with the same ESI-import value in the extended community are imported by the PE routers, if the PE router is configured with an Ethernet segment that has the same ESI value. Once the PE router receives a set of these ESI routes that have the same ESI-import extended community value, the DF and BDF election can be done locally.

  NOTE: **When the ESI-import extended community is not created implicitly, a policy should be configured to attach all the route targets to the autodiscovery route per Ethernet segment.**

- Split horizon extended community—This extended community is attached to the autodiscovery route per Ethernet segment. The value of the extended community is the split-horizon or the Poisson label itself, which is 3 bytes, and is advertised as an opaque attribute.

New EVPN Route Types

Active-standby multihoming mode supports the following EVPN route types:

- Autodiscovery route per Ethernet segment
- Ethernet segment route

These route types conform to the following naming convention:

<route-type>:<RD>::<esi>::<route-specific>/304

For example:

Autodiscovery route per Ethernet segment—**1:10.255.0.2:0::112233445566778899::0/304**

Ethernet segment route—**4:10.255.0.1:0::112233445566778899:10.255.0.1/304**

where:

- **route-type**—Type of EVPN route.
  - 1—Autodiscovery route per Ethernet segment.
  - 4—Ethernet segment route.
- **RD**—Route distinguisher value.

  The route distinguisher value is set to the IP address of the PE router followed by 0.
- **esi**—Ethernet segment identifier. Displayed as 10 bytes of hexadecimal bytes, and leading 00 bytes are not displayed.
- **route-specific**—Differs per route type.
  - Autodiscovery route per Ethernet segment—This value is an MPLS label.

    NOTE: **The MPLS label is displayed in the extensive output, although it is not included in the prefix.**
  - Ethernet segment route—This value is the originating IP address.
- **304**—Maximum number of bits in an EVPN route. This is not very useful information and could be removed from the display. However, it might be useful in quickly identifying an EVPN route, either visually or with match operators.

Update to the MAC Forwarding Table

In active-standby EVPN multihoming, the MAC addresses are treated as routable addresses, and the MP-IBGP protocol is used to carry the customer MAC addresses. MAC learning at the PE routers does not occur in the data plane but in the control plane. This leads to more control applied in terms of the learning mechanism.

A PE router performs MAC learning in the data plane for packets coming from a customer network for a particular EVI. For CE MAC addresses that are behind other PE routers, the MAC addresses are advertised in BGP NLRI using a new MAC advertisement route type.

The MAC learning is of two types:

- Local MAC learning—PE routers must support the local MAC learning process through standard protocols.

- Remote MAC learning—Once the local learning process is completed, the PE routers can advertise the locally learned MAC address to remote PE router nodes through MP-IBGP. This process of receiving the remote MAC addresses of attached customers through MP-IBGP is known as the remote MAC learning process.

The MAC advertisement route type is used to advertise locally learned MAC addresses in BGP to remote PE routers. If an individual MAC address is advertised, the IP address field corresponds to that MAC address. If the PE router sees an ARP request for an IP address from a CE device, and if the PE router has the MAC address binding for that IP address, the PE router performs ARP proxy and responds to the ARP request.

NOTE: The ARP proxy is performed only for the gateway and not for the host.

The MPLS label field depends on the type of allocation. The PE router can advertise a single MPLS label for all MAC addresses per EVI, which requires the least number of MPLS labels and saves the PE router memory. However, when forwarding to the customer network, the PE router must perform a MAC lookup which can cause a delay and increase the number of CPU cycles.

Traffic Flow

In active-standby multihoming mode, there are two types of traffic flows:

- Unicast

  Unicast traffic is a point-to-point communication with one sender and one receiver. In a multihomed EVPN, unicast traffic is forwarded as follows:

  a. CE to core—Traffic is learned and forwarded by the DF PE router.

  b. Core to CE—The remote PE router learns the MAC addresses from the DF, and forwards all unicast traffic to the DF PE router.

- BUM

  Traffic that is sent to multiple destinations, including broadcast traffic, unknown unicast traffic that is broadcast in the Ethernet segment, and multicast traffic is known as BUM traffic. In a multihomed EVPN, BUM traffic is forwarded as follows:

  a. CE to core—The CE device floods any BUM traffic to all the links in the Ethernet segment. The DF PE router with the active path forwards the BUM packets to the core. The BDF PE router in the standby mode drops all the traffic from the CE device, because the EVPN multihomed status of the interface is in blocking state.

  b. Core to CE—The remote PE routers flood all BUM traffic to both the DF and BDF PE routers. Only the DF forwards the BUM traffic to the CE device. The BDF PE router drops all the traffic, because the EVPN multihomed status of the interface is in blocking state.

Sample Configuration

The following is a sample configuration for EVPN active-standby multihoming on the following types of interfaces:

- Ethernet interface configuration

```
ge-0/1/2 {
  encapsulation ethernet-bridge;
  esi XX:XX:XX:XX:XX:XX:XX:XX:XX:XX;
  unit 0 {
    family bridge;
  }
}
```

- Single VLAN interface configuration

```
ge-0/1/3 {
  encapsulation extended-vlan-bridge;
  esi XX:XX:XX:XX:XX:XX:XX:XX:XX:XX;
  vlan-tagging
  unit 0 {
    family bridge;
    vlan-id 1;
  }
}
```

NOTE:

- An ESI value of 0 and all FFs are reserved and are not used for configuring a multihomed Ehernet segment.
- Two interfaces in the same EVI cannot be configured with the same ESI value.

The following is a sample routing instance configuration for EVPN active-standby multihoming:

- Routing instance configuration

```
routing-instances {
  evpn-0 {
    instance-type evpn;
    route-distinguisher value;
    vrf-target value;
    vlan-id vlan-ID;
    interface ge-0/1/2.0;
    interface ge-1/1/1.0;
    interface ge-2/2/2.0;
    protocols {
      evpn {
        designated-forwarder-election hold-time time;
      }
    }
  }
```

}

NOTE: With the active-standby mode configuration, the autodiscovery route per Ethernet segment is advertised with the active-standby bit set to 1 for this Ethernet segment.

Designated Forwarder Election

The following sections discuss DF election:

DF Election Roles

The designated forwarder (DF) election process involves selecting the following two roles:

- **DF**—The MAC address from the customer site is reachable only through the PE router announcing the associated MAC advertisement route. This PE router is the primary PE router that is selected to forward BUM traffic to the multihomed CE device, and is called the designated forwarder (DF) PE router.
- **BDF**—Each PE router in the set of other PE routers advertising the autodiscovery route per Ethernet segment for the same ESI, and serving as the backup path in case the DF encounters a failure, is called a backup designated forwarder (BDF) or a non-DF (non-designated forwarder).

  As a result of the DF election process, if a local PE router is elected as the BDF, the multihomed interface connecting to the customer site is put into a blocking state for the active-standby mode. The interface remains in the blocking state until the PE router is elected as the DF for the Ethernet segment that the interface belongs to.

DF Election Procedure

The default procedure for DF election at the granularity of the ESI and EVI is referred to as service carving. With service carving, it is possible to elect multiple DFs per Ethernet segment (one per EVI) in order to perform load-balancing of multidestination traffic destined to a given Ethernet segment. The load-balancing procedures carve up the EVI space among the PE nodes evenly, in such a way that every PE is the DF for a disjoint set of EVIs.

The procedure for service carving is as follows:

1. When a PE router discovers the ESI of the attached Ethernet segment, it advertises an autodiscovery route per Ethernet segment with the associated ES-import extended community attribute.
2. The PE router then starts a timer (default value of 3 seconds) to allow the reception of the autodiscovery routes from other PE nodes connected to the same Ethernet segment. This timer value must be the same across all the PE routers connected to the same Ethernet segment.

The default wait timer can be overwritten using the **designated-forwarder-election hold-time** configuration statement.

3. When the timer expires, each PE router builds an ordered list of the IP addresses of all the PE nodes connected to the Ethernet segment (including itself), in increasing numeric order. Every PE router is then given an ordinal indicating its position in the ordered list, starting with 0 as the ordinal for the PE with the numerically lowest IP address. The ordinals are used to determine which PE node is the DF for a given EVI on the Ethernet segment.
4. The PE router that is elected as the DF for a given EVI unblocks traffic for the Ethernet tags associated with that EVI. The DF PE unblocks multidestination traffic in the egress direction toward the Ethernet segment. All the non-DF PE routers continue to drop multidestination traffic (for the associated EVIs) in the egress direction toward the Ethernet segment.

DF Election Trigger

In general, a DF election process is triggered in the following conditions:

- When an interface is newly configured with a nonzero ESI, or when the PE router transitions from an isolated-from-the-core (no BGP session) state to a connected-to-the-core (has established BGP session) state, a wait timer is imposed. By default, the interface is put into a blocking state until the PE router is elected as the DF.
- After completing a DF election process, a PE router receives a new Ethernet segment route or detects the withdrawal of an existing Ethernet segment route, without an imposed wait timer.
- When an interface of a non-DF PE router recovers from a link failure, the PE router has no knowledge of the wait time imposed by other PE routers. As a result, no wait timer is imposed for the recovered PE router to avoid traffic loss.

Handling Failover

A failover can happen due to two things:

- When the DF PE router loses its DF role.
- When there is a link or port failure on the DF PE router.

On losing the DF role, the customer-facing interface on the DF PE router is put in the blocking state.

In the case of link or port failure, a DF election process is triggered, resulting in the BDF PE router to be selected as the DF. At that time, flow of traffic is affected as follows:

- Unicast Traffic on page 43
- BUM Traffic on page 43

Unicast Traffic

- CE to Core—The CE device continues to flood traffic on all the links. The previous BDF PE router changes the EVPN multihomed status of the interface from the blocking state to the forwarding state, and traffic is learned and forwarded through this PE router.
- Core to CE—The failed DF PE router withdraws the autodiscovery route per Ethernet segment and the locally-learned MAC routes, causing the remote PE routers to redirect traffic to the BDF.

NOTE: The transition of the BDF PE router to the DF role can take some time, causing the EVPN multihomed status of the interface to continue to be in the blocking state, resulting in traffic loss.

BUM Traffic

- CE to Core—All the traffic is routed toward the BDF.
- Core to CE—The remote PE routers flood the BUM traffic in the core.

Related Documentation

- *Example: Configuring EVPN with Multihoming Support*

PART 2

Configuration

CHAPTER 2

Configuring EVPN and VXLAN

Example: Configuring VNI Route Targets Automatically

This example shows how to automatically derive route targets for multiple VNIs.

Requirements

This example uses the following hardware and software components:

- A Juniper Networks QFX5100 switch.
- Junos OS version 14.1X53-D30

Overview

The **vrf-target** statement can be used to configure specific route targets for each VNI under **vni-routing-options**. You can configure the **vrf-target** statement with the **auto** option to automatically derive route targets for each VNI.

Configuration

To configure the **vrf-target** statement with the **auto** option, perform these tasks:

Configuring VNI Route Target Automatic Derivation

Step-by-Step Procedure

To configure the automatic derivation of VNI route targets, you need to configure statements under 3 different hierarchy levels, **[edit switch-options]**, **[edit protocols evpn]**, and **[edit policy-options policy-statement]**.

1. At the **switch-options** hierarchy level, configure the **vtep-source-interface**, and **route-distiguisher** statements. Then configure the **vrf-import** statement with a policy that will be configured in a later step. Next, configure the **vrf-target** statement with a **target** and the **auto** option. The route target configured under **vrf-target** will be used by type 1 EVPN routes. Type 2 and type 3 EVPN routes will use the auto-derived per-VNI route target for export and import.

   ```
   [edit switch-options]

   user@switch# set vtep-source-interface lo0.0
   user@switch# set route-distinguisher 1.2.3.11:1
   user@switch# set vrf-import import-policy
   user@switch# set vrf-target target:1111:11
   user@switch# set vrf-target auto
   ```

2. At the **evpn** hierarchy level, configure the **encapsulation** and **extended-vni-list** statements. For the purposes of this example, the **extended-vni-list** statement will be configured with **all**, to apply automatic route targets to all VNIs.

   ```
   [edit protocols evpn]

   user@switch# set encapsulation vxlan
   user@switch# set extended-vni-list all
   ```

3. Configure 2 policies at the **policy-statement** hierarchy level. The first policy will be named **comglobal** and the next policy will be named **import-policy**. These policies function as an import filter that accepts routes with the auto-derived route target.

   ```
   [edit policy-options community comglobal]

   user@switch# set members target:1111:11

   [edit policy-options policy-statement import-policy]

   user@switch# set term 1 from community comglobal
   user@switch# set term 1 then accept
   user@switch# set term 100 then reject
   ```

Results

After following the steps above, use the **show** command to verify the results of your configuration.

```
user@switch> show configuration switch-options
  vtep-source-interface lo0.0;
  route-distinguisher 1.2.3.11:1;
  vrf-import imp;
  vrf-target {
      target:1111:11;
      auto;
  }

user@switch> show configuration protocols evpn
```

```
    encapsulation vxlan;
    extended-vni-list all;
```

```
user@switch> show configuration policy-options community comglobal
    members target:1111:11;
```

```
user@switch> show configuration policy-options policy-statement import-policy
    term 1{
        from community [ comglobal ];
        then accept;
    }
    term 100 {
        then reject;
    }
```

Related Documentation

- vrf-target on page 70
- Example: Configuring VNI Route Targets Manually on page 49
- Example: Configuring VNI Route Targets Automatically with Manual Override on page 50

Example: Configuring VNI Route Targets Manually

This example shows how to manually set route targets for multiple VNIs.

Requirements

This example uses the following hardware and software components:

- A Juniper Networks QFX5100 switch.
- Junos OS version 14.1X53-D30

Overview

The **vrf-target** statement can be used to configure specific route targets for each VNI under **vni-routing-options**. You can configure the **vrf-target** statement to automatically derive route targets for each VNI or you can configure route targets manually. This example explains how to configure route targets manually.

Configuration

To manually configure VNI route targets, perform these tasks:

Configuring VNI Route Targets Manually

Step-by-Step Procedure

To manually configure VNI route targets, you need to configure statements under 2 different hierarchy levels, **[edit switch-options]** and **[edit protocols evpn]**.

1. At the **switch-options** hierarchy level, configure the **vtep-source-interface**, and **route-distiguisher** statements. Next, configure the **vrf-target** statement with a **target**. All EVPN routes for all VLANs will use the **vrf-target** address configured in this step.

   ```
   [edit switch-options]

   user@switch# set vtep-source-interface lo0.0
   user@switch# set route-distinguisher 1.2.3.11:1
   user@switch# set vrf-import import-policy
   user@switch# set vrf-target target:1111:11
   ```

2. At the **evpn** hierarchy level, configure the **encapsulation** and **extended-vni-list** statements. For the purposes of this example, the **extended-vni-list** statement will be configured with **all**, to apply automatic route targets to all VNIs.

   ```
   [edit protocols evpn]

   user@switch# set encapsulation vxlan
   user@switch# set extended-vni-list all
   ```

Results

After following the steps above, use the **show** command to verify the results of your configuration.

```
user@switch> show configuration switch-options
  vtep-source-interface lo0.0;
  route-distinguisher 1.2.3.11:1;
  vrf-target {
      target:1111:11;
  }
```

Related Documentation

- vrf-target on page 70
- Example: Configuring VNI Route Targets Automatically on page 47
- Example: Configuring VNI Route Targets Automatically with Manual Override on page 50

Example: Configuring VNI Route Targets Automatically with Manual Override

This example shows how to automatically set route targets for multiple VNIs, and manually override the route target for a single VNI.

- Requirements on page 51
- Overview on page 51
- Configuration on page 51

Requirements

This example uses the following hardware and software components:

- A Juniper Networks QFX5100 switch.
- Junos OS version 14.1X53-D30

Overview

The **vrf-target** statement can be used to configure specific route targets for each VNI under **vni-routing-options**. You can configure the **vrf-target** statement to automatically derive route targets for each VNI or you can configure route targets manually. It's also possible to automatically derive route targets for VNIs, then manually override the route target for one or more VNIs. This example explains how to manually override the automatically assigned route targets for a specific VNI.

Configuration

To manually override an automatically configured VNI route targets, perform these tasks:

- Configuring Automatic VNI Route Targets with Manual Override on page 51
- Results on page 52

Configuring Automatic VNI Route Targets with Manual Override

Step-by-Step Procedure

This example requires configuring statements under 3 different hierarchy levels, **[edit switch-options]**, **[edit protocols evpn]**, and **[edit policy-options policy-statement]**.

1. At the **switch-options** hierarchy level, configure the **vtep-source-interface**, and **route-distiguisher** statements. Then configure the **vrf-import** statement with a policy that will be configured in a later step. . Next, configure the **vrf-target** statement with a **target** and the **auto** option. The route target configured under **vrf-target** will be used by type 1 EVPN routes and all type 2 and 3 EVPN routes for all VLANs except the ones that do not match the VNI under **vni-options** in the next step.

   ```
   [edit switch-options]
   ```

   ```
   user@switch# set vtep-source-interface lo0.0
   user@switch# set route-distinguisher 1.2.3.11:1
   user@switch# set vrf-import import-policy
   user@switch# set vrf-target target:1111:11
   user@switch# set vrf-target auto
   ```

2. The **evpn** hierarchy level is where you can override the automatic assignment of VNI route targets. Configure the **vni-options** statement for VNI 100 with an export target of 1234:11. This route target will be used by type 2 and 3 EVPN routes for all VLANs that match VNI 100. Next, configure the **encapsulation** and **extended-vni-list** statements. For the purposes of this example, the **extended-vni-list** statement will be configured with only 2 VNIs.

   ```
   [edit protocols evpn]
   ```

```
user@switch# vni-options vni 100 vrf-target export target:1234:11
user@switch# set encapsulation vxlan
user@switch# set extended-vni-list 100 101
```

3. Configure 3 policies at the **policy-statement** hierarchy level. The first policy will be named **comglobal**, the next policy will be named **com1234**, and the final policy will be named **import-policy**. These policies function as an import filter that accepts routes using the auto-derived route target and the manual override route target.

```
[edit policy-options community comglobal]

user@switch# set members target:1111:11

[edit policy-options policy-statement com1234]

user@switch# set members target:1234:11

[edit policy-options policy-statement import-policy]

user@switch# set term 1 from community comglobal com1234
user@switch# set term 1 then accept
user@switch# set term 100 then reject
```

Results

After following the steps above, use the **show** command to verify the results of your configuration.

user@switch> show configuration switch-options

```
vtep-source-interface lo0.0;
route-distinguisher 1.2.3.11:1;
vrf-import imp;
vrf-target {
    target:1111:11;
    auto;
}
```

user@switch> show configuration protocols evpn

```
vni-options {
    vni 100 {
        vrf-target export target:1234:11;
    }
}
encapsulation vxlan;
extended-vni-list [ 100 101 ];
```

user@switch> show configuration policy-options community comglobal

```
members target:1111:11;
```

user@switch> show configuration policy-options community com1234

```
members target:1234:11;
```

user@switch> show configuration policy-options policy-statement import-policy

```
term 1{
    from community [ com1234 comglobal ];
    then accept;
}
term 100 {
    then reject;
}
```

Related Documentation

- **vrf-target on page 70**
- Example: Configuring VNI Route Targets Automatically on page 47
- Example: Configuring VNI Route Targets Manually on page 49

PART 3

Configuration Statements and Operational Commands

CHAPTER 3

EVPN and VXLAN Configuration Statements

designated-forwarder-election-hold-time (evpn)

Syntax

```
designated-forwarder-election-hold-time seconds {
  encapsulation
  extended-vni-list
  no-default-gateway-ext-comm
}
```

Hierarchy Level

[edit routing-instances protocols evpn]
[edit protocols evpn]

Release Information

Statement introduced in Junos OS Release 14.1X53-D30.

Description

A designated forwarder (DF) is required when customer edge devices (CEs) are multihomed to more than one provider edge device (PE). Without a designated forwarder, multihomed hosts would receive duplicate packets. Designated forwarders are chosen for an Ethernet segment identifier (ESI) based on type-4 route advertisements.

Required Privilege Level

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

Related Documentation

- Implementing EVPN with VXLAN for Data Centers on page 19
- EVPN Multihoming Overview on page 31

encapsulation (Logical Interface)

Syntax encapsulation (atm-ccc-cell-relay | atm-ccc-vc-mux | atm-cisco-nlpid | atm-mlppp-llc | atm-nlpid | atm-ppp-llc | atm-ppp-vc-mux | atm-snap | atm-tcc-snap | atm-tcc-vc-mux | atm-vc-mux | ether-over-atm-llc | ether-vpls-over-atm-llc | ether-vpls-over-fr | ether-vpls-over-ppp | ethernet | ethernet-ccc | ethernet-vpls | ethernet-vpls-fr | frame-relay-ccc | frame-relay-ether-type | frame-relay-ether-type-tcc | frame-relay-ppp | frame-relay-tcc | gre-fragmentation | multilink-frame-relay-end-to-end | multilink-ppp | ppp-over-ether | ppp-over-ether-over-atm-llc | vlan-bridge | vlan-ccc | vlan-vci-ccc | vlan-tcc | vlan-vpls | vxlan);

Hierarchy Level [edit interfaces *interface-name* unit *logical-unit-number*],
[edit logical-systems *logical-system-name* interfaces *interface-name* unit *logical-unit-number*],
[edit interfaces rlsq *number* unit *logical-unit-number*]
[edit protocols evpn]

Release Information Statement introduced before Junos OS Release 7.4.
Statement introduced in Junos OS Release 12.1X48 for PTX Series Packet Transport Routers (**ethernet**,**vlan-ccc**, and **vlan-tcc** options only).
Statement introduced in Junos OS Release 12.2 for the ACX Series Universal Access Routers. Only the **atm-ccc-cell-relay** and **atm-ccc-vc-mux** options are supported on ACX Series routers.
Support for **vlan-bridge** option introduced in Junos OS Release 14.1X53-D35 for the QFX Series.

Description Configure a logical link-layer encapsulation type.

Options **atm-ccc-cell-relay**—Use ATM cell-relay encapsulation.

atm-ccc-vc-mux—Use ATM virtual circuit (VC) multiplex encapsulation on CCC circuits. When you use this encapsulation type, you can configure the **ccc** family only.

atm-cisco-nlpid—Use Cisco ATM network layer protocol identifier (NLPID) encapsulation. When you use this encapsulation type, you can configure the **inet** family only.

atm-mlppp-llc—For ATM2 IQ interfaces only, use Multilink Point-to-Point (MLPPP) over AAL5 LLC. For this encapsulation type, your router must be equipped with a Link Services or Voice Services PIC. MLPPP over ATM encapsulation is not supported on ATM2 IQ OC48 interfaces.

atm-nlpid—Use ATM NLPID encapsulation. When you use this encapsulation type, you can configure the **inet** family only.

atm-ppp-llc—(ATM2 IQ interfaces and MX Series routers with MPC/MIC interfaces using the ATM MIC with SFP only) Use PPP over AAL5 LLC encapsulation.

atm-ppp-vc-mux—(ATM2 IQ interfaces and MX Series routers with MPC/MIC interfaces using the ATM MIC with SFP only) Use PPP over ATM AAL5 multiplex encapsulation.

atm-snap—(All interfaces including MX Series routers with MPC/MIC interfaces using the ATM MIC with SFP) Use ATM subnetwork attachment point (SNAP) encapsulation.

atm-tcc-snap—Use ATM SNAP encapsulation on translational cross-connect (TCC) circuits.

atm-tcc-vc-mux—Use ATM VC multiplex encapsulation on TCC circuits. When you use this encapsulation type, you can configure the **tcc** family only.

atm-vc-mux—(All interfaces including MX Series routers with MPC/MIC interfaces using the ATM MIC with SFP) Use ATM VC multiplex encapsulation. When you use this encapsulation type, you can configure the **inet** family only.

ether-over-atm-llc—(All IP interfaces including MX Series routers with MPC/MIC interfaces using the ATM MIC with SFP) For interfaces that carry IP traffic, use Ethernet over ATM LLC encapsulation. When you use this encapsulation type, you cannot configure multipoint interfaces.

ether-vpls-over-atm-llc—For ATM2 IQ interfaces only, use the Ethernet virtual private LAN service (VPLS) over ATM LLC encapsulation to bridge Ethernet interfaces and ATM interfaces over a VPLS routing instance (as described in RFC 2684, *Multiprotocol Encapsulation over ATM Adaptation Layer 5*). Packets from the ATM interfaces are converted to standard ENET2/802.3 encapsulated Ethernet frames with the frame check sequence (FCS) field removed.

ether-vpls-over-fr—For E1, T1, E3, T3, and SONET interfaces only, use the Ethernet virtual private LAN service (VPLS) over Frame Relay encapsulation to support Bridged Ethernet over Frame Relay encapsulated TDM interfaces for VPLS applications, per RFC 2427, *Multiprotocol Interconnect over Frame Relay*.

NOTE: **The SONET/SDH OC3/STM1 (Multi-Rate) MIC with SFP, the Channelized SONET/SDH OC3/STM1 (Multi-Rate) MIC with SFP, and the DS3/E3 MIC do not support Ethernet over Frame Relay encapsulation.**

ether-vpls-over-ppp—For E1, T1, E3, T3, and SONET interfaces only, use the Ethernet virtual private LAN service (VPLS) over Point-to-Point Protocol (PPP) encapsulation to support Bridged Ethernet over PPP-encapsulated TDM interfaces for VPLS applications.

ethernet—Use Ethernet II encapsulation (as described in RFC 894, *A Standard for the Transmission of IP Datagrams over Ethernet Networks*).

ethernet-ccc—Use Ethernet CCC encapsulation on Ethernet interfaces.

ethernet-vpls—Use Ethernet VPLS encapsulation on Ethernet interfaces that have VPLS enabled and that must accept packets carrying standard Tag Protocol ID (TPID) values.

NOTE: The built-in Gigabit Ethernet PIC on an M7i router does not support extended VLAN VPLS encapsulation.

ethernet-vpls-fr—Use in a VPLS setup when a CE device is connected to a PE router over a time-division multiplexing (TDM) link. This encapsulation type enables the PE router to terminate the outer layer 2 Frame Relay connection, use the 802.1p bits inside the inner Ethernet header to classify the packets, look at the MAC address from the Ethernet header, and use the MAC address to forward the packet into a given VPLS instance.

frame-relay-ccc—Use Frame Relay encapsulation on CCC circuits. When you use this encapsulation type, you can configure the **ccc** family only.

frame-relay-ether-type—Use Frame Relay ether type encapsulation for compatibility with Cisco Frame Relay. The physical interface must be configured with flexible-frame-relay encapsulation.

frame-relay-ether-type-tcc—Use Frame Relay ether type TCC for Cisco-compatible Frame Relay on TCC circuits to connect different media. The physical interface must be configured with flexible-frame-relay encapsulation.

frame-relay-ppp—Use PPP over Frame Relay circuits. When you use this encapsulation type, you can configure the **ppp** family only.

frame-relay-tcc—Use Frame Relay encapsulation on TCC circuits for connecting different media. When you use this encapsulation type, you can configure the **tcc** family only.

gre-fragmentation—For adaptive services interfaces only, use GRE fragmentation encapsulation to enable fragmentation of IPv4 packets in GRE tunnels. This encapsulation clears the do not fragment (DF) bit in the packet header. If the packet's size exceeds the tunnel's maximum transmission unit (MTU) value, the packet is fragmented before encapsulation.

multilink-frame-relay-end-to-end—Use MLFR FRF.15 encapsulation. This encapsulation is used only on multilink, link services, and voice services interfaces and their constituent T1 or E1 interfaces, and is supported on LSQ and redundant LSQ interfaces.

multilink-ppp—Use MLPPP encapsulation. This encapsulation is used only on multilink, link services, and voice services interfaces and their constituent T1 or E1 interfaces.

ppp-over-ether—Use PPP over Ethernet encapsulation to configure an underlying Ethernet interface for a dynamic PPPoE logical interface on M120 and M320 routers with Intelligent Queuing 2 (IQ2) PICs, and on MX Series routers with MPCs.

ppp-over-ether-over-atm-llc—(MX Series routers with MPCs using the ATM MIC with SFP only) For underlying ATM interfaces, use PPP over Ethernet over ATM LLC encapsulation. When you use this encapsulation type, you cannot configure the interface address. Instead, configure the interface address on the PPP interface.

vlan-bridge—Use Ethernet VLAN bridge encapsulation on Ethernet interfaces that have IEEE 802.1Q tagging, flexible-ethernet-services, and bridging enabled and that must accept packets carrying TPID 0x8100 or a user-defined TPID.

vlan-ccc—Use Ethernet virtual LAN (VLAN) encapsulation on CCC circuits. When you use this encapsulation type, you can configure the **ccc** family only.

vlan-vci-ccc—Use ATM-to-Ethernet interworking encapsulation on CCC circuits. When you use this encapsulation type, you can configure the **ccc** family only.

vlan-tcc—Use Ethernet VLAN encapsulation on TCC circuits. When you use this encapsulation type, you can configure the **tcc** family only.

vlan-vpls—Use Ethernet VLAN encapsulation on VPLS circuits.

vxlan—Use VXLAN data plane encapsulation for EVPN.

Required Privilege Level

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

Related Documentation

- *Configuring Layer 2 Switching Cross-Connects Using CCC*
- *Configuring the Encapsulation for Layer 2 Switching TCCs*
- *Configuring Interface Encapsulation on Logical Interfaces*
- *Configuring MPLS LSP Tunnel Cross-Connects Using CCC*
- *Circuit and Translational Cross-Connects Overview*
- *Identifying the Access Concentrator*
- *Configuring ATM Interface Encapsulation*
- *Configuring VLAN Encapsulation*
- *Configuring Extended VLAN Encapsulation*
- *Configuring ATM-to-Ethernet Interworking*
- *Configuring Interface Encapsulation on PTX Series Packet Transport Routers*
- *Configuring CCC Encapsulation for Layer 2 VPNs*
- *Configuring TCC Encapsulation for Layer 2 VPNs and Layer 2 Circuits*
- *Configuring ATM for Subscriber Access*
- *CoS on ATM IMA Pseudowire Interfaces Overview*
- *Configuring Policing on an ATM IMA Pseudowire*

evpn

Syntax

```
evpn {
  designated-forwarder-election-hold-timeseconds {
  extended-vni-list {
    vni-options
  }
  no-default-gateway-ext-comm
  encapsulation (Logical Interface)
  extended-vni-list
    vni-options{
      vni xxx vrf-target export target:xxx:xx
      vni xxx vrf-export name
    }
  extended-vlan-list vlan-id | [vlan-id set];
  extended-isid-list (single-isid | isid-list |isid-range | all)
  pbb-evpn-core
  control-word (EVPN)
  interface interface-name{
    ignore-encapsulation-mismatch;
    interface-mac-limit (vpls) limit {
      packet-action drop;
    }
    no-mac-learning;
    static-mac mac-address;
  }
  interface-mac-limit (vpls) limit {
    packet-action drop;
  }
  label-allocation per-instance;
  mac-statistics;
  mac-table-size limit {
    packet-action drop;
  }
  no-mac-learning;
  traceoptions {
    file filename <files number> <size size> <world-readable | no-world-readable>;
    flag flag <flag-modifier>;
  }
}
```

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

Release Information Statement introduced in Junos OS Release 13.2 for EVPNs on MX 3D Series routers.
designated-forwarder-election-hold-time *seconds* statement introduced in Junos OS Release 14.1.
extended-vlan-list *vlan-id* | [*vlan-id set*] statement introduced in Junos OS Release 14.1.
Statement introduced in Junos OS Release 14.1-X53-D30 for EVPNs on QFX5100 Series switches.

Description Enable an Ethernet VPN (EVPN) on the routing instance.

Options

designated-forwarder-election-hold-time *seconds*—Time in seconds to wait before electing a designated forwarder (DF).

Range: 1 through 1800 seconds

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

- Implementing EVPN with VXLAN for Data Centers on page 19
- *Configuring EVPN Routing Instances*
- *Tracing EVPN Traffic and Operations*
- Implementing EVPN with VXLAN for Data Centers on page 19

extended-vni-list

Syntax

set extended-vni-list XXXX

Hierarchy Level

For MX routers:

[edit routing-instances protocols evpn]
designated-forwarder-election-hold-time *seconds* {

For QFX switches:

[edit protocols evpn]
designated-forwarder-election-hold-time *seconds* {

Release Information

Statement introduced in Junos OS Release 14.1X53-D30.

Description

The extended-vni-list establishes which VXLAN Network Identifiers (VNIs) will be part of the Ethernet VPN (EVPN) and Virtual Extensible LAN (VXLAN) Multiprotocol BGP (MP-BGP) domain. There are different BUM replication options available in EVPN — using **extended-vni-list** forgoes a multicast underlay in favor of EVPN and VXLAN ingress replication.

Required Privilege Level

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

Related Documentation

- show configuration protocols evpn on page 76
- Implementing EVPN with VXLAN for Data Centers on page 19

ingress-node-replication (EVPN)

Syntax ingress-node-replication;

Hierarchy Level
```
[edit vlans] {
  vxlan
}
```

Release Information Statement introduced in Junos OS Release 14.1X53-D30.

Description Ingress replication is supported only for EVPN-VXLAN and enabled by default; no configuration is needed. EVPN A/A with VXLAN encapsulation is based on the local bias for traffic coming from the access layer (redundant Layer 2 gateway function through a pair of top-of-rack switches). Because the traffic has no MPLS label, the split-horizon filtering rule for multi-home Ethernet segment is modified to be based on the IP address of the EVPN PE instead of MPLS ES-label. This is called Local Bias for EVPN with VXLAN data plane encapsulation. Each EVPN PE tracks the IP address of its peer multihomed EVPN PE sharing the same Ethernet segment. This is the source VTEP IP address (outer SIP) for each VXLAN packet received from other EVPN PE. The local bias filtering rule is enforced on both ingress and egress PEs for the multi-destination traffic. For egress traffic, there is no forwarding of any multi-destination packets to the same multi-homed Ethernet segment that an egress PE shares with its ingress PE regardless of the egress PE's DF election status for that Ethernet segment. Ingress traffic is responsible for forwarding multi-destination packets coming from any directly attached access interfaces to the rest of the multi-home Ethernet segments associated with it regardless of the ingress PE's designated forwarder election status on the connected physical device segment.

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

Related Documentation
- EVPN with VXLAN Data Plane Encapsulation on page 13

no-default-gateway-ext-comm

Syntax no-default-gateway-ext-comm;

Hierarchy Level [edit routing-instances *name* protocols evpn]

Release Information Statement introduced in Junos OS Release 14X51-D30.

Description Configure a routing instance to suppress the advertisement of the extended community on the default gateway. An extended community is an 8-octet community used by EVPNs.

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

Related Documentation

- Implementing EVPN with VXLAN for Data Centers on page 19
- EVPN Multihoming Overview on page 31

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.
Statement introduced in Junos OS Release 14.1X53-D30 for QFX Series switches.

Description Specify an identifier attached to a route, enabling you to distinguish to which VPN or virtual private LAN service (VPLS) the route belongs. Each routing instance must have a unique route distinguisher associated with it. The RD 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 RD for each PE router participating in the routing instance.

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

For Layer 2 VPNs and VPLSs, if you configure mesh groups, the RD in each mesh group must be unique.

CAUTION: We strongly recommend that if you change an RD that has already been configured, make the change during a maintenance window, as follows:

1. Deactivate the routing instance.
2. Change the RD.
3. Activate the routing instance.

This is not required if you are configuring the RD 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. An RD consisting of a 4-byte AS number and a 2-byte administrative number is defined as a type 2 RD in RFC 4364 *BGP/MPLS IP 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 an RD that includes a 4-byte AS number, append the letter "L" to the end of the AS number. For example, an RD 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*
- *Example: Configuring FEC 129 BGP Autodiscovery for VPWS*
- *Configuring EVPN Routing Instances*
- *Configuring Routing Instances on PE Routers in VPNs*
- *Configuring an MPLS-Based Layer 2 VPN (CLI Procedure)*
- *Configuring an MPLS-Based Layer 3 VPN (CLI Procedure)*
- *l2vpn-use-bgp-rules*

vni-options (evpn)

Syntax

```
vni-options {
  vni xxx vrf-target export target:xxx:xx
  vni xxx vrf-export name
}
```

Hierarchy Level

For MX routers:

```
[edit routing-instances protocols evpn]
designated-forwarder-election-hold-time seconds {
```

For QFX switches:

```
[edit protocols evpn]
designated-forwarder-election-hold-time seconds {
```

Release Information

Statement introduced in Junos OS Release 14.1X53-D30.

Description

Configure VRF export targets and VRF export policies for specific VNIs.

Required Privilege Level

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

Related Documentation

- extended-vni-list on page 64
- show configuration protocols evpn on page 76
- Implementing EVPN with VXLAN for Data Centers on page 19

vrf-target

Syntax

```
vrf-target {
  community;
  auto
  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*],
[edit switch-options]

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.
Statement introduced in Junos OS Release 14.1X53-D30 for QFX Series Switches. auto option was also added at this time.

Description

Specify a virtual routing and forwarding (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.

auto—Automatically derives the route target (RT) for QFX5100 switches.

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

- Implementing EVPN with VXLAN for Data Centers on page 19
- *Configuring Policies for the VRF Table on PE Routers in VPNs*

- *Example: Configuring FEC 129 BGP Autodiscovery for VPWS*

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*]
[edit switch-options]

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 14.1X53-D30 for QFX Series Switches.

Description Specify how routes are imported into the virtual routing and forwarding (VRF) table (***routing-instance-name***.inet.0) of the local provider edge (PE) router or switch from 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 import policies on the PE router or switch.

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

- Implementing EVPN with VXLAN for Data Centers on page 19
- *instance-type*
- *Configuring Policies for the VRF Table on PE Routers in VPNs*

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*]
[edit switch-options]

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.
Statement introduced in Junos OS Release 14.1X53-D30 for QFX Series switches.

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.

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

- Implementing EVPN with VXLAN for Data Centers on page 19
- *instance-type*
- *Configuring Policies for the VRF Table on PE Routers in VPNs*

vxlan

Syntax

```
vxlan {
  encapsulate-inner-vlan
  ingress-node-replication
  multicast-group
  ovsdb-managed
  unreachable-vtep-aging-timer
  vni
  }
}
```

Hierarchy Level [edit vlans]

Release Information Statement introduced in Junos OS Release 14.1X53-D10.
ingress-node-replication option added for QFX Series switches in Junos OS Release 14.1X53-D30.

Description

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

- Understanding VXLANs on page 3
- *Configuring VXLANs on a QFX5100 Switch*
- *Examples: Manually Configuring VXLANs on QFX Series Switches*

CHAPTER 4

EVPN and VXLAN Operational Commands

- show configuration protocols evpn
- show route table

show configuration protocols evpn

Syntax show configuration protocols evpn

Release Information Command introduced in Junos OS Release 14.1X53-D30.

Description An Ethernet VPN (EVPN) enables you to connect a group of dispersed customer sites using a Layer 2 virtual bridge. This **show** command displays configured EVPN information.

Required Privilege Level view

Related Documentation
- extended-vni-list on page 64
- vni-options on page 69

List of Sample Output command-name (optional-text) on page 76

Output Fields Table 4 on page 76 describes the output fields for the command **show-protocols-evpn**.

Table 4: Output Fields for Command show protocols evpn

Field Name	Field Description
encapsulation	Encapsulation method will always be **vxlan**.
extended-vni-list	Lists the VXLAN virtual network identifiers (VNIs) that are part of the EVPN/VXLAN MP-BGP domain. These VNIs are configured with the command extended-vni-list.
multicast-mode	
vni-options	Different route targets (RTs) for each VNI instance configured with the command vni-options.

Sample Output

command-name (optional-text)

```
user@host> show configuration protocols evpn
encapsulation vxlan;
extended-vni-list [ 1100 1200 1300 1400 ];
multicast-mode ingress-replication;
vni-options {
    vni 1100 {
        vrf-target export target:1:100;
    }
    vni 1200 {
        vrf-target export target:1:200;
    }
    vni 1300 {
        vrf-target export target:1:300;
    }
    vni 1400 {
        vrf-target export target:1:400;
```

```
        }
    }
```

show route table

List of Syntax
Syntax on page 78
Syntax (EX Series Switches) on page 78

Syntax

show route table *routing-table-name*
<brief | detail | extensive | terse>
<logical-system (all | *logical-system-name*)>

Syntax (EX Series Switches)

show route table *routing-table-name*
<brief | detail | extensive | terse>

Release Information

Command introduced before Junos OS Release 7.4.
Command introduced in Junos OS Release 9.0 for EX Series switches.
Command introduced in Junos OS Release 14.1X53-D30 for QFX Series switches.

Description

Display the route entries in a particular routing table.

Options

brief | detail | extensive | terse—(Optional) Display the specified level of output.

logical-system (all | *logical-system-name*)—(Optional) Perform this operation on all logical systems or on a particular logical system.

routing-table-name—Display route entries for all routing tables whose names begin with this string (for example, inet.0 and inet6.0 are both displayed when you run the **show route table inet** command).

Required Privilege Level

view

Related Documentation

- *show route summary*

List of Sample Output

show route table bgp.l2.vpn on page 89
show route table bgp.l3vpn.0 on page 89
show route table bgp.l3vpn.0 detail on page 89
show route table bgp.rtarget.0 (When Proxy BGP Route Target Filtering Is Configured) on page 90
show route table bgp.evpn.0 on page 91
show route table evpna.evpn.0 on page 91
show route table inet.0 on page 91
show route table inet6.0 on page 92
show route table inet6.3 on page 92
show route table inetflow detail on page 92
show route table l2circuit.0 on page 93
show route table mpls on page 93
show route table mpls extensive on page 93
show route table mpls.0 on page 94
show route table mpls.0 detail (PTX Series) on page 95
show route table mpls.0 extensive (PTX Series) on page 95

show route table mpls.0 (RSVP Route—Transit LSP) on page 96
show route table vpls_1 detail on page 96
show route table vpn-a on page 96
show route table vpn-a.mdt.0 on page 97
show route table VPN-A detail on page 97
show route table VPN-AB.inet.0 on page 98
show route table VPN_blue.mvpn-inet6.0 on page 98
show route table vrf1.mvpn.0 extensive on page 98
show route table inetflow detail on page 99

Output Fields Table 5 on page 79 describes the output fields for the **show route table** command. Output fields are listed in the approximate order in which they appear.

Table 5: show route table Output Fields

Field Name	Field Description
routing-table-name	Name of the routing table (for example, inet.0).

Restart complete	All protocols have restarted for this routing table.
	
	Restart state:
	
	• **Pending:*protocol-name***—List of protocols that have not yet completed graceful restart for this routing table.
	• **Complete**—All protocols have restarted for this routing table.
	
	For example, if the output shows-
	
	• `LDP.inet.0 : 5 routes (4 active, 1 holddown, 0 hidden)`
	`Restart Pending: OSPF LDP VPN`
	
	This indicates that **OSPF**, **LDP**, and **VPN** protocols did not restart for the **LDP.inet.0** routing table.
	• `vpls_1.l2vpn.0: 1 destinations, 1 routes (1 active, 0 holddown, 0 hidden)`
	`Restart Complete`
	
	This indicates that all protocols have restarted for the **vpls_1.l2vpn.0** routing table.

number **destinations**	Number of destinations for which there are routes in the routing table.
number **routes**	Number of routes in the routing table and total number of routes in the following states: • **active** (routes that are active) • **holddown** (routes that are in the pending state before being declared inactive) • **hidden** (routes that are not used because of a routing policy)

Table 5: show route table Output Fields *(continued)*

Field Name	Field Description

route-destination **(entry, announced)**	Route destination (for example:10.0.0.1/24). The **entry** value is the number of routes for this destination, and the **announced** value is the number of routes being announced for this destination. Sometimes the route destination is presented in another format, such as:
	• ***MPLS-label*** (for example, 80001).
	• ***interface-name*** (for example, ge-1/0/2).
	• ***neighbor-address*:*control-word-status*:*encapsulation type*:*vc-id*:*source*** (Layer 2 circuit only; for example, 10.1.1.195:NoCtrlWord:1:1:Local/96).
	  ◦ ***neighbor-address***—Address of the neighbor.
	  ◦ ***control-word-status***—Whether the use of the control word has been negotiated for this virtual circuit: **NoCtrlWord** or **CtrlWord**.
	  ◦ ***encapsulation type***—Type of encapsulation, represented by a number: (1) Frame Relay DLCI, (2) ATM AAL5 VCC transport, (3) ATM transparent cell transport, (4) Ethernet, (5) VLAN Ethernet, (6) HDLC, (7) PPP, (8) ATM VCC cell transport, (10) ATM VPC cell transport.
	  ◦ ***vc-id***—Virtual circuit identifier.
	  ◦ ***source***—Source of the advertisement: **Local** or **Remote**.
	• ***inclusive multicast Ethernet tag route***—Type of route destination represented by (for example, 3:100.100.100.10:100::0::10::100.100.100.10/384):
	  ◦ *route distinguisher*—(8 octets) Route distinguisher (RD) must be the RD of the EVPN instance (EVI) that is advertising the NLRI.
	  ◦ *Ethernet tag ID*—(4 octets) Identifier of the Ethernet tag. Can set to 0 or to a valid Ethernet tag value.
	  ◦ *IP address length*—(1 octet) Length of IP address in bits.
	  ◦ *originating router's IP address*—(4 or 16 octets) Must set to the provider edge (PE) device's IP address. This address should be common for all EVIs on the PE device, and may be the PE device's loopback address.

label stacking	**(**Next-to-the-last-hop router for MPLS only) Depth of the MPLS label stack, where the label-popping operation is needed to remove one or more labels from the top of the stack. A pair of routes is displayed, because the pop operation is performed only when the stack depth is two or more labels. • **S=0 route** indicates that a packet with an incoming label stack depth of 2 or more exits on this router with one fewer label (the label-popping operation is performed). • If there is no **S=** information, the route is a normal MPLS route, which has a stack depth of 1 (the label-popping operation is not performed).
[*protocol*, *preference*]	Protocol from which the route was learned and the preference value for the route. • **+**—A plus sign indicates the active route, which is the route installed from the routing table into the forwarding table. • **-** —A hyphen indicates the last active route. • *****—An asterisk indicates that the route is both the active and the last active route. An asterisk before a **to** line indicates the best subpath to the route. In every routing metric except for the BGP **LocalPref** attribute, a lesser value is preferred. In order to use common comparison routines, Junos OS stores the 1's complement of the **LocalPref** value in the **Preference2** field. For example, if the **LocalPref** value for Route 1 is 100, the **Preference2** value is -101. If the **LocalPref** value for Route 2 is 155, the **Preference2** value is -156. Route 2 is preferred because it has a higher **LocalPref** value and a lower **Preference2** value.

Table 5: show route table Output Fields *(continued)*

Field Name	Field Description
Level	(IS-IS only). In IS-IS, a single AS can be divided into smaller groups called areas. Routing between areas is organized hierarchically, allowing a domain to be administratively divided into smaller areas. This organization is accomplished by configuring Level 1 and Level 2 intermediate systems. Level 1 systems route within an area. When the destination is outside an area, they route toward a Level 2 system. Level 2 intermediate systems route between areas and toward other ASs.
Route Distinguisher	IP subnet augmented with a 64-bit prefix.
PMSI	Provider multicast service interface (MVPN routing table).
Next-hop type	Type of next hop. For a description of possible values for this field, see Table 6 on page 84.
Next-hop reference count	Number of references made to the next hop.
Flood nexthop branches exceed maximum message	Indicates that the number of flood next-hop branches exceeded the system limit of 32 branches, and only a subset of the flood next-hop branches were installed in the kernel.
Source	IP address of the route source.
Next hop	Network layer address of the directly reachable neighboring system.
via	Interface used to reach the next hop. If there is more than one interface available to the next hop, the name of the interface that is actually used is followed by the word **Selected**. This field can also contain the following information: • Weight—Value used to distinguish primary, secondary, and fast reroute backup routes. Weight information is available when MPLS label-switched path (LSP) link protection, node-link protection, or fast reroute is enabled, or when the standby state is enabled for secondary paths. A lower weight value is preferred. Among routes with the same weight value, load balancing is possible. • Balance—Balance coefficient indicating how traffic of unequal cost is distributed among next hops when a router is performing unequal-cost load balancing. This information is available when you enable BGP multipath load balancing.
Label-switched-path *lsp-path-name*	Name of the LSP used to reach the next hop.
Label operation	MPLS label and operation occurring at this router. The operation can be **pop** (where a label is removed from the top of the stack), **push** (where another label is added to the label stack), or **swap** (where a label is replaced by another label).
Interface	(Local only) Local interface name.
Protocol next hop	Network layer address of the remote router that advertised the prefix. This address is used to derive a forwarding next hop.
Indirect next hop	Index designation used to specify the mapping between protocol next hops, tags, kernel export policy, and the forwarding next hops.
State	State of the route (a route can be in more than one state). See Table 7 on page 86.

Table 5: show route table Output Fields *(continued)*

Field Name	Field Description
Local AS	AS number of the local routing devices.
Age	How long the route has been known.
AIGP	Accumulated interior gateway protocol (AIGP) BGP attribute.
Metric*n*	Cost value of the indicated route. For routes within an AS, the cost is determined by IGP and the individual protocol metrics. For external routes, destinations, or routing domains, the cost is determined by a preference value.
MED-plus-IGP	Metric value for BGP path selection to which the IGP cost to the next-hop destination has been added.
TTL-Action	For MPLS LSPs, state of the TTL propagation attribute. Can be enabled or disabled for all RSVP-signaled and LDP-signaled LSPs or for specific VRF routing instances.
Task	Name of the protocol that has added the route.
Announcement bits	The number of BGP peers or protocols to which Junos OS has announced this route, followed by the list of the recipients of the announcement. Junos OS can also announce the route to the kernel routing table (KRT) for installing the route into the Packet Forwarding Engine, to a resolve tree, a Layer 2 VC, or even a VPN. For example, ***n*-Resolve inet** indicates that the specified route is used for route resolution for next hops found in the routing table. • ***n***—An index used by Juniper Networks customer support only.
AS path	AS path through which the route was learned. The letters at the end of the AS path indicate the path origin, providing an indication of the state of the route at the point at which the AS path originated: • **I**—IGP. • **E**—EGP. • **Recorded**—The AS path is recorded by the sample process (sampled). • **?**—Incomplete; typically, the AS path was aggregated. When AS path numbers are included in the route, the format is as follows: • **[ ]**—Brackets enclose the number that precedes the AS path. This number represents the number of ASs present in the AS path, when calculated as defined in RFC 4271. This value is used in the AS-path merge process, as defined in RFC 4893. • **[ ]**—If more than one AS number is configured on the routing device, or if AS path prepending is configured, brackets enclose the local AS number associated with the AS path. • **{ }**—Braces enclose AS sets, which are groups of AS numbers in which the order does not matter. A set commonly results from route aggregation. The numbers in each AS set are displayed in ascending order. • **( )**—Parentheses enclose a confederation. • **([ ])**—Parentheses and brackets enclose a confederation set. NOTE: In Junos OS Release 10.3 and later, the AS path field displays an unrecognized attribute and associated hexadecimal value if BGP receives attribute 128 (attribute set) and you have not configured an independent domain in any routing instance.

Table 5: show route table Output Fields *(continued)*

Field Name	Field Description
validation-state	(BGP-learned routes) Validation status of the route: • **Invalid**—Indicates that the prefix is found, but either the corresponding AS received from the EBGP peer is not the AS that appears in the database, or the prefix length in the BGP update message is longer than the maximum length permitted in the database. • **Unknown**—Indicates that the prefix is not among the prefixes or prefix ranges in the database. • **Unverified**—Indicates that the origin of the prefix is not verified against the database. This is because the database got populated and the validation is not called for in the BGP import policy, although origin validation is enabled, or the origin validation is not enabled for the BGP peers. • **Valid**—Indicates that the prefix and autonomous system pair are found in the database.
FECs bound to route	Indicates point-to-multipoint root address, multicast source address, and multicast group address when multipoint LDP (M-LDP) inband signaling is configured.
Primary Upstream	When multipoint LDP with multicast-only fast reroute (MoFRR) is configured, indicates the primary upstream path. MoFRR transmits a multicast join message from a receiver toward a source on a primary path, while also transmitting a secondary multicast join message from the receiver toward the source on a backup path.
RPF Nexthops	When multipoint LDP with MoFRR is configured, indicates the reverse-path forwarding (RPF) next-hop information. Data packets are received from both the primary path and the secondary paths. The redundant packets are discarded at topology merge points due to the RPF checks.
Label	Multiple MPLS labels are used to control MoFRR stream selection. Each label represents a separate route, but each references the same interface list check. Only the primary label is forwarded while all others are dropped. Multiple interfaces can receive packets using the same label.
weight	Value used to distinguish MoFRR primary and backup routes. A lower weight value is preferred. Among routes with the same weight value, load balancing is possible.
VC Label	MPLS label assigned to the Layer 2 circuit virtual connection.
MTU	Maximum transmission unit (MTU) of the Layer 2 circuit.
VLAN ID	VLAN identifier of the Layer 2 circuit.
Prefixes bound to route	Forwarding equivalent class (FEC) bound to this route. Applicable only to routes installed by LDP.
Communities	Community path attribute for the route. See Table 8 on page 88 for all possible values for this field.
Layer2-info: encaps	Layer 2 encapsulation (for example, VPLS).
control flags	Control flags: **none** or **Site Down**.
mtu	Maximum transmission unit (MTU) information.
Label-Base, range	First label in a block of labels and label block size. A remote PE router uses this first label when sending traffic toward the advertising PE router.
status vector	Layer 2 VPN and VPLS network layer reachability information (NLRI).

Table 5: show route table Output Fields *(continued)*

Field Name	Field Description
Accepted Multipath	Current active path when BGP multipath is configured.
Accepted LongLivedStale	The LongLivedStale flag indicates that the route was marked LLGR-stale by this router, as part of the operation of LLGR receiver mode. Either this flag or the LongLivedStaleImport flag may be displayed for a route. Neither of these flags are displayed at the same time as the Stale (ordinary GR stale) flag.
Accepted LongLivedStaleImport	The LongLivedStaleImport flag indicates that the route was marked LLGR-stale when it was received from a peer, or by import policy. Either this flag or the LongLivedStale flag may be displayed for a route. Neither of these flags are displayed at the same time as the Stale (ordinary GR stale) flag. Accept all received BGP long-lived graceful restart (LLGR) and LLGR stale routes learned from configured neighbors and import into the inet.0 routing table
ImportAccepted LongLivedStaleImport	Accept all received BGP long-lived graceful restart (LLGR) and LLGR stale routes learned from configured neighbors and imported into the inet.0 routing table The LongLivedStaleImport flag indicates that the route was marked LLGR-stale when it was received from a peer, or by import policy.
Accepted MultipathContrib	Path currently contributing to BGP multipath.
Localpref	Local preference value included in the route.
Router ID	BGP router ID as advertised by the neighbor in the open message.
Primary Routing Table	In a routing table group, the name of the primary routing table in which the route resides.
Secondary Tables	In a routing table group, the name of one or more secondary tables in which the route resides.

Table 6 on page 84 describes all possible values for the Next-hop Types output field.

Table 6: Next-hop Types Output Field Values

Next-Hop Type	Description
Broadcast (bcast)	Broadcast next hop.
Deny	Deny next hop.
Discard	Discard next hop.
Flood	Flood next hop. Consists of components called branches, up to a maximum of 32 branches. Each flood next-hop branch sends a copy of the traffic to the forwarding interface. Used by point-to-multipoint RSVP, point-to-multipoint LDP, point-to-multipoint CCC, and multicast.

Table 6: Next-hop Types Output Field Values *(continued)*

Next-Hop Type	Description
Hold	Next hop is waiting to be resolved into a unicast or multicast type.
Indexed (idxd)	Indexed next hop.
Indirect (indr)	Used with applications that have a protocol next hop address that is remote. You are likely to see this next-hop type for internal BGP (IBGP) routes when the BGP next hop is a BGP neighbor that is not directly connected.
Interface	Used for a network address assigned to an interface. Unlike the router next hop, the interface next hop does not reference any specific node on the network.
Local (locl)	Local address on an interface. This next-hop type causes packets with this destination address to be received locally.
Multicast (mcst)	Wire multicast next hop (limited to the LAN).
Multicast discard (mdsc)	Multicast discard.
Multicast group (mgrp)	Multicast group member.
Receive (recv)	Receive.
Reject (rjct)	Discard. An ICMP unreachable message was sent.
Resolve (rslv)	Resolving next hop.
Routed multicast (mcrt)	Regular multicast next hop.
Router	A specific node or set of nodes to which the routing device forwards packets that match the route prefix. To qualify as a next-hop type router, the route must meet the following criteria: • Must not be a direct or local subnet for the routing device. • Must have a next hop that is directly connected to the routing device.
Table	Routing table next hop.
Unicast (ucst)	Unicast.
Unilist (ulst)	List of unicast next hops. A packet sent to this next hop goes to any next hop in the list.

Table 7 on page 86 describes all possible values for the State output field. A route can be in more than one state (for example, **<Active NoReadvrt Int Ext>**).

Table 7: State Output Field Values

Value	Description
Accounting	Route needs accounting.
Active	Route is active.
Always Compare MED	Path with a lower multiple exit discriminator (MED) is available.
AS path	Shorter AS path is available.
Cisco Non-deterministic MED selection	Cisco nondeterministic MED is enabled, and a path with a lower MED is available.
Clone	Route is a clone.
Cluster list length	Length of cluster list sent by the route reflector.
Delete	Route has been deleted.
Ex	Exterior route.
Ext	BGP route received from an external BGP neighbor.
FlashAll	Forces all protocols to be notified of a change to any route, active or inactive, for a prefix. When not set, protocols are informed of a prefix only when the active route changes.
Hidden	Route not used because of routing policy.
IfCheck	Route needs forwarding RPF check.
IGP metric	Path through next hop with lower IGP metric is available.
Inactive reason	Flags for this route, which was not selected as best for a particular destination.
Initial	Route being added.
Int	Interior route.
Int Ext	BGP route received from an internal BGP peer or a BGP confederation peer.
Interior > Exterior > Exterior via Interior	Direct, static, IGP, or EBGP path is available.

Table 7: State Output Field Values *(continued)*

Value	Description
Local Preference	Path with a higher local preference value is available.
Martian	Route is a martian (ignored because it is obviously invalid).
MartianOK	Route exempt from martian filtering.
Next hop address	Path with lower metric next hop is available.
No difference	Path from neighbor with lower IP address is available.
NoReadvrt	Route not to be advertised.
NotBest	Route not chosen because it does not have the lowest MED.
Not Best in its group	Incoming BGP AS is not the best of a group (only one AS can be the best).
NotInstall	Route not to be installed in the forwarding table.
Number of gateways	Path with a greater number of next hops is available.
Origin	Path with a lower origin code is available.
Pending	Route pending because of a hold-down configured on another route.
Release	Route scheduled for release.
RIB preference	Route from a higher-numbered routing table is available.
Route Distinguisher	64-bit prefix added to IP subnets to make them unique.
Route Metric or MED comparison	Route with a lower metric or MED is available.
Route Preference	Route with lower preference value is available
Router ID	Path through a neighbor with lower ID is available.
Secondary	Route not a primary route.
Unusable path	Path is not usable because of one of the following conditions: • The route is damped. • The route is rejected by an import policy. • The route is unresolved.
Update source	Last tiebreaker is the lowest IP address value.

Table 8 on page 88 describes the possible values for the Communities output field.

Table 8: Communities Output Field Values

Value	Description
area-number	4 bytes, encoding a 32-bit area number. For AS-external routes, the value is **0**. A nonzero value identifies the route as internal to the OSPF domain, and as within the identified area. Area numbers are relative to a particular OSPF domain.
bandwidth: ***local AS number:link-bandwidth-number***	Link-bandwidth community value used for unequal-cost load balancing. When BGP has several candidate paths available for multipath purposes, it does not perform unequal-cost load balancing according to the link-bandwidth community unless all candidate paths have this attribute.
domain-id	Unique configurable number that identifies the OSPF domain.
domain-id-vendor	Unique configurable number that further identifies the OSPF domain.
link-bandwidth-number	Link-bandwidth number: from **0** through **4,294,967,295** (bytes per second).
local AS number	Local AS number: from **1** through **65,535**.
options	1 byte. Currently this is only used if the route type is **5** or **7**. Setting the least significant bit in the field indicates that the route carries a type 2 metric.
origin	(Used with VPNs) Identifies where the route came from.
ospf-route-type	1 byte, encoded as **1** or **2** for intra-area routes (depending on whether the route came from a type 1 or a type 2 LSA); **3** for summary routes; **5** for external routes (area number must be**0**); **7** for NSSA routes; or **129** for sham link endpoint addresses.
route-type-vendor	Displays the area number, OSPF route type, and option of the route. This is configured using the BGP extended community attribute **0x8000**. The format is ***area-number:ospf-route-type:options***.
rte-type	Displays the area number, OSPF route type, and option of the route. This is configured using the BGP extended community attribute **0x0306**. The format is ***area-number:ospf-route-type:options***.
target	Defines which VPN the route participates in; **target** has the format ***32-bit IP address:16-bit number***. For example, 10.19.0.0:100.
unknown IANA	Incoming IANA codes with a value between **0x1** and **0x7fff**. This code of the BGP extended community attribute is accepted, but it is not recognized.
unknown OSPF vendor community	Incoming IANA codes with a value above **0x8000**. This code of the BGP extended community attribute is accepted, but it is not recognized.

Sample Output

show route table bgp.l2.vpn

```
user@host> show route table bgp.l2.vpn
bgp.l2vpn.0: 1 destinations, 1 routes (1 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

192.168.24.1:1:4:1/96
                   *[BGP/170] 01:08:58, localpref 100, from 192.168.24.1
                      AS path: I
                    > to 10.0.16.2 via fe-0/0/1.0, label-switched-path am
```

show route table bgp.l3vpn.0

```
user@host> show route table bgp.l3vpn.0
      bgp.l3vpn.0: 2 destinations, 2 routes (2 active, 0 holddown, 0 hidden)
      + = Active Route, - = Last Active, * = Both

      10.255.71.15:100:10.255.71.17/32
                         *[BGP/170] 00:03:59, MED 1, localpref 100, from
      10.255.71.15
                           AS path: I
                          > via so-2/1/0.0, Push 100020, Push 100011(top)
      10.255.71.15:200:10.255.71.18/32
                         *[BGP/170] 00:03:59, MED 1, localpref 100, from
      10.255.71.15
                           AS path: I
                          > via so-2/1/0.0, Push 100021, Push 100011(top)
```

show route table bgp.l3vpn.0 detail

```
user@host> show route table bgp.l3vpn.0 detail
bgp.l3vpn.0: 8 destinations, 8 routes (8 active, 0 holddown, 0 hidden)

10.255.245.12:1:4.0.0.0/8 (1 entry, 1 announced)
  *BGP   Preference: 170/-101
         Route Distinguisher: 10.255.245.12:1
         Source: 10.255.245.12
         Next hop: 192.168.208.66 via fe-0/0/0.0, selected
         Label operation: Push 182449
         Protocol next hop: 10.255.245.12
         Push 182449
         Indirect next hop: 863a630 297
         State: <Active Int Ext>
         Local AS:    35 Peer AS:    35
         Age: 12:19      Metric2: 1
         Task: BGP_35.10.255.245.12+179
         Announcement bits (1): 0-BGP.0.0.0.0+179
         AS path: 30 10458 14203 2914 3356 I (Atomic) Aggregator: 3356 4.68.0.11

         Communities: 2914:420 target:11111:1 origin:56:78
         VPN Label: 182449
         Localpref: 100
          Router ID: 10.255.245.12

10.255.245.12:1:4.17.225.0/24 (1 entry, 1 announced)
  *BGP   Preference: 170/-101
         Route Distinguisher: 10.255.245.12:1
         Source: 10.255.245.12
         Next hop: 192.168.208.66 via fe-0/0/0.0, selected
```

```
                Label operation: Push 182465
                Protocol next hop: 10.255.245.12
                Push 182465
                Indirect next hop: 863a8f0 305
                State: <Active Int Ext>
                Local AS:    35 Peer AS:    35
                Age: 12:19      Metric2: 1
                Task: BGP_35.10.255.245.12+179
                Announcement bits (1): 0-BGP.0.0.0.0+179
AS path: 30 10458 14203 2914 11853 11853 11853 6496 6496 6496 6496 6496 6496 I
                       Communities: 2914:410 target:12:34 target:11111:1 origin:12:34
                       VPN Label: 182465
                       Localpref: 100
                       Router ID: 10.255.245.12

10.255.245.12:1:4.17.226.0/23 (1 entry, 1 announced)
  *BGP   Preference: 170/-101
                Route Distinguisher: 10.255.245.12:1
                Source: 10.255.245.12
                Next hop: 192.168.208.66 via fe-0/0/0.0, selected
                Label operation: Push 182465
                Protocol next hop: 10.255.245.12
                Push 182465
                Indirect next hop: 86bd210 330
                State: <Active Int Ext>
                Local AS:    35 Peer AS:    35
                Age: 12:19      Metric2: 1
                Task: BGP_35.10.255.245.12+179
                Announcement bits (1): 0-BGP.0.0.0.0+179
                AS path: 30 10458 14203 2914 11853 11853 11853 6496 6496 6496 6496 6496

                6496 I
                Communities: 2914:410 target:12:34 target:11111:1 origin:12:34
                VPN Label: 182465
                Localpref: 100
                Router ID: 10.255.245.12

10.255.245.12:1:4.17.251.0/24 (1 entry, 1 announced)
  *BGP   Preference: 170/-101
                Route Distinguisher: 10.255.245.12:1
                Source: 10.255.245.12
                Next hop: 192.168.208.66 via fe-0/0/0.0, selected
                Label operation: Push 182465
                Protocol next hop: 10.255.245.12
                Push 182465
                Indirect next hop: 86bd210 330
                State: <Active Int Ext>
                Local AS:    35 Peer AS:    35
                Age: 12:19      Metric2: 1
                Task: BGP_35.10.255.245.12+179
                Announcement bits (1): 0-BGP.0.0.0.0+179
                AS path: 30 10458 14203 2914 11853 11853 11853 6496 6496 6496 6496 6496

                6496 I
                Communities: 2914:410 target:12:34 target:11111:1 origin:12:34
                VPN Label: 182465
                Localpref: 100
```

show route table bgp.rtarget.0 (When Proxy BGP Route Target Filtering Is Configured)

```
user@host> show route table bgp.rtarget.o
```

```
bgp.rtarget.0: 1 destinations, 1 routes (1 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

100:100:100/96
                   *[RTarget/5] 00:03:14
                      Type Proxy
                        for 10.255.165.103
                        for 10.255.166.124
                      Local
```

show route table bgp.evpn.0

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

2:100.100.100.2:100::0::00:26:88:5f:67:b0/304
                   *[BGP/170] 11:00:05, localpref 100, from 100.100.100.2
                      AS path: I, validation-state: unverified
                    > to 100.1.12.2 via xe-2/2/0.0, label-switched-path R0toR1
2:100.100.100.2:100::0::00:51:51:51:51:51/304
                   *[BGP/170] 11:00:05, localpref 100, from 100.100.100.2
                      AS path: I, validation-state: unverified
                    > to 100.1.12.2 via xe-2/2/0.0, label-switched-path R0toR1
2:100.100.100.3:100::0::00:52:52:52:52:52/304
                   *[BGP/170] 10:59:58, localpref 100, from 100.100.100.3
                      AS path: I, validation-state: unverified
                    > to 100.1.13.3 via ge-2/0/8.0, label-switched-path R0toR2
2:100.100.100.3:100::0::a8:d0:e5:5b:01:c8/304
                   *[BGP/170] 10:59:58, localpref 100, from 100.100.100.3
                      AS path: I, validation-state: unverified
                    > to 100.1.13.3 via ge-2/0/8.0, label-switched-path R0toR2
3:100.100.100.2:100::1000::100.100.100.2/304
                   *[BGP/170] 11:00:16, localpref 100, from 100.100.100.2
                      AS path: I, validation-state: unverified
                    > to 100.1.12.2 via xe-2/2/0.0, label-switched-path R0toR1
3:100.100.100.2:100::2000::100.100.100.2/304
                   *[BGP/170] 11:00:16, localpref 100, from 100.100.100.2
                      AS path: I, validation-state: unverified
                    > to 100.1.12.2 via xe-2/2/0.0, label-switched-path R0toR1
```

show route table evpna.evpn.0

```
user@host> show route table evpna.evpn.0
evpna.evpn.0: 2 destinations, 2 routes (2 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

3:100.100.100.10:100::0::10::100.100.100.10/384
                   *[EVPN/170] 01:37:09
                      Indirect
3:100.100.100.2:100::2000::100.100.100.2/304
                   *[EVPN/170] 01:37:12
                      Indirect
```

show route table inet.0

```
user@host> show route table inet.0
inet.0: 12 destinations, 12 routes (11 active, 0 holddown, 1 hidden)
+ = Active Route, - = Last Active, * = Both

0.0.0.0/0          *[Static/5] 00:51:57
                    > to 111.222.5.254 via fxp0.0
```

```
1.0.0.1/32          *[Direct/0] 00:51:58
                     > via at-5/3/0.0
1.0.0.2/32          *[Local/0] 00:51:58
                       Local
12.12.12.21/32      *[Local/0] 00:51:57
                       Reject
13.13.13.13/32      *[Direct/0] 00:51:58
                     > via t3-5/2/1.0
13.13.13.14/32      *[Local/0] 00:51:58
                       Local
13.13.13.21/32      *[Local/0] 00:51:58
                       Local
13.13.13.22/32      *[Direct/0] 00:33:59
                     > via t3-5/2/0.0
127.0.0.1/32         [Direct/0] 00:51:58
                     > via lo0.0
111.222.5.0/24      *[Direct/0] 00:51:58
                     > via fxp0.0
111.222.5.81/32     *[Local/0] 00:51:58
                       Local
```

show route table inet6.0

```
user@host> show route table inet6.0
inet6.0: 3 destinations, 3 routes (3 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Route, * = Both

fec0:0:0:3::/64 *[Direct/0] 00:01:34
>via fe-0/1/0.0

fec0:0:0:3::/128 *[Local/0] 00:01:34
>Local

fec0:0:0:4::/64 *[Static/5] 00:01:34
>to fec0:0:0:3::ffff via fe-0/1/0.0
```

show route table inet6.3

```
user@router> show route table inet6.3
inet6.3: 2 destinations, 2 routes (2 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

::10.255.245.195/128
                   *[LDP/9] 00:00:22, metric 1
                    > via so-1/0/0.0
::10.255.245.196/128
                   *[LDP/9] 00:00:08, metric 1
                    > via so-1/0/0.0, Push 100008
```

show route table inetflow detail

```
user@host> show route table inetflow detail
inetflow.0: 2 destinations, 2 routes (2 active, 0 holddown, 0 hidden)
10.12.44.1,*/48 (1 entry, 1 announced)
        *BGP    Preference: 170/-101
                Next-hop reference count: 2
                State: <Active Ext>
                Local AS: 65002 Peer AS: 65000
                Age: 4
                Task: BGP_65000.10.12.99.5+3792
                Announcement bits (1): 0-Flow
                AS path: 65000 I
```

```
                    Communities: traffic-rate:0:0
                    Validation state: Accept, Originator: 10.12.99.5
                    Via: 10.12.44.0/24, Active
                    Localpref: 100
                    Router ID: 10.255.71.161

10.12.56.1,*/48 (1 entry, 1 announced)
        *Flow   Preference: 5
                Next-hop reference count: 2
                State: <Active>
                Local AS: 65002
                Age: 6:30
                Task: RT Flow
                Announcement bits (2): 0-Flow 1-BGP.0.0.0.0+179
                AS path: I
                Communities: 1:1
```

show route table l2circuit.0

```
user@host> show route table l2circuit.0
l2circuit.0: 4 destinations, 4 routes (4 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

10.1.1.195:NoCtrlWord:1:1:Local/96
                   *[L2CKT/7] 00:50:47
                    > via so-0/1/2.0, Push 100049
                      via so-0/1/3.0, Push 100049
10.1.1.195:NoCtrlWord:1:1:Remote/96
                   *[LDP/9] 00:50:14
                      Discard
10.1.1.195:CtrlWord:1:2:Local/96
                   *[L2CKT/7] 00:50:47
                    > via so-0/1/2.0, Push 100049
                      via so-0/1/3.0, Push 100049
10.1.1.195:CtrlWord:1:2:Remote/96
                   *[LDP/9] 00:50:14
                      Discard
```

show route table mpls

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

0                  *[MPLS/0] 00:13:55, metric 1
                      Receive
1                  *[MPLS/0] 00:13:55, metric 1
                      Receive
2                  *[MPLS/0] 00:13:55, metric 1
                      Receive
1024               *[VPN/0] 00:04:18
                      to table red.inet.0, Pop
```

show route table mpls extensive

```
user@host> show route table mpls extensive
100000 (1 entry, 1 announced)
TSI:
KRT in-kernel 100000 /36 -> {so-1/0/0.0}
        *LDP    Preference:   9
                Next hop:  via so-1/0/0.0, selected
                Pop
```

```
                    State: <Active Int>
                    Age: 29:50      Metric: 1
                    Task: LDP
                    Announcement bits (1): 0-KRT
                    AS path: I
                    Prefixes bound to route: 10.0.0.194/32
```

show route table mpls.0

```
user@host> show route table mpls.0
mpls.0: 18 destinations, 19 routes (18 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

0                  *[MPLS/0] 11:39:56, metric 1
                      to table inet.0
0(S=0)             *[MPLS/0] 11:39:56, metric 1
                      to table mpls.0
1                  *[MPLS/0] 11:39:56, metric 1
                      Receive
2                  *[MPLS/0] 11:39:56, metric 1
                      to table inet6.0
2(S=0)             *[MPLS/0] 11:39:56, metric 1
                      to table mpls.0
13                 *[MPLS/0] 11:39:56, metric 1
                      Receive
303168             *[EVPN/7] 11:00:49, routing-instance pbbn10, route-type
Ingress-MAC, ISID 0
                      to table pbbn10.evpn-mac.0
303184             *[EVPN/7] 11:00:53, routing-instance pbbn10, route-type
Ingress-IM, ISID 1000
                      to table pbbn10.evpn-mac.0
                    [EVPN/7] 11:00:53, routing-instance pbbn10, route-type
Ingress-IM, ISID 2000
                      to table pbbn10.evpn-mac.0
303264            *[EVPN/7] 11:00:53, remote-pe 100.100.100.2, routing-instance
 pbbn10, route-type Egress-IM, ISID 1000
                     > to 100.1.12.2 via xe-2/2/0.0, label-switched-path R0toR1
303280            *[EVPN/7] 11:00:53, remote-pe 100.100.100.2, routing-instance
 pbbn10, route-type Egress-IM, ISID 2000
                     > to 100.1.12.2 via xe-2/2/0.0, label-switched-path R0toR1
303328            *[EVPN/7] 11:00:49, remote-pe 100.100.100.2, routing-instance
 pbbn10, route-type Egress-MAC, ISID 0
                     > to 100.1.12.2 via xe-2/2/0.0, label-switched-path R0toR1
303344            *[EVPN/7] 11:00:49, remote-pe 100.100.100.2, routing-instance
 pbbn10, route-type Egress-MAC, ISID 0
                     > to 100.1.12.2 via xe-2/2/0.0, label-switched-path R0toR1
303360            *[EVPN/7] 11:00:47, routing-instance pbbn10, route-type
Egress-MAC, ISID 0, BMAC 00:26:88:5f:67:b0
                     > to 100.1.12.2 via xe-2/2/0.0, label-switched-path R0toR1
303376            *[EVPN/7] 11:00:47, routing-instance pbbn10, route-type
Egress-MAC, ISID 0, BMAC 00:51:51:51:51:51
                     > to 100.1.12.2 via xe-2/2/0.0, label-switched-path R0toR1
303392            *[EVPN/7] 11:00:35, remote-pe 100.100.100.3, routing-instance
 pbbn10, route-type Egress-MAC, ISID 0
                     > to 100.1.13.3 via ge-2/0/8.0, label-switched-path R0toR2
303408            *[EVPN/7] 11:00:35, remote-pe 100.100.100.3, routing-instance
 pbbn10, route-type Egress-MAC, ISID 0
                     > to 100.1.13.3 via ge-2/0/8.0, label-switched-path R0toR2
303424            *[EVPN/7] 11:00:33, routing-instance pbbn10, route-type
Egress-MAC, ISID 0, BMAC a8:d0:e5:5b:01:c8
                     > to 100.1.13.3 via ge-2/0/8.0, label-switched-path R0toR2
```

```
303440             *[EVPN/7] 11:00:33, routing-instance pbbn10, route-type
Egress-MAC, ISID 0, BMAC 00:52:52:52:52:52
                    > to 100.1.13.3 via ge-2/0/8.0, label-switched-path R0toR2
```

show route table mpls.0 detail (PTX Series)

```
user@host> show route table mpls.0 detail
ge-0/0/2.600 (1 entry, 1 announced)
        *L2VPN  Preference: 7
                Next hop type: Indirect
                Address: 0x9438f34
                Next-hop reference count: 2
                Next hop type: Router, Next hop index: 567
                Next hop: 3.0.0.1 via ge-0/0/1.0, selected
                Label operation: Push 299808
                Label TTL action: prop-ttl
                Load balance label: Label 299808:None;
                Session Id: 0x1
                Protocol next hop: 10.255.255.1
                Label operation: Push 299872 Offset: 252
                Label TTL action: no-prop-ttl
                Load balance label: Label 299872:Flow label PUSH;
                Composite next hop: 0x9438ed8 570 INH Session ID: 0x2
                Indirect next hop: 0x9448208 262142 INH Session ID: 0x2
                State: <Active Int>
                Age: 21         Metric2: 1
                Validation State: unverified
                Task: Common L2 VC
                Announcement bits (2): 0-KRT 2-Common L2 VC
                AS path: I
```

show route table mpls.0 extensive (PTX Series)

```
user@host> show route table mpls.0 extensive
ge-0/0/2.600 (1 entry, 1 announced)
TSI:
KRT in-kernel ge-0/0/2.600.0      /32 -> {composite(570)}
        *L2VPN  Preference: 7
                Next hop type: Indirect
                Address: 0x9438f34
                Next-hop reference count: 2
                Next hop type: Router, Next hop index: 567
                Next hop: 3.0.0.1 via ge-0/0/1.0, selected
                Label operation: Push 299808
                Label TTL action: prop-ttl
                Load balance label: Label 299808:None;
                Session Id: 0x1
                Protocol next hop: 10.255.255.1
                Label operation: Push 299872 Offset: 252
                Label TTL action: no-prop-ttl
                Load balance label: Label 299872:Flow label PUSH;
                Composite next hop: 0x9438ed8 570 INH Session ID: 0x2
                Indirect next hop: 0x9448208 262142 INH Session ID: 0x2
                State: <Active Int>
                Age: 47         Metric2: 1
                Validation State: unverified
                Task: Common L2 VC
                Announcement bits (2): 0-KRT 2-Common L2 VC
                AS path: I
                Composite next hops: 1
                        Protocol next hop: 10.255.255.1 Metric: 1
```

```
                                Label operation: Push 299872 Offset: 252
                                Label TTL action: no-prop-ttl
                                Load balance label: Label 299872:Flow label PUSH;
                                Composite next hop: 0x9438ed8 570 INH Session ID: 0x2
                                Indirect next hop: 0x9448208 262142 INH Session ID: 0x2
                                Indirect path forwarding next hops: 1
                                        Next hop type: Router
                                        Next hop: 3.0.0.1 via ge-0/0/1.0
                                        Session Id: 0x1
                                10.255.255.1/32 Originating RIB: inet.3
                                  Metric: 1                     Node path count: 1
                                  Forwarding nexthops: 1
                                        Nexthop: 3.0.0.1 via ge-0/0/1.0
```

show route table mpls.0 (RSVP Route—Transit LSP)

```
user@host> show route table mpls.0

mpls.0: 8 destinations, 8 routes (8 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

0                  *[MPLS/0] 00:37:31, metric 1
                      Receive
1                  *[MPLS/0] 00:37:31, metric 1
                      Receive
2                  *[MPLS/0] 00:37:31, metric 1
                      Receive
13                 *[MPLS/0] 00:37:31, metric 1
                      Receive
300352             *[RSVP/7/1] 00:08:00, metric 1
                    > to 8.64.0.106 via ge-1/0/1.0, label-switched-path lsp1_p2p
300352(S=0)        *[RSVP/7/1] 00:08:00, metric 1
                    > to 8.64.0.106 via ge-1/0/1.0, label-switched-path lsp1_p2p
300384             *[RSVP/7/2] 00:05:20, metric 1
                    > to 8.64.1.106 via ge-1/0/0.0, Pop
300384(S=0)        *[RSVP/7/2] 00:05:20, metric 1
                    > to 8.64.1.106 via ge-1/0/0.0, Pop
```

show route table vpls_1 detail

```
user@host> show route table vpls_1 detail
vpls_1.l2vpn.0: 1 destinations, 1 routes (1 active, 0 holddown, 0 hidden)
Restart Complete

1.1.1.11:1000:1:1/96 (1 entry, 1 announced)
*L2VPN Preference: 170/-1
Receive table: vpls_1.l2vpn.0
Next-hop reference count: 2
State: <Active Int Ext>
Age: 4:29:47 Metric2: 1
Task: vpls_1-l2vpn
Announcement bits (1): 1-BGP.0.0.0.0+179
AS path: I
Communities: Layer2-info: encaps:VPLS, control flags:Site-Down
Label-base: 800000, range: 8, status-vector: 0xFF
```

show route table vpn-a

```
user@host> show route table vpn-a
vpn-a.l2vpn.0: 3 destinations, 3 routes (3 active, 0 holddown, 0 hidden)

+ = Active Route, - = Last Active, * = Both
```

```
192.168.16.1:1:1:1/96
                   *[VPN/7] 05:48:27
                     Discard
192.168.24.1:1:2:1/96
                   *[BGP/170] 00:02:53, localpref 100, from 192.168.24.1
                      AS path: I
                    > to 10.0.16.2 via fe-0/0/1.0, label-switched-path am
192.168.24.1:1:3:1/96
                   *[BGP/170] 00:02:53, localpref 100, from 192.168.24.1
                      AS path: I
                    > to 10.0.16.2 via fe-0/0/1.0, label-switched-path am
```

show route table vpn-a.mdt.0

```
user@host> show route table vpn-a.mdt.0
vpn-a.mdt.0: 3 destinations, 3 routes (3 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

1:1:0:10.255.14.216:232.1.1.1/144
                   *[MVPN/70] 01:23:05, metric2 1
                      Indirect
1:1:1:10.255.14.218:232.1.1.1/144
                   *[BGP/170] 00:57:49, localpref 100, from 10.255.14.218
                      AS path: I
                    > via so-0/0/0.0, label-switched-path r0e-to-r1
1:1:2:10.255.14.217:232.1.1.1/144
                   *[BGP/170] 00:57:49, localpref 100, from 10.255.14.217
                      AS path: I
> via so-0/0/1.0, label-switched-path r0-to-r2
```

show route table VPN-A detail

```
user@host> show route table VPN-A detail
VPN-AB.inet.0: 8 destinations, 8 routes (8 active, 0 holddown, 0 hidden)
10.255.179.9/32 (1 entry, 1 announced)
        *BGP    Preference: 170/-101
                Route Distinguisher: 10.255.179.13:200
                Next hop type: Indirect
                Next-hop reference count: 5
                Source: 10.255.179.13
                Next hop type: Router, Next hop index: 732
                Next hop: 10.39.1.14 via fe-0/3/0.0, selected
                Label operation: Push 299824, Push 299824(top)
                Protocol next hop: 10.255.179.13
                Push 299824
                Indirect next hop: 8f275a0 1048574
                State: (Secondary Active Int Ext)
                Local AS: 1 Peer AS: 1
                Age: 3:41:06 Metric: 1 Metric2: 1
                Task: BGP_1.10.255.179.13+64309
                Announcement bits (2): 0-KRT 1-BGP RT Background
                AS path: I
                Communities: target:1:200 rte-type:0.0.0.0:1:0
                Import Accepted
                VPN Label: 299824 TTL Action: vrf-ttl-propagate
                Localpref: 100
                Router ID: 10.255.179.13
                Primary Routing Table bgp.l3vpn.0
```

show route table VPN-AB.inet.0

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

      10.39.1.0/30       *[OSPF/10] 00:07:24, metric 1
                          > via so-7/3/1.0
      10.39.1.4/30       *[Direct/0] 00:08:42
                          > via so-5/1/0.0
      10.39.1.6/32       *[Local/0] 00:08:46
                            Local
      10.255.71.16/32    *[Static/5] 00:07:24
                          > via so-2/0/0.0
      10.255.71.17/32    *[BGP/170] 00:07:24, MED 1, localpref 100, from
      10.255.71.15
                             AS path: I
                          > via so-2/1/0.0, Push 100020, Push 100011(top)
      10.255.71.18/32    *[BGP/170] 00:07:24, MED 1, localpref 100, from
      10.255.71.15
                             AS path: I
                          > via so-2/1/0.0, Push 100021, Push 100011(top)
      10.255.245.245/32  *[BGP/170] 00:08:35, localpref 100
                             AS path: 2 I
                          > to 10.39.1.5 via so-5/1/0.0
      10.255.245.246/32  *[OSPF/10] 00:07:24, metric 1
                          > via so-7/3/1.0
```

show route table VPN_blue.mvpn-inet6.0

```
user@host> show route table VPN_blue.mvpn-inet6.0
vpn_blue.mvpn-inet6.0: 6 destinations, 6 routes (6 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

1:10.255.2.202:65535:10.255.2.202/432
                   *[BGP/170] 00:02:37, localpref 100, from 10.255.2.202
                      AS path: I
                    > via so-0/1/3.0
1:10.255.2.203:65535:10.255.2.203/432
                   *[BGP/170] 00:02:37, localpref 100, from 10.255.2.203
                      AS path: I
                    > via so-0/1/0.0
1:10.255.2.204:65535:10.255.2.204/432
                   *[MVPN/70] 00:57:23, metric2 1
                      Indirect
5:10.255.2.202:65535:128:::192.168.90.2:128:ffff::1/432
                   *[BGP/170] 00:02:37, localpref 100, from 10.255.2.202
                      AS path: I
                    > via so-0/1/3.0
6:10.255.2.203:65535:65000:128:::10.12.53.12:128:ffff::1/432
                   *[PIM/105] 00:02:37
                      Multicast (IPv6)
7:10.255.2.202:65535:65000:128:::192.168.90.2:128:ffff::1/432
                   *[MVPN/70] 00:02:37, metric2 1
                      Indirect
```

show route table vrf1.mvpn.0 extensive

```
user@host> show route table vrf1.mvpn.0 extensive
1:10.255.50.77:1:10.255.50.77/240 (1 entry, 1 announced)
        *MVPN   Preference: 70
```

```
                PMSI: Flags 0x0: Label 0: RSVP-TE:
Session_13[10.255.50.77:0:25624:10.255.50.77]
                Next hop type: Indirect
                Address: 0xbb2c944
                Next-hop reference count: 360
                Protocol next hop: 10.255.50.77
                Indirect next hop: 0x0 - INH Session ID: 0x0
                State: <Active Int Ext>
                Age: 53:03      Metric2: 1
                Validation State: unverified
                Task: mvpn global task
                Announcement bits (3): 0-PIM.vrf1 1-mvpn global task 2-rt-export

                AS path: I
```

show route table inetflow detail

```
user@host> show route table inetflow detail
inetflow.0: 2 destinations, 2 routes (2 active, 0 holddown, 0 hidden)
10.12.44.1,*/48 (1 entry, 1 announced)
        *BGP    Preference: 170/-101
                Next-hop reference count: 2
                State: <Active Ext>
                Local AS: 65002 Peer AS: 65000
                Age: 4
                Task: BGP_65000.10.12.99.5+3792
                Announcement bits (1): 0-Flow
                AS path: 65000 I
                Communities: traffic-rate:0:0
                Validation state: Accept, Originator: 10.12.99.5
                Via: 10.12.44.0/24, Active
                Localpref: 100
                Router ID: 10.255.71.161

10.12.56.1,*/48 (1 entry, 1 announced)
        *Flow   Preference: 5
                Next-hop reference count: 2
                State: <Active>
                Local AS: 65002
                Age: 6:30
                Task: RT Flow
                Announcement bits (2): 0-Flow 1-BGP.0.0.0.0+179
                AS path: I
                Communities: 1:1

user@PE1> show route table green.l2vpn.0 (VPLS Multihoming with FEC 129)
green.l2vpn.0: 6 destinations, 6 routes (6 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

1.1.1.2:100:1.1.1.2/96 AD
                   *[VPLS/170] 1d 03:11:03, metric2 1
                         Indirect
1.1.1.4:100:1.1.1.4/96 AD
                   *[BGP/170] 1d 03:11:02, localpref 100, from 1.1.1.4
                      AS path: I, validation-state: unverified
                    >    via ge-1/2/1.5
1.1.1.2:100:1:0/96 MH
                   *[VPLS/170] 1d 03:11:03, metric2 1
                         Indirect
1.1.1.4:100:1:0/96 MH
                   *[BGP/170] 1d 03:11:02, localpref 100, from 1.1.1.4
```

```
                    AS path: I, validation-state: unverified
                  >    via ge-1/2/1.5
1.1.1.4:NoCtrlWord:5:100:100:1.1.1.2:1.1.1.4/176
                  *[VPLS/7] 1d 03:11:02, metric2 1
                   >    via ge-1/2/1.5
1.1.1.4:NoCtrlWord:5:100:100:1.1.1.4:1.1.1.2/176
                  *[LDP/9] 1d 03:11:02
                        Discard

user@host> show route table red extensive
red.inet.0: 364481 destinations, 714087 routes (364480 active, 48448 holddown, 1
 hidden)
22.0.0.0/32 (3 entries, 1 announced)
        State: <OnList CalcForwarding>
TSI:
KRT in-kernel 22.0.0.0/32 -> {composite(1048575)} Page 0 idx 1 Type 1 val 0x934342c

    Nexthop: Self
    AS path: [2] I
    Communities: target:2:1
Path 22.0.0.0 from 2.3.0.0 Vector len 4.  Val: 1
        @BGP    Preference: 170/-1
                Route Distinguisher: 2:1
                Next hop type: Indirect
                Address: 0x258059e4
                Next-hop reference count: 2
                Source: 2.2.0.0
                Next hop type: Router
                Next hop: 10.1.1.1 via ge-1/1/9.0, selected
                Label operation: Push 707633
                Label TTL action: prop-ttl
                Session Id: 0x17d8
                Protocol next hop: 2.2.0.0
                Push 16
                Composite next hop: 0x25805988 - INH Session ID: 0x193c
                Indirect next hop: 0x23eea900 - INH Session ID: 0x193c
                State: <Secondary Active Int Ext ProtectionPath ProtectionCand>
                Local AS:     2 Peer AS:     2
                Age: 23         Metric2: 35
                Validation State: unverified
                Task: BGP_2.2.2.0.0+34549
                AS path: I
                Communities: target:2:1
                Import Accepted
                VPN Label: 16
                Localpref: 0
                Router ID: 2.2.0.0
                Primary Routing Table bgp.l3vpn.0
                Composite next hops: 1
                        Protocol next hop: 2.2.0.0 Metric: 35
                        Push 16
                        Composite next hop: 0x25805988 - INH Session ID: 0x193c
                        Indirect next hop: 0x23eea900 - INH Session ID: 0x193c
                        Indirect path forwarding next hops: 1
                                Next hop type: Router
                                Next hop: 10.1.1.1 via ge-1/1/9.0
                                Session Id: 0x17d8
                        2.2.0.0/32 Originating RIB: inet.3
                          Metric: 35                    Node path count: 1
                          Forwarding nexthops: 1
                                Nexthop: 10.1.1.1 via ge-1/1/9.0
```

```
                BGP     Preference: 170/-1
                        Route Distinguisher: 2:1
                        Next hop type: Indirect
                        Address: 0x9347028
                        Next-hop reference count: 3
                        Source: 2.3.0.0
                        Next hop type: Router, Next hop index: 702
                        Next hop: 10.1.4.2 via ge-1/0/0.0, selected
                        Label operation: Push 634278
                        Label TTL action: prop-ttl
                        Session Id: 0x17d9
                        Protocol next hop: 2.3.0.0
                        Push 16
                        Composite next hop: 0x93463a0 1048575 INH Session ID: 0x17da
                        Indirect next hop: 0x91e8800 1048574 INH Session ID: 0x17da
                        State: <Secondary NotBest Int Ext ProtectionPath ProtectionCand>

                        Inactive reason: Not Best in its group - IGP metric
                        Local AS:     2 Peer AS:     2
                        Age: 3:34       Metric2: 70
                        Validation State: unverified
                        Task: BGP_2.2.3.0.0+32805
                        Announcement bits (2): 0-KRT 1-BGP_RT_Background
                        AS path: I
                        Communities: target:2:1
                        Import Accepted
                        VPN Label: 16
                        Localpref: 0
                        Router ID: 2.3.0.0
                        Primary Routing Table bgp.l3vpn.0
                        Composite next hops: 1
                                Protocol next hop: 2.3.0.0 Metric: 70
                                Push 16
                                Composite next hop: 0x93463a0 1048575 INH Session ID:
0x17da
                                Indirect next hop: 0x91e8800 1048574 INH Session ID:
0x17da
                                Indirect path forwarding next hops: 1
                                        Next hop type: Router
                                        Next hop: 10.1.4.2 via ge-1/0/0.0
                                        Session Id: 0x17d9
                                2.3.0.0/32 Originating RIB: inet.3
                                  Metric: 70                    Node path count: 1
                                  Forwarding nexthops: 1
                                        Nexthop: 10.1.4.2 via ge-1/0/0.0
                #Multipath Preference: 255
                        Next hop type: Indirect
                        Address: 0x24afca30
                        Next-hop reference count: 1
                        Next hop type: Router
                        Next hop: 10.1.1.1 via ge-1/1/9.0, selected
                        Label operation: Push 707633
                        Label TTL action: prop-ttl
                        Session Id: 0x17d8
                        Next hop type: Router, Next hop index: 702
                        Next hop: 10.1.4.2 via ge-1/0/0.0
                        Label operation: Push 634278
                        Label TTL action: prop-ttl
                        Session Id: 0x17d9
                        Protocol next hop: 2.2.0.0
                        Push 16
```

```
                    Composite next hop: 0x25805988 - INH Session ID: 0x193c
                   Indirect next hop: 0x23eea900 - INH Session ID: 0x193c Weight 0x1

                    Protocol next hop: 2.3.0.0
                    Push 16
                    Composite next hop: 0x93463a0 1048575 INH Session ID: 0x17da
                  Indirect next hop: 0x91e8800 1048574 INH Session ID: 0x17da Weight
0x4000
                    State: <ForwardingOnly Int Ext>
                    Inactive reason: Forwarding use only
                    Age: 23         Metric2: 35
                    Validation State: unverified
                    Task: RT
                    AS path: I
                    Communities: target:2:1
```