



Junos[®] OS

EVPN Control Plane and VXLAN Data Plane Feature Guide for QFX Series Switches

Release
15.1



Modified: 2016-12-19

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

Juniper Networks, Junos, Steel-Belted Radius, NetScreen, and ScreenOS are registered trademarks of Juniper Networks, Inc. in the United States and other countries. The Juniper Networks Logo, the Junos logo, and JunosE are trademarks of Juniper Networks, Inc. All other trademarks, service marks, registered trademarks, or registered service marks are the property of their respective owners.

Juniper Networks assumes no responsibility for any inaccuracies in this document. Juniper Networks reserves the right to change, modify, transfer, or otherwise revise this publication without notice.

Junos[®] OS EVPN Control Plane and VXLAN Data Plane Feature Guide for QFX Series Switches

15.1

Copyright © 2016, Juniper Networks, Inc.
All rights reserved.

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

	About the Documentation	xi
	Documentation and Release Notes	xi
	Supported Platforms	xi
	Using the Examples in This Manual	xi
	Merging a Full Example	xii
	Merging a Snippet	xii
	Documentation Conventions	xiii
	Documentation Feedback	xv
	Requesting Technical Support	xv
	Self-Help Online Tools and Resources	xv
	Opening a Case with JTAC	xvi
Part 1	Overview	
Chapter 1	EVPN and VXLAN Overview	3
	Understanding VXLANs	3
	VXLAN Benefits	3
	How Does VXLAN Work?	4
	VXLAN Implementation Methods	5
	Using QFX5100 and QFX5110 Switches with VXLANs	6
	Changing the UDP Port on QFX5100 and QFX5110 Switches	6
	Controlling Transit Multicast Traffic on QFX5100 and QFX5110 Switches	7
	Using an MX Series Router, EX9200 Switch, or QFX10000 Switch as a VTEP	7
	Manual VXLANs Require PIM	8
	Load Balancing VXLAN Traffic	8
	VXLAN Constraints on QFX Series Switches	9
	VXLAN Constraints on QFX5100 Switches	9
	VXLAN Constraints on QFX10000 Switches	11
	EVPN Overview	11
	Supported EVPN Standards	13
	Understanding EVPN with VXLAN Data Plane Encapsulation	14
	Understanding EVPN	14
	Understanding VXLAN	16
	EVPN-VXLAN Integration Overview	17
	Understanding Contrail Virtual Networks Use with EVPN-VXLAN	18
	EVPN-VXLAN Packet Format	19
	Q & A	19
	EVPN-VXLAN Constraints on QFX Series Switches	20
	Implementing EVPN-VXLAN for Data Centers	20

	Understanding EVPN Pure Route Type-5 on QFX Series Switches	24
	Defining the Five EVPN-VXLAN Route Types	24
	Implementing Pure Type-5 Routes in an EVPN-VXLAN Environment	25
	Understanding Pure Type 5-Route Forwarding	26
	Advantages of Using EVPN Pure Type-5	27
	Best Practices and Caveats	27
	EVPN Multihoming Overview	27
	Introduction to EVPN Multihoming	27
	Understanding EVPN Multihoming Concepts	29
	EVPN Multihoming Mode of Operation	31
	EVPN Multihoming Implementation	31
	New BGP NLRI	32
	New Extended Communities	34
	New EVPN Route Types	35
	Update to the MAC Forwarding Table	36
	Traffic Flow	37
	Aliasing	38
	Sample Configuration	39
	Designated Forwarder Election	40
	DF Election Roles	40
	DF Election Procedure	41
	DF Election Trigger	41
	DF Election for Virtual Switch	42
	Handling Failover	42
Part 2	Configuration	
Chapter 2	Configuring EVPN-VXLAN	47
	Example: Configuring VNI Route Targets Automatically	47
	Example: Configuring VNI Route Targets Manually	49
	Example: Configuring VNI Route Targets Automatically with Manual Override	51
	Example: Configuring IRB Interfaces in an EVPN-VXLAN Environment to Provide Layer 3 Connectivity for Hosts in a Data Center	53
	Example: Configuring EVPN-VXLAN In a Collapsed IP Fabric Topology Within a Data Center	98
	Preventing Traffic Loss in an EVPN/VXLAN Environment With GRES and NSR	108
Part 3	Configuration Statements and Operational Commands	
Chapter 3	EVPN-VXLAN Configuration Statements	113
	designated-forwarder-election-hold-time (evpn)	114
	encapsulation (Logical Interface)	115
	evpn	119
	extended-vni-list	120
	ingress-node-replication (EVPN)	121
	ip-prefix-routes	122
	ip-prefix-support	124

	multicast-mode (EVPN)	125
	no-default-gateway-ext-comm	126
	nsr-phantom-holdtime	127
	proxy-macip-advertisement	128
	route-distinguisher	129
	vni-options	131
	vrf-export	132
	vrf-import	133
	vrf-target	134
	vxlan	135
Chapter 4	EVPN-VXLAN Operational Commands	137
	show configuration protocols evpn	138
	show evpn ip-prefix-database	140
	show evpn l3-context	147
	show route table	150

List of Figures

Part 1	Overview	
Chapter 1	EVPN and VXLAN Overview	3
	Figure 1: VXLAN Packet Format	5
	Figure 2: EVPN Connecting Data Center 1 and Data Center 2	12
	Figure 3: EVPN Overview	15
	Figure 4: EVPN-VXLAN Integration Overview	18
	Figure 5: EVPN-VXLAN Packet Format	19
	Figure 6: DCI Option: Layer 3 VPN-MPLS	21
	Figure 7: DCI Option: EVPN-MPLS	21
	Figure 8: DCI Option: EVPN-VXLAN over the Internet	22
	Figure 9: DCI Option: Layer 3 VPN-MPLS Direct Connection	22
	Figure 10: EVPN-VXLAN Connection with Pure Type-5 Route Between Two Data Centers	26
	Figure 11: CE Device Multihomed to Two PE Routers	29
	Figure 12: Simple EVPN Topology	30
Part 2	Configuration	
Chapter 2	Configuring EVPN-VXLAN	47
	Figure 13: EVPN-VXLAN Topology with IRB Interfaces	55
	Figure 14: Two-Layer IP Fabric	99
	Figure 15: Collapsed IP Fabric	99
	Figure 16: Collapsed IP Fabric Topology Within a Data Center	100
	Figure 17: Method 1 and 2 IRB Interface Configurations on Multiple Leaf Devices	102

List of Tables

	About the Documentation	xi
	Table 1: Notice Icons	xiii
	Table 2: Text and Syntax Conventions	xiii
Part 1	Overview	
Chapter 1	EVPN and VXLAN Overview	3
	Table 3: CLI Commands for EVPN-VXLAN	22
Part 2	Configuration	
Chapter 2	Configuring EVPN-VXLAN	47
	Table 4: Key Entities Configured for Customer Groups 1 and 2	55
Part 3	Configuration Statements and Operational Commands	
Chapter 4	EVPN-VXLAN Operational Commands	137
	Table 5: Output Fields for Command show protocols evpn	138
	Table 6: show evpn ip-prefix-database Output Fields	141
	Table 7: show evpn l3-context Output Fields	147
	Table 8: show route table Output Fields	151
	Table 9: Next-hop Types Output Field Values	156
	Table 10: State Output Field Values	158
	Table 11: Communities Output Field Values	160

About the Documentation

- Documentation and Release Notes on page xi
- Supported Platforms on page xi
- Using the Examples in This Manual on page xi
- Documentation Conventions on page xiii
- Documentation Feedback on page xv
- Requesting Technical Support on page xv

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.xml;
    }
  }
}
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.xml; }
```

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.	<code>user@host> show chassis alarms</code> <code>No alarms currently active</code>
<i>Italic text like this</i>	<ul style="list-style-type: none"> Introduces or emphasizes important new terms. Identifies guide names. Identifies RFC and Internet draft titles. 	<ul style="list-style-type: none"> A policy <i>term</i> is a named structure that defines match conditions and actions. <i>Junos OS CLI User Guide</i> RFC 1997, <i>BGP Communities Attribute</i>
<i>Italic text like this</i>	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 <i>domain-name</i>
Text like this	Represents names of configuration statements, commands, files, and directories; configuration hierarchy levels; or labels on routing platform components.	<ul style="list-style-type: none"> 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 [<i>community-ids</i>]
Indentation and braces ({ })	Identifies a level in the configuration hierarchy.	[edit] routing-options { static { route default { nexthop <i>address</i> ; 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.	<ul style="list-style-type: none"> 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

- [EVPN and VXLAN Overview on page 3](#)

CHAPTER 1

EVPN and VXLAN Overview

- [Understanding VXLANs on page 3](#)
- [VXLAN Constraints on QFX Series Switches on page 9](#)
- [EVPN Overview on page 11](#)
- [Supported EVPN Standards on page 13](#)
- [Understanding EVPN with VXLAN Data Plane Encapsulation on page 14](#)
- [Implementing EVPN-VXLAN for Data Centers on page 20](#)
- [Understanding EVPN Pure Route Type-5 on QFX Series Switches on page 24](#)
- [EVPN Multihoming Overview on page 27](#)

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 on page 3](#)
- [How Does VXLAN Work? on page 4](#)
- [VXLAN Implementation Methods on page 5](#)
- [Using QFX5100 and QFX5110 Switches with VXLANs on page 6](#)
- [Changing the UDP Port on QFX5100 and QFX5110 Switches on page 6](#)
- [Controlling Transit Multicast Traffic on QFX5100 and QFX5110 Switches on page 7](#)
- [Using an MX Series Router, EX9200 Switch, or QFX10000 Switch as a VTEP on page 7](#)
- [Manual VXLANs Require PIM on page 8](#)
- [Load Balancing VXLAN Traffic on page 8](#)

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 32,000 VXLANs, 32,000 multicast groups, and 8000 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.
- QFX10000 switches support 4000 VXLANs and 2000 VTEPs.
- QFX5100 and QFX5110 switches support 4000 VXLANs, 4000 multicast groups, and 2000 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 do not need to use Spanning Tree Protocol (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 fields:

- Outer media access control (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)

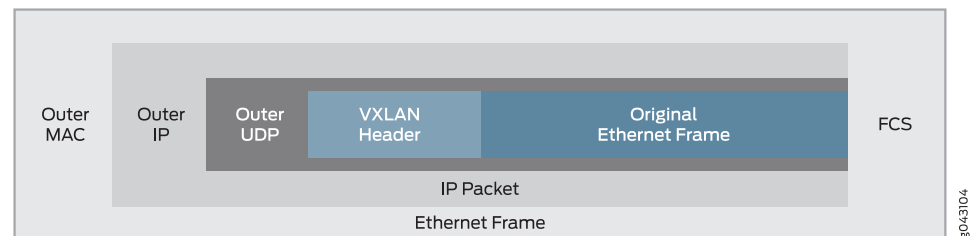
- Outer UDP header
- A VXLAN header that includes a 24-bit field—called the *VXLAN network identifier (VNI)*—that is used to uniquely identify the VXLAN. The VNI is similar to a VLAN ID, but having 24 bits allows you to create many more VXLANs than VLANs.



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:

- **Manual VXLAN**—In this environment, a Juniper Networks device acts as a transit device for downstream devices acting as VTEPs, or a gateway that provides connectivity for downstream servers that host virtual machines (VMs), which communicate over a Layer 3 network. In this environment, software-defined networking (SDN) controllers are not deployed.



NOTE: QFX10000 switches do not support manual VXLANs.

- **OVSDB-VXLAN**—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 Networks Contrail controller) and Juniper Networks devices that support OVSDB can communicate.
- **EVPN-VXLAN**—In this environment, Ethernet VPN (EVPN) is a control plane technology that enables hosts (physical servers and VMs) to be placed anywhere in a network and remain connected to the same logical Layer 2 overlay network, and VXLAN creates the data plane for the Layer 2 overlay network.

Using QFX5100 and QFX5110 Switches with VXLANs

You can configure the switches 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 8](#) for more information.)
- In an environment with or without an SDN controller, act as a Layer 2 gateway between virtualized and nonvirtualized networks in the same data center or between data centers. For example, you can use the 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 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: The QFX Series switches described in this section 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, EX9200 switch, or QFX10000 switch.

Because the additional headers add 50 to 54 bytes, you might need to increase the MTU on a 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 QFX5100 and QFX5110 Switches

Starting with Junos OS Release 14.1X53-D25 on QFX5100 switches and Junos OS Release 15.1X53-D210 on QFX5110 switches, you can configure the UDP port used as the destination port for VXLAN traffic. To configure the VXLAN destination port to be something other than the default UDP port of 4789, enter the following statement:

```
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 QFX5100 and QFX5110 Switches

When the switch acting as a 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 Release 14.1X53-D30 and later, if you know that there are no multicast receivers attached to other VTEPs in the VXLAN that 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, EX9200 Switch, or QFX10000 Switch as a VTEP

You can configure an MX Series router, EX9200 switch, or QFX10000 switch to act as a VTEP and perform all of the following roles:

- Act as a Layer 2 gateway between virtualized and nonvirtualized 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 virtual private LAN service (VPLS) tunnels.



NOTE: If you want one of the devices described in this section 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 a VMware NSX or Juniper Networks Contrail controller), 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 Protocol Independent Multicast (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, although 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 to 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 and QFX5110 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.)

Related Documentation

- *Examples: Manually Configuring VXLANs on QFX Series Switches*
- *delete*
- *OVSDDB Support on Juniper Networks Devices*
- *mtu*

VXLAN Constraints on QFX Series Switches

When configuring VXLANs on QFX Series switches, be aware of the constraints described in the following sections. In these sections, “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.

- [VXLAN Constraints on QFX5100 Switches on page 9](#)
- [VXLAN Constraints on QFX10000 Switches on page 11](#)

VXLAN Constraints on QFX5100 Switches

- You can use VXLANs on a Virtual Chassis or Virtual Chassis Fabric (VCF) if all of the members are supported QFX5100 switches. You cannot use VXLANs if any of the members is not a supported QFX5100 switch.
- VXLAN configuration is supported only in the default routing instance.
- Routing traffic between different VXLANs is not supported.
- 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 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.



NOTE: In an EVPN-VXLAN environment, multihoming active-active mode is used instead of MC-LAG for redundant connectivity between hosts and leaf devices.

- 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.
- The ability to shut down a Layer 2 interface or temporarily disable the interface when a storm control level is exceeded is not supported.
- Redundant trunk groups (RTGs).
- 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.



NOTE: An exception to this constraint is that MAC limiting is supported on OVSDB-managed interfaces in an OVSDB-VXLAN environment with Contrail controllers. For more information, see *Features Supported on OVSDB-Managed Interfaces*.

- Ingress node replication is not supported in the following cases:
 - When PIM is used for the control plane (manual VXLAN).
 - When an SDN controller is used for the control plane (OVSDB-VXLAN).

Ingress node replication is supported for EVPN-VXLAN.

- PIM-BIDIR and PIM-SSM are not supported with VXLANs.
- Class of service (CoS) features are not supported with VXLANs.



NOTE: An exception to CoS constraint is that CoS features are supported on OVSDB-managed interfaces in an OVSDB-VXLAN environment with Contrail controllers. For more information, see *Features Supported on OVSDB-Managed Interfaces*.

- If you configure a port-mirroring instance to mirror traffic exiting 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.

VXLAN Constraints on QFX10000 Switches

- MC-LAGs are not supported with VXLAN.



NOTE: In an EVPN-VXLAN environment, multihoming active-active mode is used instead of MC-LAG for redundant connectivity between hosts and leaf devices.

- IP fragmentation is 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.
- Ingress node replication is not supported when an SDN controller is used for the control plane (OVSDB-VXLAN). Ingress node replication is supported for EVPN-VXLAN.
- CoS features are not supported with VXLANs.

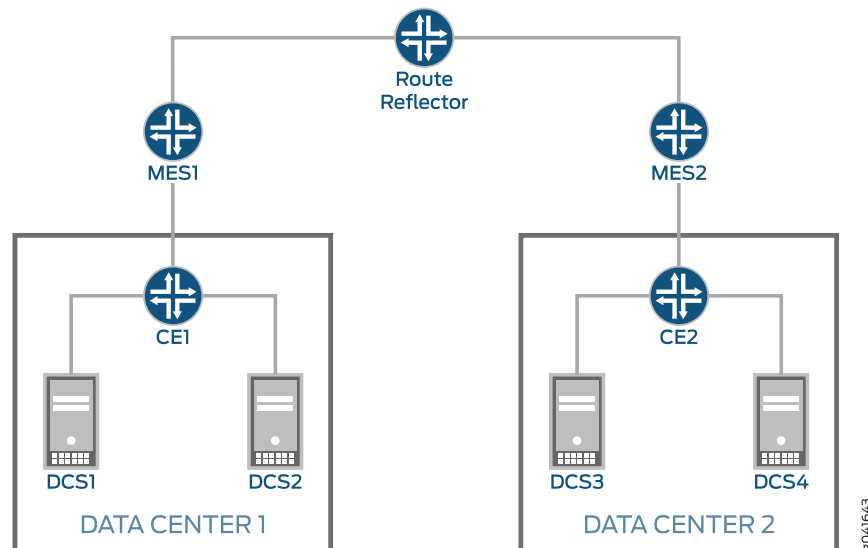
Related Documentation

- [Understanding VXLANs on page 3](#)

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 switch 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 12](#) 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:

- Active/active multihoming

**Related
Documentation**

- [Supported EVPN Standards on page 13](#)
- [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*

Understanding EVPN with VXLAN Data Plane Encapsulation

Ethernet VPN (EVPN), offers an end-to-end solution for datacenter VXLAN networks. EVPN uses a new address family, L2VPN EVPN, on the multi-protocol BGP control plane to distribute VXLAN EVPN routes that include both Layer-3 Host IP routes and Layer-2 MAC routes.

Ethernet VPNs (EVPNs) enable you to connect groups of dispersed customer sites using Layer 2 virtual bridges. Virtual Extensible LANs (VXLANs) allow you to stretch a Layer 2 connection over an intervening Layer 3 network while providing the network segmentation a VLAN provides, 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-VXLAN can be used with and without Juniper Networks Contrail virtualization software—use Contrail with EVPN-VXLAN when you have an environment that includes both virtual and bare-metal devices.



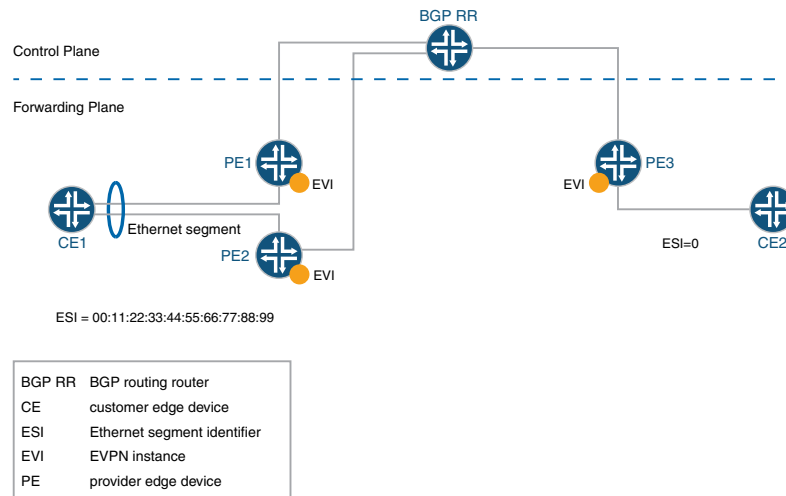
NOTE: In an EVPN-VXLAN topology, MX Series routers and QFX10000 switches can function as a provider edge (PE) device, and QFX Series switches can function as a customer edge (CE) device.

- [Understanding EVPN on page 14](#)
- [Understanding VXLAN on page 16](#)
- [EVPN-VXLAN Integration Overview on page 17](#)
- [Understanding Contrail Virtual Networks Use with EVPN-VXLAN on page 18](#)
- [EVPN-VXLAN Packet Format on page 19](#)
- [Q & A on page 19](#)
- [EVPN-VXLAN Constraints on QFX Series Switches on page 20](#)

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 15](#).

Figure 3: EVPN Overview



EVPN technology offers multitenancy, 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 media access control (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 switches—only MX Series routers support this feature.

Understanding VXLAN

Virtual eXtensible Local Area Networks (VXLAN) introduced an overlay scheme that expands the layer-2 network address space from 4K to 16 million, largely solving the scaling issues seen in VLAN-based environments.

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 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.

In the virtual network, 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 that is used for routing the UDP packets between VTEPs. For example, 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.

EVPN-VXLAN 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 router, 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.

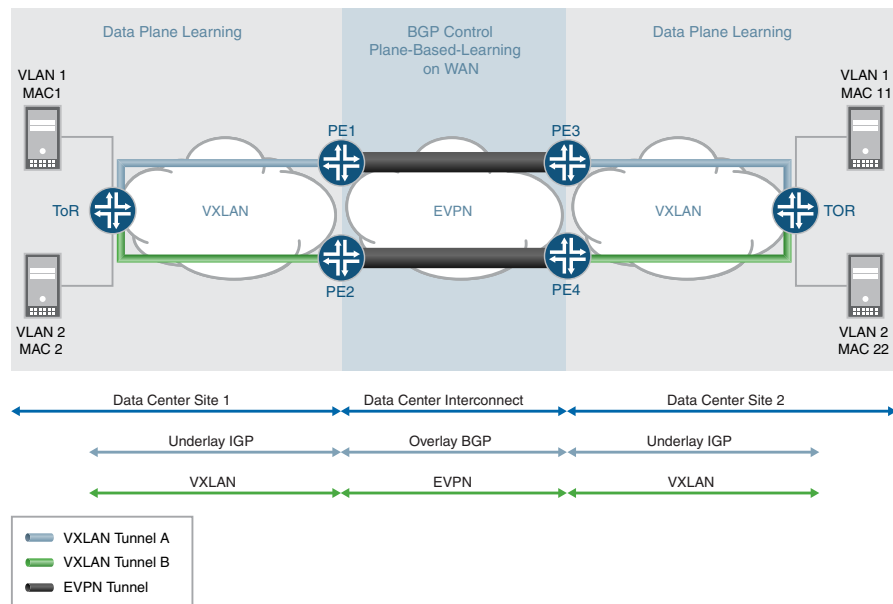
Although there are various DCI technologies available, EVPN has an added advantage over the other MPLS technologies because of its unique features, such as active/active redundancy, aliasing, and mass MAC withdrawal. As a result, to provide a solution for DCI, VXLAN is integrated with EVPN.

As shown in [“Overview” on page 1](#), every VXLAN, 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. 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.

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: EVPN-VXLAN Integration Overview



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.

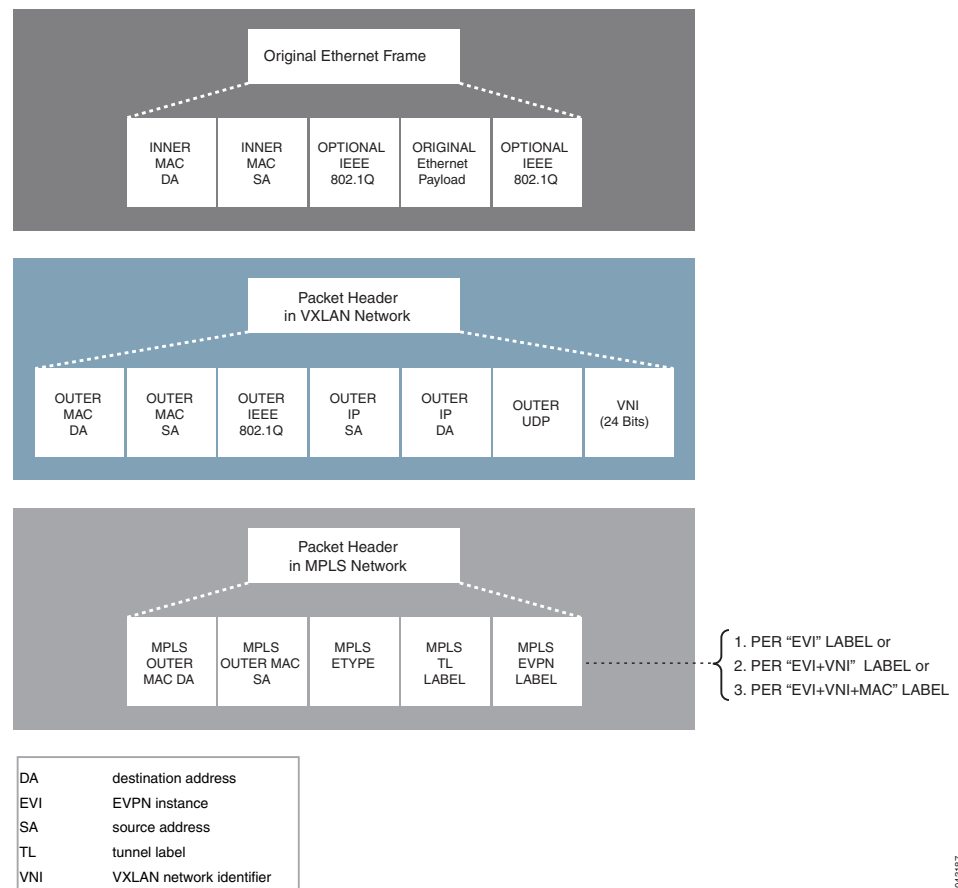
For details about Contrail support, see <http://pathfinder.juniper.net/feature-explorer> and then search on EVPN.

For more information on using EVPN-VXLAN, see [Configuring EVPN and VXLAN](#).

EVPN-VXLAN Packet Format

The EVPN-VXLAN packet format is shown in [Figure 5 on page 19](#).

Figure 5: EVPN-VXLAN Packet Format



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.

EVPN-VXLAN Constraints on QFX Series Switches

QFX Series switches do not support the following features in an EVPN-VXLAN overlay network:

- Connectivity fault management (CFM)
- Internet Group Management Protocol (IGMP) snooping
- Multichassis link aggregation groups (MC-LAGs)
- Q-in-Q tunneling
- Spanning Tree Protocol (STP) (any variant)

Related Documentation

- [EVPN Multihoming Overview on page 27](#)
- [Understanding EVPN Pure Route Type-5 on QFX Series Switches on page 24](#)
- [Understanding VXLANs on page 3](#)
- [Implementing EVPN-VXLAN for Data Centers on page 20](#)
- *Example: Configuring an EVPN Control Plane and VXLAN Data Plane on MX Series Routers and QFX5100 Switches*

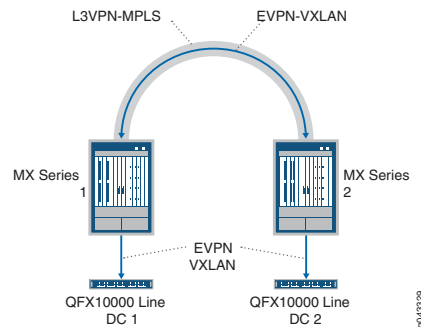
Implementing EVPN-VXLAN for Data Centers

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:

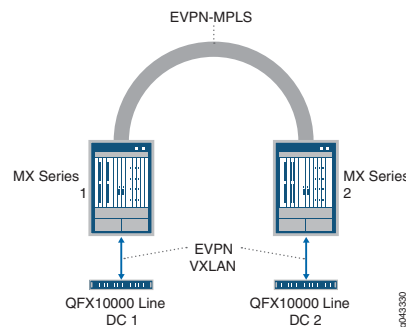
- DCI can connect multiple data centers in your WAN using MX Series edge routers with a Layer 3 VPN MPLS network between them. QFX10000 switches start and stop the VXLAN tunnel. This option requires no changes to your WAN.

Figure 6: DCI Option: Layer 3 VPN-MPLS



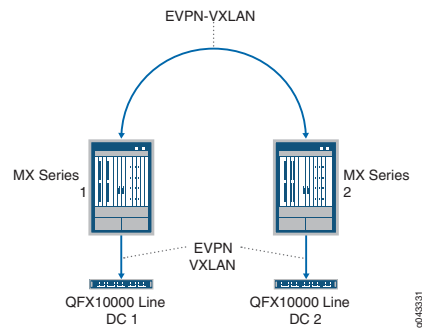
- A second option connects multiple data centers in your WAN using either MX Series edge routers or supported QFX Series switches with an EVPN-MPLS network between them. This option uses an EVPN data plane and an MPLS control plane and requires changes to your WAN. You must change your LAN architecture to natively support EVPN, and you must implement EVPN stitching between each MX router/QFX Series switch and the corresponding QFX10000 switch. For details about releases where QFX Series switches are supported, see <http://pathfinder.juniper.net/feature-explorer> and then search on EVPN.

Figure 7: DCI Option: EVPN-MPLS



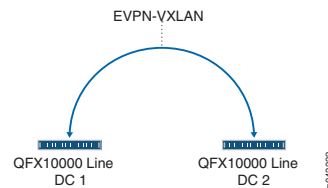
- You can also tunnel two branch locations across the Internet. In this case, implementation requires neither a traditional WAN nor MPLS. This method can use the Internet or an IP tunnel, where VXLAN rides on top of IP and EVPN is used throughout.

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: Layer 3 VPN-MPLS Direct Connection



You can alternately create an 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 a QFX10000 switch. The underlying fabric is built on BGP.

EVPN-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 Series routers are configured under a routing instance with the instance type **virtual switch**. QFX Series switches are configured under global **switching-options** and global **protocol evpn**. See [Table 3 on page 22](#) for a list of CLI commands used by EVPN-VXLAN.

Table 3: CLI Commands for EVPN-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.	<code>route-distinguisher</code>
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 QFX Series switches. The option auto applies to QFX Series switches only.	<code>vrf-target</code>

Table 3: CLI Commands for EVPN-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-VXLAN ingress replication.	extended-vni-list
You configure different route targets (RTs) for each VNI instance under vni-options .	vni-options (QFX Series switches 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 • [Understanding EVPN with VXLAN Data Plane Encapsulation on page 14](#)

Understanding EVPN Pure Route Type-5 on QFX Series Switches

Ethernet VPN (EVPN), offers an end-to-end solution for data center Virtual Extensible LAN (VXLAN) networks. A main application of EVPN is Data Center Interconnect (DCI), which provides the ability to extend Layer 2 connectivity between different data centers. EVPN uses the concept of route types to establish sessions between the provider edge and the customer edge. There are five route types. A type-5 route, also called the IP prefix route, is used to communicate between data centers (DC) when the Layer 2 connection does not extend across DCs and the IP subnet in a Layer 2 domain is confined within a single DC. In this scenario, the route type-5 enables connectivity across DCs by advertising the IP prefixes assigned to the VXLANs confined within a single DC. Data packets are sent as Layer 2 Ethernet frames encapsulated in the VXLAN header. Additionally, the gateway device for the DC must be able to perform Layer 3 routing and provide IRB functionality.



NOTE: Only pure type-5 routes are supported. Support was added in Junos OS Release 15.1x53-D30 for QFX100002 switches only. Starting with Junos OS Release 15.1x53-D60, pure type-5 routes are also supported on QFX10008 and QFX10016 switches.

A pure type-5 route operates without an overlay next hop or a type-2 route for recursive route resolution. With pure type-5 routing, the type-5 route is advertised with the MAC extended community so that the type-5 route provides all necessary forwarding information required for sending VXLAN packets in the data plane to the egress network virtual endpoint. You do not need to define IRB interfaces and use an IP address as an overlay next hop to interconnect Layer 3 virtual routing and forwarding (VRF) routes sitting in different data centers. Because no type-2 routes are used for route recursive resolution, this provisioning model is also called the IP-VRF-to-IP-VRF model without a core-facing IRB interface.

- [Defining the Five EVPN-VXLAN Route Types on page 24](#)
- [Implementing Pure Type-5 Routes in an EVPN-VXLAN Environment on page 25](#)
- [Best Practices and Caveats on page 27](#)

Defining the Five EVPN-VXLAN Route Types

The five EVPN-VXLAN route types are:

- **Route type-1, Ethernet autodiscovery route**—Type-1 routes are for networkwide messages. Ethernet autodiscovery routes are advertised on a per end virtual identifier (EVI) and per Ethernet segment identifier (ESI) basis. The Ethernet autodiscovery routes are required when a customer edge (CE) device is multihomed. When a CE device is single-homed, the ESI is zero. This route type is supported by all EVPN switches and routers.

An ESI can participate in more than one broadcast domain; for example, when a port is trunked. An ingress provider edge (PE) device that reaches the MAC on that ESI must have type-1 routes to perform split horizon and fast withdraw. Therefore, a type-1 route for an ESI must reach all ingress PE devices importing a virtual network identifier (VNI) or tag (broadcast domains) in which that ESI is a member. The Junos OS supports this by exporting a separate route target for the type-1 route.

- **Route type-2, MAC with IP advertisement route**—Type-2 routes are per-VLAN routes, so only PEs that are part of a VNI need these routes. EVPN allows an end host's IP and MAC addresses to be advertised within the EVPN Network Layer reachability information (NLRI). This allows for control plane learning of ESI MAC addresses. Because there are many type-2 routes, a separate route-target auto-derived per VNI helps to confine their propagation. This route type is supported by all EVPN switches and routers.
- **Route type-3, inclusive multicast Ethernet tag route**—Type-3 routes are per-VLAN routes; therefore, only PE devices that are part of a VNI need these routes. An inclusive multicast Ethernet tag route sets up a path for broadcast, unknown unicast, and multicast (BUM) traffic from a PE device to the remote PE device on a per-VLAN, per-ESI basis. Because there are many type-3 routes, a separate route-target auto-derived per VNI helps in confining their propagation. This route type is supported by all EVPN switches and routers.

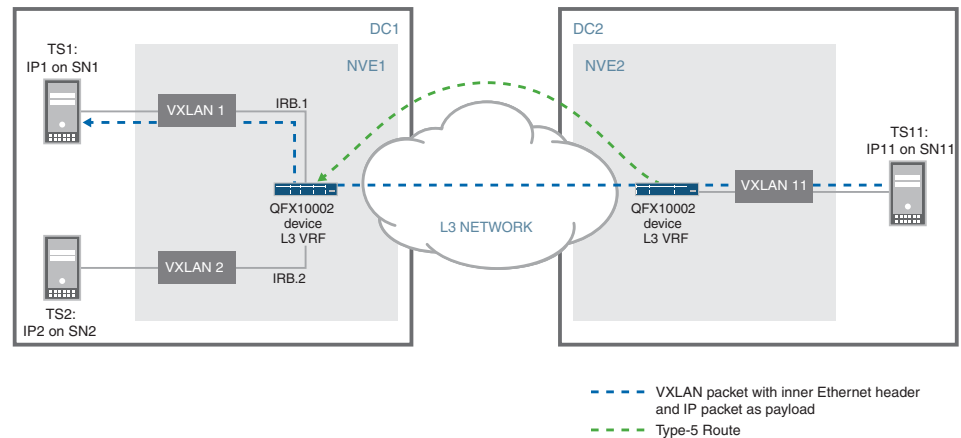
Route type-4 Ethernet segment Route—An Ethernet segment identifier (ESI) allows a CE device to be multihomed to two or more PE devices—in single/active or active/active mode. PE devices that are connected to the same Ethernet segment discover each other through the ESI. This route type is supported by all EVPN switches and routers.

- **Route type-5 IP prefix Route**—An IP prefix route provides encoding for inter-subnet forwarding. In the control plane, EVPN type-5 routes are used to advertise IP prefixes for inter-subnet connectivity across data centers. To reach a tenant using connectivity provided by the EVPN type-5 IP prefix route, data packets are sent as Layer 2 Ethernet frames encapsulated in the VXLAN header over the IP network across the data centers.

Implementing Pure Type-5 Routes in an EVPN-VXLAN Environment

You can use EVPN pure type-5 routes on QFX10000 switches to communicate between data centers through a Layer 3 network. See [Figure 10 on page 26](#). A unified EVPN control plane accomplishes L3 route advertisement between multiple data center locations so that you do not have to rely on an additional L3 VPN protocol family. On the customer edge (CE), hosts such as servers, storage devices, or any bare-metal devices are attached to leaf switches on the provider edge. Between those leaf devices, an MP-BGP session is established for EVPN routes to be used in the overlay control protocol.

Figure 10: EVPN-VXLAN Connection with Pure Type-5 Route Between Two Data Centers



A global unique virtual network identifier (VNI) is provisioned for each customer L3 VRF and identifies the customer L3 VRF at the egress. A chassis MAC is used as the inner destination MAC (DMAC) for the VXLAN packet. The chassis MAC is shared among different customer L3 VRF instances



NOTE: When a virtual machine (VM) moves from one QFX10000 data center to another, a route type-5 no longer works. This is because both the VXLAN and IP subnet that belong to the VM are no longer confined within a single data center.



NOTE: For an example of communicating within a single data center without type-5 routing, see [“Example: Configuring IRB Interfaces in an EVPN-VXLAN Environment to Provide Layer 3 Connectivity for Hosts in a Data Center”](#) on page 53.

Understanding Pure Type 5-Route Forwarding

Pure type-5 route forwarding is also called the IP-VRF-to-IP-VRF (virtual routing and forwarding) model. In IP-based computer networks, Layer 3 VRF allows multiple instances of a routing table to coexist within the same router at the same time. Because the routing instances are independent, the same or overlapping IP addresses can be used without conflicting with each other. In this scenario, for a given tenant, such as an IP VPN service, a network virtualization edge (NVE) has one MAC VRF, which consists of multiple VXLANs (one VXLAN per VLAN). The MAC VRFs on an NVE for a given tenant are associated with an IP VRF corresponding to that tenant (or IP VPN service) through their IRB interfaces. A global unique VNI is provisioned for each customer Layer 3 VRF. The VNI is used to identify the Layer 3 VRF for the customer on each data center.

Advantages of Using EVPN Pure Type-5

There are two main advantages to using EVPN pure type-5 routing:

- There is no need to exchange all host routes between data center locations. This results in smaller requirements for the routing information base (RIB), also known as the routing table, and the forwarding information base (FIB), also known as the forwarding table, on DCI equipment.
- There is no need to use multiple protocol families, such as both EVPN and an L3 VPN, to advertise L2 and L3 reachability information.

Best Practices and Caveats



BEST PRACTICE: You can use pure route type-5 within a single data center to interconnect points of delivery (pods) as long as the IP prefix can be confined within the pod.



BEST PRACTICE: Note that there are differences between EVPN VXLAN and EVPN MPLS. EVPN VXLAN exports a separate route target for type-1 routes. EVPN-MPLS exports the type-1 route with the collective set of route-targets of the VNI or tags (broadcast domains) in which the Ethernet segment identifier is participating.



NOTE: You cannot use Contrail with pure route type-5.

Related Documentation

- [Understanding EVPN with VXLAN Data Plane Encapsulation on page 14](#)
- [ip-prefix-routes on page 122](#)
- [ip-prefix-support on page 124](#)

EVPN Multihoming Overview

- [Introduction to EVPN Multihoming on page 27](#)
- [Understanding EVPN Multihoming Concepts on page 29](#)
- [EVPN Multihoming Mode of Operation on page 31](#)
- [EVPN Multihoming Implementation on page 31](#)
- [Designated Forwarder Election on page 40](#)

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.



NOTE: Prior to Junos OS Release 15.1, EVPN functionality support on MX Series routers was limited to routers using MPC and MIC interfaces only. However, starting with Junos OS Release 15.1, MX Series routers using DPCs can be leveraged to provide EVPN support on the CE device-facing interface.

DPC support for EVPN is provided with the following considerations:

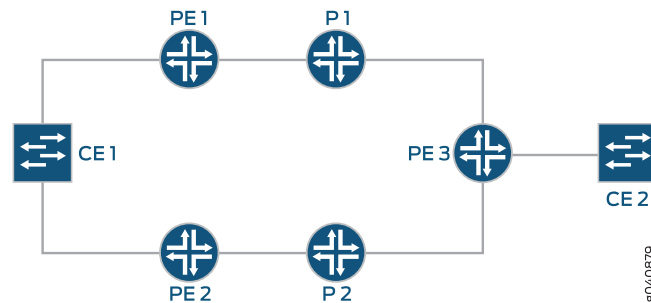
- DPCs provide support for EVPN in the active-standby mode of operation including support for the following:
 - EVPN instance (EVI)
 - Virtual switch (VS)
 - Integrated routing and bridging (IRB) interfaces
- DPCs intended for providing the EVPN active-standby support should be the CE device-facing line card. The PE device interfaces in the EVPN domain should use only MPC and MIC interfaces.

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 11 on page 29](#) 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, and depending on the multihoming mode of redundancy, only one path or all paths are active at any one time. The multihoming mode of operation also determines which PE router or routers forward traffic to the CE device. The PE router forwarding traffic to the CE device (also called a designated forwarder) uses MPLS LSP or GRE tunnels to forward traffic. If a failure occurs over this path, a new designated forwarder is elected to forward the traffic to Device CE1.

Figure 11: CE Device Multihomed to Two PE Routers



Understanding EVPN Multihoming Concepts

Figure 12 on page 30 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.

In active-standby multihoming, the links that constitute an Ethernet segment form a bridge domain. In active-active multihoming, an Ethernet segment appears as a LAG to the CE device.

- **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 manually configuring an ESI value, the most significant octet, known as the type byte, must be 00. When a single-homed CE device is attached to an Ethernet segment, the entire 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.

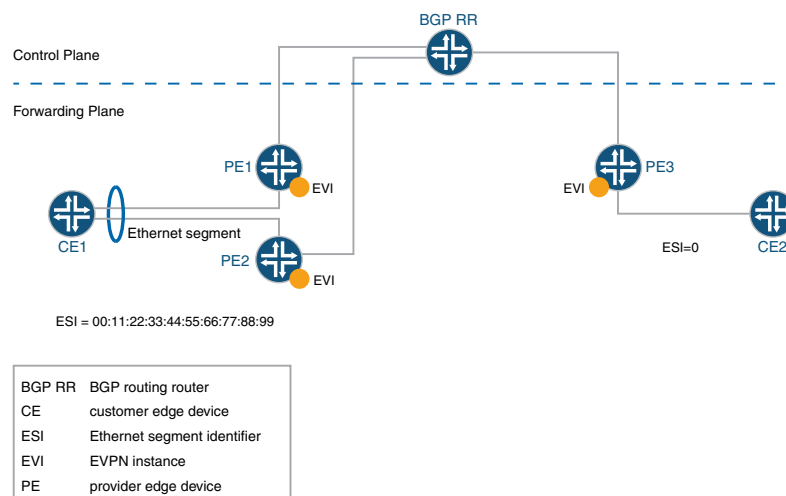
- **Ethernet tag**—An Ethernet tag identifies a particular broadcast domain, such as a VLAN. An EVPN instance consists of one or more broadcast domains. Ethernet tags are assigned to the broadcast domains of a given EVPN instance by the provider of that EVPN. Each PE router in that EVPN instance performs 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, either one or all of the multihomed PE routers are used to reach the customer site depending on the multihoming mode of operation. 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 12: 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 the **single-active** statement under the **[edit interfaces]** hierarchy level.

- **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: 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.

EVPN 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, and 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 NLRI on page 32](#)
- [New Extended Communities on page 34](#)
- [New EVPN Route Types on page 35](#)
- [Update to the MAC Forwarding Table on page 36](#)
- [Traffic Flow on page 37](#)
- [Aliasing on page 38](#)
- [Sample Configuration on page 39](#)

New BGP NLRI

To support EVPN multihoming, the following new BGP network layer reachability information (NLRI) routes have been introduced:

- [Autodiscovery Route per Ethernet Segment on page 32](#)
- [Ethernet Segment Route on page 33](#)

Autodiscovery Route per Ethernet Segment

- [Autodiscovery Route Features on page 32](#)
- [Autodiscovery Route Advertisement on page 32](#)
- [Autodiscovery Route Withdrawal on page 32](#)

Autodiscovery Route Features

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.

Autodiscovery Route Advertisement

- Active-standby mode

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.

Autodiscovery Route Withdrawal

The autodiscovery route per Ethernet segment withdrawal is also referred to as the mass withdrawal. The mass withdrawal feature is used when there is a link failure on the ESI, or when the ESI configuration changes.

When the link between a multihomed CE device and a multihomed PE device fails, the PE device withdraws the autodiscovery route per Ethernet segment. In such a case, the mass withdrawal feature is handled in the following ways by the other PE devices:

- Remote PE device

When a remote PE device receives the BGP update for mass withdrawal, the following is performed at the remote PE device:

1. The current next hop to reach the remote ESI or CE device is deleted.
2. A new next hop through the remaining multihomed PE devices is created to reach the remote ESI or CE device.
3. All the MAC routes behind the CE device are updated with the newly created next hop.

- Other multihomed PE device

As a result of the mass withdrawal, load balancing on the multihomed CE device happens because of the following:

- When the other multihomed PE devices receive the same set of MAC addresses on the link to the concerned ESI.

In this case, the local routes are preferred. If the remote routes learned from the DF PE device gets withdrawn, it does not affect routes pointing to the local ESI.

- When the other multihomed PE devices have not received the same set of MAC addresses on the link to the concerned ESI.

In this case, the PE devices install the MAC routes pointing to the concerned ESI, although the MACs are remotely learned from the DF PE device. When the DF PE device withdraws these routes, the withdrawn routes are flushed. Packets that are destined to the flushed MAC addresses are flooded on all the local segments.

Ethernet Segment Route

- [Ethernet Segment Route Features on page 33](#)
- [Ethernet Segment Route Advertisement on page 34](#)

Ethernet Segment Route Features

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.

Ethernet Segment Route Advertisement

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.

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, the following extended communities have been introduced:

- [ESI-Import on page 34](#)
- [Split Horizon on page 34](#)

ESI-Import

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

- Split horizon signaling

The split horizon 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.

- Split horizon advertisement
 - In active-standby mode, the standby bit in the split horizon extended community is set to 1, and the ESI split horizon label is set to 0.
- Split horizon MPLS routes

The DF PE device advertises an autodiscovery route per Ethernet segment with a split horizon label A, and an inclusive multicast route with label B for BUM traffic forwarding. On the DF, the BUM packet from the core can come with following labels:

- When the non-DF PE devices receive a BUM packet on their single-homed ESIs, the BUM packet is sent to the DF PE device with multicast label B.
- When the non-DF PE devices receive a BUM packet on ESI1, the BUM packet is sent to the DF PE device with two MPLS labels — the multicast label B as the outer label, and the split horizon label A as the inner label.

In the EVPN multihoming scenario, the multicast label B has the S-bit set to 1 when it is the only label in the label stack. In this case, the BUM packet needs to be flooded on all the local ESIs on the DF PE device. But the label B has the S-bit set to 0 when split horizon label A is the innermost label in the label stack. In this case, the BUM packets need to be flooded on all local ESIs on the DF PE device, except the ESI that maps to the split horizon label A.

Assuming that packets originated from a multihomed CE device to a non-DF PE device on multihomed segment ESI1, when the non-DF PE device sends this packet to the DF PE device, the ESI label that the DF advertised to the non-DF PE device in its autodiscovery route per Ethernet segment is pushed first. The non-DF PE device also pushes the inclusive multicast label that the DF PE device advertised in its inclusive multicast route and further pushes the LSP label. The MPLS header thus contains two labels within a 32-bit field.

The base EVPN functionality uses a table-next hop to stitch the MPLS table with its corresponding EVPN EVI table. In the EVPN EVI table, the mac-lookup is performed to switch the packet.

The following routes are programmed in the mpls.0 table for EVPN multicast:

- The (multicast-label, S=1) route points to the EVPN-EVI table-next hop.
- The (multicast-label, S=0) route points to the MPLS table-next hop. This route loops the packet back to the MPLS table after popping the multicast-label.
- The (split horizon-label) route points to the EVPN-EVI table-next hop. This is the same table-next hop that is used by the multicast-label, S=1 route.

New EVPN Route Types

EVPN multihoming mode supports the following EVPN route types:

- Autodiscovery route per Ethernet segment
- Autodiscovery route per EVPN instance (EVI)
- 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

Autodiscovery route per EVI—1:100.100.100.1:1::22222222222222222222::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.
 - 1—Autodiscovery route per EVI.
 - 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 and autodiscovery route per EVI—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 EVPN multihoming, traffic flow is performed in the forwarding-plane. Flood routes are created for flooding the packets, and are used in the following scenarios:

- When a packet is received on a local ESI
- When a packet is received from the core

The traffic flows in EVPN multihoming can be based on the two traffic types:

- Unicast traffic

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

- In active-standby mode
 - CE to core—Traffic is learned and forwarded by the DF PE router.
 - 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

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:

- In active-standby mode
 - 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.
 - 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.

Aliasing

Aliasing is the ability of a remote PE device to load balance Layer 2 unicast traffic on all the other PE devices that have same Ethernet segment towards a CE device.

- [Aliasing and Autodiscovery Routes on page 38](#)
- [Aliasing and Label Route on page 38](#)
- [Aliasing and Unicast Packet Forwarding on page 39](#)

Aliasing and Autodiscovery Routes

Autodiscovery routes from Routers PE2 and PE3 can come in any order. As a result, these routes are installed by the Layer 2 process as follows:

1. After receiving MAC1 from Router PE1, and if any autodiscovery routes have not been received by Router PE4, MAC1 is programmed by PE4 with a next hop pointing toward Router PE1. When PE4 receives the autodiscovery route from Router PE2 for the same ESI, the next hop is installed so the traffic for MAC1 is load-balanced to Routers PE1 and PE2. When PE4 receives the autodiscovery route from Router PE3 for the same ESI, the next hop is updated to load-balance the traffic for MAC1 among Routers PE1, PE2, and PE3.
2. If Router PE4 has already received the autodiscovery routes from more than one PE device (PE1, PE2, and PE3), PE4 installs the MAC routes with the multi-destination next hop.

Aliasing and Label Route

Any PE device that advertises the autodiscovery route per EVI with a valid MPLS label programs the advertised label in the mpls.0 routing table. For instance, if Router PE2 advertised the autodiscovery route per EVI with label A, the mpls.0 entry is as follows:

Label A route points to the EVPN-EVI table-next hop.

When the remote Router PE4 sends a unicast data packet toward Router PE2 with this label A, lookup is done in Router PE2's forwarding table, and as a result of this lookup, the packet is forwarded on ESI1.

Aliasing and Unicast Packet Forwarding

When the unicast packets for MAC1 come from the remote Router PE4 to Router PE2, there could be two cases:

- Router PE2 also received the same set of MACs on its link to ES11—In this case, local routes are preferred and as a result of the MAC lookup, packets are forwarded to ES11.
- Router PE2 has not received the same set of MACs on its link to ES11—In this case, Router PE2 still installs MAC routes pointing to ES11, although MACs are remotely learned from Router PE1. As a result, the packets are forwarded to ES11.

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 Ethernet 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 on page 40](#)
- [DF Election Procedure on page 41](#)
- [DF Election Trigger on page 41](#)
- [DF Election for Virtual Switch on page 42](#)
- [Handling Failover on page 42](#)

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.

DF Election for Virtual Switch

The virtual switch allows multiple bridge domains in a single EVPN instance (EVI). The virtual switch also supports trunk and access ports. Junos OS allows flexible Ethernet services on the port, therefore different VLANs on a single port can be part of different EVIs.

The DF election for virtual switch depends on the following:

- Port mode—Sub-interface, trunk interface, and access port
- EVI mode—Virtual switch with EVPN, EVPN-EVI

In the virtual switch, multiple Ethernet tags can be associated with a single EVI, wherein the numerically lowest Ethernet tag value in the EVI is used for the DF election.

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 42](#)
- [BUM Traffic on page 42](#)

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.

Release History Table

Release	Description
14.X53-D30	Starting with Junos OS Release 14.1X53-D30, QFX5100 switches support the active-active mode of operation for EVPN multihoming.

**Related
Documentation**

- *Example: Configuring EVPN Active-Standby Multihoming*

PART 2

Configuration

- [Configuring EVPN-VXLAN on page 47](#)

CHAPTER 2

Configuring EVPN-VXLAN

- [Example: Configuring VNI Route Targets Automatically on page 47](#)
- [Example: Configuring VNI Route Targets Manually on page 49](#)
- [Example: Configuring VNI Route Targets Automatically with Manual Override on page 51](#)
- [Example: Configuring IRB Interfaces in an EVPN-VXLAN Environment to Provide Layer 3 Connectivity for Hosts in a Data Center on page 53](#)
- [Example: Configuring EVPN-VXLAN In a Collapsed IP Fabric Topology Within a Data Center on page 98](#)
- [Preventing Traffic Loss in an EVPN/VXLAN Environment With GRES and NSR on page 108](#)

Example: Configuring VNI Route Targets Automatically

This example shows how to automatically derive route targets for multiple VNIs in an EVPN-VXLAN topology.

- [Requirements on page 47](#)
- [Overview on page 47](#)
- [Configuration on page 48](#)

Requirements

This example uses the following hardware and software components:

- A QFX Series switch.
- Junos OS version 15.1X53-D30

Overview

The **vrf-target** statement can be used to configure specific route targets for each VNI under **vni-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 on page 48](#)
- [Results on page 48](#)

Configuring VNI Route Target Automatic Derivation

Step-by-Step Procedure

To configure the automatic derivation of VNI route targets:

1. At the **[switch-options]** hierarchy level, configure the **vtep-source-interface**, and **route-distinguisher** 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 100.0
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 two 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 policy-statement 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

Use the **show** command to verify the results of your configuration.

```
user@switch> show configuration switch-options
vtep-source-interface 100.0;
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 134](#)
 - [Example: Configuring VNI Route Targets Manually on page 49](#)
 - [Example: Configuring VNI Route Targets Automatically with Manual Override on page 51](#)

Example: Configuring VNI Route Targets Manually

This example shows how to manually set route targets for multiple VNIs in an EVPN-VXLAN topology.

- [Requirements on page 49](#)
- [Overview on page 49](#)
- [Configuration on page 50](#)

Requirements

This example uses the following hardware and software components:

- A QFX Series switch.
- Junos OS version 15.1X53-D30

Overview

The **vrf-target** statement can be used to configure specific route targets for each VNI under **vni-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 on page 50](#)
- [Results on page 50](#)

Configuring VNI Route Targets Manually

Step-by-Step Procedure

To manually configure VNI route targets:

1. At the **[switch-options]** hierarchy level, configure the **vtep-source-interface**, and **route-distinguisher** 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 134](#)
- [Example: Configuring VNI Route Targets Automatically on page 47](#)
- [Example: Configuring VNI Route Targets Automatically with Manual Override on page 51](#)

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 in an EVPN-VXLAN topology.

- [Requirements on page 51](#)
- [Overview on page 51](#)
- [Configuration on page 51](#)

Requirements

This example uses the following hardware and software components:

- A QFX Series switch.
- Junos OS version 15.1X53-D30

Overview

The **vrf-target** statement can be used to configure specific route targets for each VNI under **vni-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

To configure automatic VNI route targets with manual override:

1. At the **[switch-options]** hierarchy level, configure the **vtep-source-interface**, and **route-distinguisher** 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 three 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 policy-statement 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 134](#)
- [Example: Configuring VNI Route Targets Automatically on page 47](#)
- [Example: Configuring VNI Route Targets Manually on page 49](#)

Example: Configuring IRB Interfaces in an EVPN-VXLAN Environment to Provide Layer 3 Connectivity for Hosts in a Data Center

You can use integrated routing and bridging (IRB) interfaces to route data packets between Virtual Extensible LANs (VXLANs) that are implemented in an Ethernet VPN (EVPN)-VXLAN environment. In this example, the IRB interfaces provide Layer 3 connectivity for physical servers, on which applications directly run, in the same data center.

- [Requirements on page 54](#)
- [Overview on page 54](#)
- [Spine 1: Underlay Network Configuration on page 56](#)
- [Spine 1: Overlay Network Configuration on page 58](#)
- [Spine 1: Customer Profile Configuration on page 59](#)
- [Spine 2: Underlay Network Configuration on page 64](#)
- [Spine 2: Overlay Network Configuration on page 66](#)
- [Spine 2: Customer Profile Configuration on page 67](#)
- [Leaf 1: Underlay Network Configuration on page 71](#)
- [Leaf 1: Overlay Network Configuration on page 73](#)
- [Leaf 1: Customer Profile Configuration on page 74](#)
- [Leaf 2: Underlay Network Configuration on page 76](#)
- [Leaf 2: Overlay Network Configuration on page 78](#)
- [Leaf 2: Customer Profile Configuration on page 79](#)
- [Leaf 3: Underlay Network Configuration on page 82](#)
- [Leaf 3: Overlay Network Configuration on page 84](#)
- [Leaf 3: Customer Profile Configuration on page 85](#)
- [Leaf 4: Underlay Network Configuration on page 87](#)
- [Leaf 4: Overlay Network Configuration on page 89](#)
- [Leaf 4: Customer Profile Configuration on page 90](#)
- [Verification on page 93](#)

Requirements

This example uses the following hardware and software components:

- Two spine devices:
 - A QFX10008 switch (Spine 1) running Junos OS Release 15.1X53-D30 software.
 - A QFX10002 switch (Spine 2) running Junos OS Release 15.1X53-D30 software.
- Four QFX5100 switches (Leaf 1 through Leaf 4) running Junos OS Release 14.1X53-D30 software.
- Leaf 1 and Leaf 2 that are connected to one physical server, and Leaf 3 and Leaf 4 that are connected to another physical server.

Overview

In this example, physical servers that support two groups in a customer's organization must exchange large volumes of data. To meet this need, the following protocols are implemented:

- EVPN is used to establish a Layer 2 virtual bridge to connect the two physical servers.
- Within the EVPN topology, BGP is used to exchange route information.
- VXLAN is used to tunnel the data packets through the underlying Layer 3 fabric. The use of VXLAN encapsulation preserves the Layer 3 fabric for use by routing protocols.
- IRB interfaces are used to route data packets between the VXLANs.

In this example, IRB interfaces are configured on the spine devices only. With this configuration, the spine devices function as Layer 3 gateways for the physical servers that are connected through their respective leaf devices. When the physical servers in the two customer sites exchange data, the spine devices route the data packets to their respective destinations.

Topology

The simple IP Clos topology shown in [Figure 13 on page 55](#) includes two spine switches, four leaf switches, and two physical servers. Each leaf switch has a connection to each of the spine switches for redundancy. Physical server A supports group 1, and physical server B supports group 2 in the same customer organization. The customer network is segmented into four VXLANs. VXLANs v0001 and v0002 support group 1, and VXLANs v0005 and v0006 support group 2. [Table 4 on page 55](#) shows key entities, including IRB interfaces, that are configured for customer groups 1 and 2. Each VXLAN is supported by an IRB interface, over which data packets from the other VXLANs are routed.

Figure 13: EVPN-VXLAN Topology with IRB Interfaces

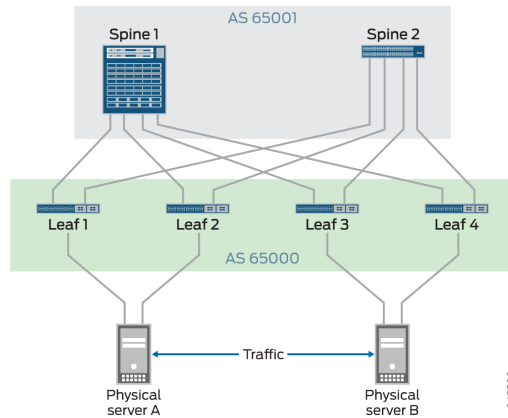


Table 4: Key Entities Configured for Customer Groups 1 and 2

Entity	Entity for Customer Group 1	Entity for Customer Group 2
VXLAN	v0001	v0005
	v0002	v0006
VNI	1	5
	2	6
IRB interface	irb.1	irb.5
	irb.2	irb.6

When configuring the entities in [Table 4 on page 55](#), keep the following in mind:

- Each VXLAN must be assigned an IP address with a different IP subnet than the other VXLANs.
- Each VXLAN must be assigned a unique VXLAN network identifier (VNI).
- Each IRB interface must be specified as part of a Layer 3 virtual routing forwarding (VRF) instance.
- The configuration of each IRB interface must include a default gateway address, which you can specify by including the **virtual-gateway-address** configuration statement at the **[interfaces irb unit logical-unit-number family inet address ip-address]** hierarchy level. Configuring the default gateway sets up a redundant default gateway for each IRB interface.
- Policies must be set up to enable the exchange of data packets between the four VXLANs.



NOTE: When setting up a routing instance for EVPN-VXLAN, you must include a loopback interface. You must also specify an IP address for the interface using the `set interfaces lo0 unit logical-unit-number family inet address ip-address/prefix`. If you inadvertently omit the loopback interface and associated IP address, EVPN control packets cannot be processed.

Spine 1: Underlay Network Configuration

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

```
set interfaces et-1/0/0 vlan-tagging
set interfaces et-1/0/0 unit 0 vlan-id 1
set interfaces et-1/0/0 unit 0 family inet address 10.1.24.10/24
set interfaces et-1/0/1 unit 0 family inet address 10.1.25.10/24
set interfaces et-1/0/4 unit 0 family inet address 10.1.28.10/24
set interfaces et-1/0/5 unit 0 family inet address 10.1.29.10/24
set interfaces et-2/0/0 vlan-tagging
set interfaces et-2/0/0 unit 0 vlan-id 1
set interfaces et-2/0/0 unit 0 family inet address 10.1.30.10/24
set interfaces et-2/0/1 unit 0 family inet address 10.1.31.10/24
set interfaces et-2/0/4 unit 0 family inet address 10.1.34.10/24
set interfaces et-2/0/5 unit 0 family inet address 10.1.35.10/24
set interfaces lo0 unit 0 family inet address 10.0.0.9/32 primary
set routing-options router-id 10.0.0.9
set routing-options autonomous-system 65001
set routing-options forwarding-table export load-balancing-policy
set protocols bgp multihop
set protocols bgp group underlay type external
set protocols bgp group underlay accept-remote-nexthop
set protocols bgp group underlay advertise-peer-as
set protocols bgp group underlay family inet unicast loops 5
set protocols bgp group underlay export send-direct
set protocols bgp group underlay peer-as 65000
set protocols bgp group underlay multipath
set protocols bgp group underlay neighbor 10.1.24.3
set protocols bgp group underlay neighbor 10.1.25.4
set protocols bgp group underlay neighbor 10.1.28.7
set protocols bgp group underlay neighbor 10.1.29.8
set protocols bgp group underlay neighbor 10.1.30.3
set protocols bgp group underlay neighbor 10.1.31.4
set protocols bgp group underlay neighbor 10.1.34.7
set protocols bgp group underlay neighbor 10.1.35.8
set policy-options policy-statement load-balancing-policy then load-balance per-packet
set policy-options policy-statement send-direct term 10 from protocol direct
set policy-options policy-statement send-direct term 10 then accept
```


Spine 1: Configure the Underlay Network

Step-by-Step Procedure

To configure the underlay network on Spine 1:

1. Configure Layer 2 and Layer 3 interfaces.

```
[edit interfaces]
user@switch# set et-1/0/0 vlan-tagging
user@switch# set et-1/0/0 unit 0 vlan-id 1
user@switch# set et-1/0/0 unit 0 family inet address 10.1.24.10/24
user@switch# set et-1/0/1 unit 0 family inet address 10.1.25.10/24
user@switch# set et-1/0/4 unit 0 family inet address 10.1.28.10/24
user@switch# set et-1/0/5 unit 0 family inet address 10.1.29.10/24
user@switch# set et-2/0/0 vlan-tagging
user@switch# set et-2/0/0 unit 0 vlan-id 1
user@switch# set et-2/0/0 unit 0 family inet address 10.1.30.10/24
user@switch# set et-2/0/1 unit 0 family inet address 10.1.31.10/24
user@switch# set et-2/0/4 unit 0 family inet address 10.1.34.10/24
user@switch# set et-2/0/5 unit 0 family inet address 10.1.35.10/24
```

2. Specify an IP address for the loopback interface. This IP address serves as the source IP address in the outer header of any VXLAN-encapsulated packets.

```
[edit interfaces]
user@switch# set lo0 unit 0 family inet address 10.0.0.9/32 primary
```

3. Set the routing options.

```
[edit routing-options]
user@switch# set router-id 10.0.0.10
user@switch# set autonomous-system 65001
user@switch# set forwarding-table export load-balancing-policy
```

4. Configure multihop external BGP (EBGP).

```
[edit protocols]
user@switch# set bgp multihop
```

5. Configure a BGP group that includes peers that handle underlay functions.

```
[edit protocols]
user@switch# set bgp group underlay type external
user@switch# set bgp group underlay accept-remote-nexthop
user@switch# set bgp group underlay advertise-peer-as
user@switch# set bgp group underlay family inet unicast loops 5
user@switch# set bgp group underlay export send-direct
user@switch# set bgp group underlay peer-as 65000
user@switch# set bgp group underlay multipath
user@switch# set bgp group underlay neighbor 10.1.24.3
user@switch# set bgp group underlay neighbor 10.1.25.4
user@switch# set bgp group underlay neighbor 10.1.28.7
user@switch# set bgp group underlay neighbor 10.1.29.8
user@switch# set bgp group underlay neighbor 10.1.30.3
user@switch# set bgp group underlay neighbor 10.1.31.4
user@switch# set bgp group underlay neighbor 10.1.34.7
user@switch# set bgp group underlay neighbor 10.1.35.8
```



NOTE: You must enter the set protocols bgp group underlay accept-remote-nexthop command so that the BGP routes appear in the routing table as direct routes. VXLAN requires direct next hops for equal-cost multipath (ECMP) load balancing.

6. Configure a per-packet load-balancing policy.

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

7. Configure a policy that enables BGP to advertise direct interfaces such as loopback interface lo0.

```
[edit policy-options]
user@switch# set policy-statement send-direct term 10 from protocol direct
user@switch# set policy-statement send-direct term 10 then accept
```

Spine 1: Overlay Network Configuration

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

```
set protocols bgp group evpn type internal
set protocols bgp group evpn local-address 10.0.0.10
set protocols bgp group evpn family evpn signaling
set protocols bgp group evpn local-as 65000
set protocols bgp group evpn multipath
set protocols bgp group evpn neighbor 10.0.0.7
set protocols bgp group evpn neighbor 10.0.0.8
set protocols bgp group evpn neighbor 10.0.0.9
set protocols bgp group evpn neighbor 10.0.0.4
set protocols bgp group evpn neighbor 10.0.0.3
set protocols evpn encapsulation vxlan
set protocols evpn multicast-mode ingress-replication
set policy-options policy-statement EVPN_VRF_IMPORT term switch_options_comm
from community switch_options_comm
set policy-options policy-statement EVPN_VRF_IMPORT term switch_options_comm
then accept
set policy-options community switch_options_comm members target:65000:2
set switch-options vtep-source-interface lo0.0
set switch-options route-distinguisher 10.0.0.10:1
set switch-options vrf-import EVPN_VRF_IMPORT
set switch-options vrf-target target:65000:2
set switch-options vrf-target auto
```

Configuring the Overlay Network on Spine 1

Step-by-Step Procedure

To configure the overlay network on Spine 1:

1. Configure an internal BGP (iBGP) group for the EVPN-VXLAN overlay.


```
[edit protocols]
user@switch# set bgp group evpn type internal
user@switch# set bgp group evpn local-address 10.0.0.10
user@switch# set bgp group evpn family evpn signaling
user@switch# set bgp group evpn local-as 65000
user@switch# set bgp group evpn multipath
user@switch# set bgp group evpn neighbor 10.0.0.7
user@switch# set bgp group evpn neighbor 10.0.0.8
user@switch# set bgp group evpn neighbor 10.0.0.9
user@switch# set bgp group evpn neighbor 10.0.0.4
user@switch# set bgp group evpn neighbor 10.0.0.3
```
2. Configure VXLAN encapsulation for the data packets exchanged between the EVPN neighbors.


```
[edit protocols]
user@switch# set evpn encapsulation vxlan
```
3. Specify a tunnel type used for passing multicast traffic between Juniper Networks switches in an EVPN-VXLAN environment.


```
[edit protocols]
user@switch# set evpn multicast-mode ingress-replication
```
4. Set up a policy to handle Layer 2 EVPN traffic.


```
[edit policy-options]
user@switch# set policy-statement EVPN_VRF_IMPORT term switch_options_comm
  from community switch_options_comm
user@switch# set policy-statement EVPN_VRF_IMPORT term switch_options_comm
  then accept
user@switch# set community switch_options_comm members target:65000:2
```
5. Configure options for the default routing instance (virtual switch type).


```
[edit switch-options]
user@switch# set vtep-source-interface lo0.0
user@switch# set route-distinguisher 10.0.0.10:1
user@switch# set vrf-import EVPN_VRF_IMPORT
user@switch# set vrf-target target:65000:2
user@switch# set vrf-target auto
```

Spine 1: Customer Profile Configuration

CLI Quick Configuration

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



NOTE: When setting up a routing instance for EVPN-VXLAN, you must include a loopback interface. You must also specify an IP address for the interface using the `set interfaces lo0 unit logical-unit-number family inet address ip-address/prefix`. If you inadvertently omit the loopback interface and associated IP address, EVPN control packets cannot be processed.

```

set interfaces irb unit 1 family inet address 10.2.1.10/24 virtual-gateway-address 10.2.1.254
set interfaces irb unit 2 family inet address 10.2.2.10/24 virtual-gateway-address 10.2.2.254
set interfaces irb unit 5 family inet address 10.2.5.10/24 virtual-gateway-address 10.2.5.254
set interfaces irb unit 6 family inet address 10.2.6.10/24 virtual-gateway-address 10.2.6.254
set interfaces lo0 unit 1 family inet address 127.10.0.1/32
set interfaces lo0 unit 2 family inet address 127.10.0.2/32
set protocols evpn extended-vni-list [ 1 2 5 6 ]
set protocols evpn vni-options vni 1 vrf-target export target:10003:1
set protocols evpn vni-options vni 2 vrf-target export target:10003:2
set protocols evpn vni-options vni 5 vrf-target export target:10003:5
set protocols evpn vni-options vni 6 vrf-target export target:10003:6
set policy-options policy-statement EVPN_VRF_IMPORT term cust0001 from community
    cust0001
set policy-options policy-statement EVPN_VRF_IMPORT term cust0001 then accept
set policy-options policy-statement EVPN_VRF_IMPORT term cust0002 from community
    cust0002
set policy-options policy-statement EVPN_VRF_IMPORT term cust0002 then accept
set policy-options policy-statement EVPN_VRF_IMPORT term vni0001 from community
    vni0001
set policy-options policy-statement EVPN_VRF_IMPORT term vni0001 then accept
set policy-options policy-statement EVPN_VRF_IMPORT term vni0002 from community
    vni0002
set policy-options policy-statement EVPN_VRF_IMPORT term vni0002 then accept
set policy-options policy-statement EVPN_VRF_IMPORT term vni0005 from community
    vni0005
set policy-options policy-statement EVPN_VRF_IMPORT term vni0005 then accept
set policy-options policy-statement EVPN_VRF_IMPORT term vni0006 from community
    vni0006
set policy-options policy-statement EVPN_VRF_IMPORT term vni0006 then accept
set policy-options community vni0001 members target:10003:1
set policy-options community vni0002 members target:10003:2
set policy-options community vni0005 members target:10003:5
set policy-options community vni0006 members target:10003:6
set routing-instances cust0001 instance-type vrf
set routing-instances cust0001 interface irb.1
set routing-instances cust0001 interface irb.2
set routing-instances cust0001 interface lo0.1
set routing-instances cust0001 route-distinguisher 10.0.0.10:2001
set routing-instances cust0001 vrf-target target:10001:1
set routing-instances cust0001 routing-options auto-export
set routing-instances cust0002 instance-type vrf
set routing-instances cust0002 interface irb.5
set routing-instances cust0002 interface irb.6
set routing-instances cust0002 interface lo0.2
set routing-instances cust0002 route-distinguisher 10.0.0.10:2002
set routing-instances cust0002 vrf-target target:10001:2

```

```

set routing-instances cust0002 routing-options auto-export
set vlans v0001 vlan-id 1
set vlans v0001 l3-interface irb.1
set vlans v0001 vxlan vni 1
set vlans v0001 vxlan ingress-node-replication
set vlans v0002 vlan-id 2
set vlans v0002 l3-interface irb.2
set vlans v0002 vxlan vni 2
set vlans v0002 vxlan ingress-node-replication
set vlans v0005 vlan-id 5
set vlans v0005 l3-interface irb.5
set vlans v0005 vxlan vni 5
set vlans v0005 vxlan ingress-node-replication
set vlans v0006 vlan-id 6
set vlans v0006 l3-interface irb.6
set vlans v0006 vxlan vni 6
set vlans v0006 vxlan ingress-node-replication
set policy-options policy-statement cust0001_vrf_imp term 1 from community cust0002
set policy-options policy-statement cust0001_vrf_imp term 1 then accept
set policy-options policy-statement cust0002_vrf_imp term 1 from community cust0001
set policy-options policy-statement cust0002_vrf_imp term 1 then accept
set policy-options community cust0001 members target:10001:1
set policy-options community cust0002 members target:10001:2
set routing-instances cust0001 vrf-import cust0001_vrf_imp
set routing-instances cust0002 vrf-import cust0002_vrf_imp

```

Step-by-Step Procedure

To configure profiles for the two customer groups:

1. Configure the IRB interfaces that enable communication among VXLANs 1, 2, 5, and 6.

```

[edit interfaces]
user@switch# set irb unit 1 family inet address 10.2.1.10/24 virtual-gateway-address 10.2.1.254
user@switch# set irb unit 2 family inet address 10.2.2.10/24 virtual-gateway-address 10.2.2.254
user@switch# set irb unit 5 family inet address 10.2.5.10/24 virtual-gateway-address 10.2.5.254
user@switch# set irb unit 6 family inet address 10.2.6.10/24 virtual-gateway-address 10.2.6.254

```

2. Configure a loopback interface for each customer group.

```

[edit interfaces]
user@switch# set lo0 unit 1 family inet address 127.10.0.1/32
user@switch# set lo0 unit 2 family inet address 127.10.0.2/32

```

3. Specify which VNIs are included in the EVPN-VXLAN domains.

```

[edit protocols]
user@switch# set evpn extended-vni-list [ 1 2 5 6 ]

```

4. Configure a route target for each VNI.

```

[edit protocols]
user@switch# set evpn vni-options vni 1 vrf-target export target:10003:1
user@switch# set evpn vni-options vni 2 vrf-target export target:10003:2

```

```

user@switch# set evpn vni-options vni 5 vrf-target export target:10003:5
user@switch# set evpn vni-options vni 6 vrf-target export target:10003:6

```

5. Configure policies that accept and import overlay routes.

```

[edit policy-options]
user@switch# set policy-statement EVPN_VRF_IMPORT term cust0001 from
community cust0001
user@switch# set policy-statement EVPN_VRF_IMPORT term cust0001 then accept
user@switch# set policy-statement EVPN_VRF_IMPORT term cust0002 from
community cust0002
user@switch# set policy-statement EVPN_VRF_IMPORT term cust0002 then accept
user@switch# set policy-statement EVPN_VRF_IMPORT term vni0001 from
community vni0001
user@switch# set policy-statement EVPN_VRF_IMPORT term vni0001 then accept
user@switch# set policy-statement EVPN_VRF_IMPORT term vni0002 from
community vni0002
user@switch# set policy-statement EVPN_VRF_IMPORT term vni0002 then accept
user@switch# set policy-statement EVPN_VRF_IMPORT term vni0005 from
community vni0005
user@switch# set policy-statement EVPN_VRF_IMPORT term vni0005 then accept
user@switch# set policy-statement EVPN_VRF_IMPORT term vni0006 from
community vni0006
user@switch# set policy-statement EVPN_VRF_IMPORT term vni0006 then accept

```

6. Set up communities for the customer groups and VXLANs. .

```

[edit policy-options]
user@switch# set community vni0001 members target:10003:1
user@switch# set community vni0002 members target:10003:2
user@switch# set community vni0005 members target:10003:5
user@switch# set community vni0006 members target:10003:6

```

7. Set up a routing instance for customer group 1.



NOTE: When setting up a routing instance for EVPN-VXLAN, you must include a loopback interface. You must also specify an IP address for the interface using the `set interfaces lo0 unit logical-unit-number family inet address ip-address/prefix`. If you inadvertently omit the loopback interface and associated IP address, EVPN control packets cannot be processed.

```

[edit routing-instances]
user@switch# set cust0001 instance-type vrf
user@switch# set cust0001 interface irb.1
user@switch# set cust0001 interface irb.2
user@switch# set cust0001 interface lo0.1
user@switch# set cust0001 route-distinguisher 10.0.0.10:2001
user@switch# set cust0001 vrf-target target:10001:1
user@switch# set cust0001 routing-options auto-export

```

8. Set up a routing instance for customer group 2.



NOTE: When setting up a routing instance for EVPN-VXLAN, you must include a loopback interface. You must also specify an IP address for the interface using the `set interfaces lo0 unit logical-unit-number family inet address ip-address/prefix`. If you inadvertently omit the loopback interface and associated IP address, EVPN control packets cannot be processed.

```
[edit routing-instances]
user@switch# set cust0002 instance-type vrf
user@switch# set cust0002 interface irb.5
user@switch# set cust0002 interface irb.6
user@switch# set cust0002 interface lo0.2
user@switch# set cust0002 route-distinguisher 10.0.0.10:2002
user@switch# set cust0002 vrf-target target:10001:2
user@switch# set cust0002 routing-options auto-export
```

9. Configure VXLANs v0001, v0002, v0005, and v0006, and associate the corresponding VNIs and IRB interfaces with each VXLAN.

```
[edit vlans]
user@switch# set v0001 vlan-id 1
user@switch# set v0001 l3-interface irb.1
user@switch# set v0001 vxlan vni 1
user@switch# set v0001 vxlan ingress-node-replication
user@switch# set v0002 vlan-id 2
user@switch# set v0002 l3-interface irb.2
user@switch# set v0002 vxlan vni 2
user@switch# set v0002 vxlan ingress-node-replication
user@switch# set v0005 vlan-id 5
user@switch# set v0005 l3-interface irb.5
user@switch# set v0005 vxlan vni 5
user@switch# set v0005 vxlan ingress-node-replication
user@switch# set v0006 vlan-id 6
user@switch# set v0006 l3-interface irb.6
user@switch# set v0006 vxlan vni 6
user@switch# set v0006 vxlan ingress-node-replication
```

10. Configure the importing of routes from one customer VRF instance into the other.

```
[edit policy-options]
user@switch# set policy-statement cust0001_vrf_imp term 1 from community
cust0002
user@switch# set policy-statement cust0001_vrf_imp term 1 then accept
user@switch# set policy-statement cust0002_vrf_imp term 1 from community
cust0001
user@switch# set policy-statement cust0002_vrf_imp term 1 then accept
user@switch# set community cust0001 members target:10001:1
user@switch# set community cust0002 members target:10001:2
[edit routing-instances]
user@switch# set cust0001 vrf-import cust0001_vrf_imp
user@switch# set cust0002 vrf-import cust0002_vrf_imp
```

Spine 2: Underlay Network Configuration

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

```
set interfaces et-0/0/0 vlan-tagging
set interfaces et-0/0/0 unit 0 vlan-id 1
set interfaces et-0/0/0 unit 0 family inet address 10.1.12.9/24
set interfaces et-0/0/2 unit 0 family inet address 10.1.13.9/24
set interfaces et-0/0/8 unit 0 family inet address 10.1.4.9/24
set interfaces et-0/0/9 unit 0 family inet address 10.1.40.9/24
set interfaces et-0/0/10 unit 0 family inet address 10.1.5.9/24
set interfaces et-0/0/11 unit 0 family inet address 10.1.41.9/24
set interfaces lo0 unit 0 family inet address 10.0.0.9/32 primary
set routing-options router-id 10.0.0.9
set routing-options autonomous-system 65001
set routing-options forwarding-table export load-balancing-policy
set protocols bgp multihop
set protocols bgp group underlay type external
set protocols bgp group underlay accept-remote-nexthop
set protocols bgp group underlay advertise-peer-as
set protocols bgp group underlay family inet unicast loops 5
set protocols bgp group underlay export send-direct
set protocols bgp group underlay peer-as 65000
set protocols bgp group underlay multipath
set protocols bgp group underlay neighbor 10.1.4.7
set protocols bgp group underlay neighbor 10.1.5.8
set protocols bgp group underlay neighbor 10.1.12.3
set protocols bgp group underlay neighbor 10.1.13.4
set protocols bgp group underlay neighbor 10.1.40.7
set protocols bgp group underlay neighbor 10.1.41.8
set policy-options policy-statement load-balancing-policy then load-balance per-packet
set policy-options policy-statement send-direct term 10 from protocol direct
set policy-options policy-statement send-direct term 10 then accept
```

Configuring the Underlay Network on Spine 2

Step-by-Step Procedure To configure the underlay network on Spine 2:

1. Configure Layer 2 and Layer 3 interfaces.

```
[edit interfaces]
set et-0/0/0 vlan-tagging
set et-0/0/0 unit 0 vlan-id 1
set et-0/0/0 unit 0 family inet address 10.1.12.9/24
set et-0/0/2 unit 0 family inet address 10.1.13.9/24
set et-0/0/8 unit 0 family inet address 10.1.4.9/24
set et-0/0/9 unit 0 family inet address 10.1.40.9/24
set et-0/0/10 unit 0 family inet address 10.1.5.9/24
set et-0/0/11 unit 0 family inet address 10.1.41.9/24
```


- Specify an IP address for the loopback interface. This IP address serves as the source IP address in the outer header of any VXLAN-encapsulated packets.

```
[edit interfaces]
user@switch# set lo0 unit 0 family inet address 10.0.0.9/32 primary
```

- Set the routing options.

```
[edit routing-options]
user@switch# set router-id 10.0.0.9
user@switch# set autonomous-system 65001
user@switch# set forwarding-table export load-balancing-policy
```

- Configure multihop EBGP.

```
[edit protocols]
user@switch# set bgp multihop
```

- Configure a BGP group that includes peers that handle underlay functions.

```
[edit protocols]
user@switch# set bgp group underlay type external
user@switch# set bgp group underlay accept-remote-nexthop
user@switch# set bgp group underlay advertise-peer-as
user@switch# set bgp group underlay family inet unicast loops 5
user@switch# set bgp group underlay export send-direct
user@switch# set bgp group underlay peer-as 65000
user@switch# set bgp group underlay multipath
user@switch# set bgp group underlay neighbor 10.1.4.7
user@switch# set bgp group underlay neighbor 10.1.5.8
user@switch# set bgp group underlay neighbor 10.1.12.3
user@switch# set bgp group underlay neighbor 10.1.13.4
user@switch# set bgp group underlay neighbor 10.1.40.7
user@switch# set bgp group underlay neighbor 10.1.41.8
```



NOTE: You must enter the `set protocols bgp group underlay accept-remote-nexthop` command so that the BGP routes appear in the routing table as direct routes. VXLAN requires direct next hops for equal-cost multipath (ECMP) load balancing.

- Configure a policy that spreads traffic across multiple paths between the Juniper Networks switches.

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

- Configure a policy that enables BGP to advertise direct interfaces such as loopback interface lo0.

```
[edit policy-options]
user@switch# set policy-statement send-direct term 10 from protocol direct
user@switch# set policy-statement send-direct term 10 then accept
```

Spine 2: Overlay Network Configuration

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

```
set protocols bgp group evpn type internal
set protocols bgp group evpn local-address 10.0.0.9
set protocols bgp group evpn family evpn signaling
set protocols bgp group evpn local-as 65000
set protocols bgp group evpn multipath
set protocols bgp group evpn neighbor 10.0.0.7
set protocols bgp group evpn neighbor 10.0.0.8
set protocols bgp group evpn neighbor 10.0.0.3
set protocols bgp group evpn neighbor 10.0.0.4
set protocols bgp group evpn neighbor 10.0.0.10
set protocols evpn encapsulation vxlan
set protocols evpn multicast-mode ingress-replication
set policy-options policy-statement EVPN_VRF_IMPORT term switch_options_comm
  from community switch_options_comm
set policy-options policy-statement EVPN_VRF_IMPORT term switch_options_comm
  then accept
set policy-options community switch_options_comm members target:65000:2
set switch-options vtep-source-interface lo0.0
set switch-options route-distinguisher 10.0.0.9:1
set switch-options vrf-import EVPN_VRF_IMPORT
set switch-options vrf-target target:65000:2
set switch-options vrf-target auto
```

Configuring the Overlay Network on Spine 2

Step-by-Step Procedure To configure the overlay network on Spine 2:

1. Configure an IBGP group for the EVPN-VXLAN overlay.

```
[edit protocols]
user@switch# set bgp group evpn type internal
user@switch# set bgp group evpn local-address 10.0.0.9
user@switch# set bgp group evpn family evpn signaling
user@switch# set bgp group evpn local-as 65000
user@switch# set bgp group evpn multipath
user@switch# set bgp group evpn neighbor 10.0.0.7
user@switch# set bgp group evpn neighbor 10.0.0.8
user@switch# set bgp group evpn neighbor 10.0.0.10
user@switch# set bgp group evpn neighbor 10.0.0.4
user@switch# set bgp group evpn neighbor 10.0.0.3
```

2. Configure VXLAN encapsulation for the data packets exchanged between the EVPN neighbors.

```
[edit protocols]
user@switch# set evpn encapsulation vxlan
```

- Specify a tunnel type used for passing multicast traffic between Juniper Networks switches in an EVPN-VXLAN environment.

```
[edit protocols]
user@switch# set evpn multicast-mode ingress-replication
```

- Set up a policy to handle Layer 2 EVPN traffic.

```
[edit policy-options]
user@switch# set policy-statement EVPN_VRF_IMPORT term switch_options_comm
  from community switch_options_comm
user@switch# set policy-statement EVPN_VRF_IMPORT term switch_options_comm
  then accept
user@switch# set community switch_options_comm members target:65000:2
```

- Configure options for the default routing instance (virtual switch type).

```
[edit switch-options]
user@switch# set vtep-source-interface lo0.0
user@switch# set route-distinguisher 10.0.0.9:1
user@switch# set vrf-import EVPN_VRF_IMPORT
user@switch# set vrf-target target:65000:2
user@switch# set vrf-target auto
```

Spine 2: Customer Profile Configuration

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



NOTE: When setting up a routing instance for EVPN-VXLAN, you must include a loopback interface. You must also specify an IP address for the interface using the `set interfaces lo0 unit logical-unit-number family inet address ip-address/prefix`. If you inadvertently omit the loopback interface and associated IP address, EVPN control packets cannot be processed.

```
set interfaces irb unit 1 family inet address 10.2.1.9/24 virtual-gateway-address 10.2.1.254
set interfaces irb unit 2 family inet address 10.2.2.9/24 virtual-gateway-address 10.2.2.254
set interfaces irb unit 5 family inet address 10.2.5.9/24 virtual-gateway-address 10.2.5.254
set interfaces irb unit 6 family inet address 10.2.6.9/24 virtual-gateway-address 10.2.6.254
set interfaces lo0 unit 1 family inet address 127.9.0.1/32
set interfaces lo0 unit 2 family inet address 127.9.0.2/32
set protocols evpn extended-vni-list [ 1 2 5 6 ]
set protocols evpn vni-options vni 1 vrf-target export target:10003:1
set protocols evpn vni-options vni 2 vrf-target export target:10003:2
set protocols evpn vni-options vni 5 vrf-target export target:10003:5
set protocols evpn vni-options vni 6 vrf-target export target:10003:6
set policy-options policy-statement EVPN_VRF_IMPORT term cust0001 from community
  cust0001
set policy-options policy-statement EVPN_VRF_IMPORT term cust0001 then accept
```

```
set policy-options policy-statement EVPN_VRF_IMPORT term cust0002 from community
  cust0002
set policy-options policy-statement EVPN_VRF_IMPORT term cust0002 then accept
set policy-options policy-statement EVPN_VRF_IMPORT term vni0001 from community
  vni0001
set policy-options policy-statement EVPN_VRF_IMPORT term vni0001 then accept
set policy-options policy-statement EVPN_VRF_IMPORT term vni0002 from community
  vni0002
set policy-options policy-statement EVPN_VRF_IMPORT term vni0002 then accept
set policy-options policy-statement EVPN_VRF_IMPORT term vni0005 from community
  vni0005
set policy-options policy-statement EVPN_VRF_IMPORT term vni0005 then accept
set policy-options policy-statement EVPN_VRF_IMPORT term vni0006 from community
  vni0006
set policy-options policy-statement EVPN_VRF_IMPORT term vni0006 then accept
set policy-options community vni0001 members target:10003:1
set policy-options community vni0002 members target:10003:2
set policy-options community vni0005 members target:10003:5
set policy-options community vni0006 members target:10003:6
set routing-instances cust0001 instance-type vrf
set routing-instances cust0001 interface irb.1
set routing-instances cust0001 interface irb.2
set routing-instances cust0001 interface lo0.1
set routing-instances cust0001 route-distinguisher 10.0.0.9:2001
set routing-instances cust0001 vrf-target target:10001:1
set routing-instances cust0001 routing-options auto-export
set routing-instances cust0002 instance-type vrf
set routing-instances cust0002 interface irb.5
set routing-instances cust0002 interface irb.6
set routing-instances cust0002 interface lo0.2
set routing-instances cust0002 route-distinguisher 10.0.0.9:2002
set routing-instances cust0002 vrf-target target:10001:2
set routing-instances cust0002 routing-options auto-export
set vlans v0001 vlan-id 1
set vlans v0001 l3-interface irb.1
set vlans v0001 vxlan vni 1
set vlans v0001 vxlan ingress-node-replication
set vlans v0002 vlan-id 2
set vlans v0002 l3-interface irb.2
set vlans v0002 vxlan vni 2
set vlans v0002 vxlan ingress-node-replication
set vlans v0005 vlan-id 5
set vlans v0005 l3-interface irb.5
set vlans v0005 vxlan vni 5
set vlans v0005 vxlan ingress-node-replication
set vlans v0006 vlan-id 6
set vlans v0006 l3-interface irb.6
set vlans v0006 vxlan vni 6
set vlans v0006 vxlan ingress-node-replication
set policy-options policy-statement cust0001_vrf_imp term 1 from community cust0002
set policy-options policy-statement cust0001_vrf_imp term 1 then accept
set policy-options policy-statement cust0002_vrf_imp term 1 from community cust0001
set policy-options policy-statement cust0002_vrf_imp term 1 then accept
set policy-options community cust0001 members target:10001:1
set policy-options community cust0002 members target:10001:2
set routing-instances cust0001 vrf-import cust0001_vrf_imp
```

```
set routing-instances cust0002 vrf-import cust0002_vrf_imp
```

Step-by-Step Procedure

To configure profiles for the two customer groups:

1. Configure the IRB interfaces that enable communication among VXLANs v0001, v0002, v0005, and v0006.

```
[edit interfaces]
user@switch# set irb unit 1 family inet address 10.2.1.9/24 virtual-gateway-address
10.2.1.254
user@switch# set irb unit 2 family inet address 10.2.2.9/24 virtual-gateway-address
10.2.2.254
user@switch# set irb unit 5 family inet address 10.2.5.9/24 virtual-gateway-address
10.2.5.254
user@switch# set irb unit 6 family inet address 10.2.6.9/24 virtual-gateway-address
10.2.6.254
```

2. Configure a loopback interface for each customer group.

```
[edit interfaces]
user@switch# set lo0 unit 1 family inet address 127.9.0.1/32
user@switch# set lo0 unit 2 family inet address 127.9.0.2/32
```

3. Specify which VNIs are included in the EVPN-VXLAN domains.

```
[edit protocols]
user@switch# set evpn extended-vni-list [ 1 2 5 6 ]
```

4. Configure a route target for each VNI.

```
[edit protocols]
user@switch# set evpn vni-options vni 1 vrf-target export target:10003:1
user@switch# set evpn vni-options vni 2 vrf-target export target:10003:2
user@switch# set evpn vni-options vni 5 vrf-target export target:10003:5
user@switch# set evpn vni-options vni 6 vrf-target export target:10003:6
```

5. Configure policies that accept and import overlay routes.

```
[edit policy-options]
set policy-statement EVPN_VRF_IMPORT term cust0001 from community cust0001
set policy-statement EVPN_VRF_IMPORT term cust0001 then accept
set policy-statement EVPN_VRF_IMPORT term cust0002 from community cust0002
set policy-statement EVPN_VRF_IMPORT term cust0002 then accept
set policy-statement EVPN_VRF_IMPORT term vni0001 from community vni0001
set policy-statement EVPN_VRF_IMPORT term vni0001 then accept
set policy-statement EVPN_VRF_IMPORT term vni0002 from community vni0002
set policy-statement EVPN_VRF_IMPORT term vni0002 then accept
set policy-statement EVPN_VRF_IMPORT term vni0005 from community vni0005
set policy-statement EVPN_VRF_IMPORT term vni0005 then accept
set policy-statement EVPN_VRF_IMPORT term vni0006 from community vni0006
set policy-statement EVPN_VRF_IMPORT term vni0006 then accept
```

6. Set up communities for the customer groups and VXLANs.

```
user@switch# set community vni0001 members target:10003:1
user@switch# set community vni0002 members target:10003:2
user@switch# set community vni0005 members target:10003:5
user@switch# set community vni0006 members target:10003:6
```

7. Set up a routing instance for customer 1.



NOTE: When setting up a routing instance for EVPN-VXLAN, you must include a loopback interface. You must also specify an IP address for the interface using the `set interfaces lo0 unit logical-unit-number family inet address ip-address/prefix`. If you inadvertently omit the loopback interface and associated IP address, EVPN control packets cannot be processed.

```
[edit routing-instances]
user@switch# set cust0001 instance-type vrf
user@switch# set cust0001 interface irb.1
user@switch# set cust0001 interface irb.2
user@switch# set cust0001 interface lo0.1
user@switch# set cust0001 route-distinguisher 10.0.0.9:2001
user@switch# set cust0001 vrf-target target:10001:1
user@switch# set cust0001 routing-options auto-export
```

8. Set up a routing instance for customer 2.



NOTE: When setting up a routing instance for EVPN-VXLAN, you must include a loopback interface. You must also specify an IP address for the interface using the `set interfaces lo0 unit logical-unit-number family inet address ip-address/prefix`. If you inadvertently omit the loopback interface and associated IP address, EVPN control packets cannot be processed.

```
[edit routing-instances]
user@switch# set cust0002 instance-type vrf
user@switch# set cust0002 interface irb.5
user@switch# set cust0002 interface irb.6
user@switch# set cust0002 interface lo0.2
user@switch# set cust0002 route-distinguisher 10.0.0.9:2002
user@switch# set cust0002 vrf-target target:10001:2
user@switch# set cust0002 routing-options auto-export
```

9. Configure VXLANs v0001, v0002, v0005, and v0006, and associate the corresponding VNIs and IRB interfaces with each VXLAN.

```
[edit vlans]
user@switch# set v0001 vlan-id 1
user@switch# set v0001 l3-interface irb.1
user@switch# set v0001 vxlan vni 1
user@switch# set v0001 vxlan ingress-node-replication
user@switch# set v0002 vlan-id 2
user@switch# set v0002 l3-interface irb.2
user@switch# set v0002 vxlan vni 2
user@switch# set v0002 vxlan ingress-node-replication
user@switch# set v0005 vlan-id 5
user@switch# set v0005 l3-interface irb.5
user@switch# set v0005 vxlan vni 5
```

```

user@switch# set v0005 vxlan ingress-node-replication
user@switch# set v0006 vlan-id 6
user@switch# set v0006 l3-interface irb.6
user@switch# set v0006 vxlan vni 6
user@switch# set v0006 vxlan ingress-node-replication

```

10. Configure the importing of routes from one customer VRF instance into the other.

```

[edit policy-options]
user@switch# set policy-statement cust0001_vrf_imp term 1 from community
cust0002
user@switch# set policy-statement cust0001_vrf_imp term 1 then accept
user@switch# set policy-statement cust0002_vrf_imp term 1 from community
cust0001
user@switch# set policy-statement cust0002_vrf_imp term 1 then accept
user@switch# set community cust0001 members target:10001:1
user@switch# set community cust0002 members target:10001:2
[edit routing-instances]
user@switch# set cust0001 vrf-import cust0001_vrf_imp
user@switch# set cust0002 vrf-import cust0002_vrf_imp

```

Leaf 1: Underlay Network Configuration

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

```

set interfaces et-0/0/49 vlan-tagging
set interfaces et-0/0/49 unit 0 vlan-id 1
set interfaces et-0/0/49 unit 0 family inet address 10.1.12.3/24
set interfaces et-0/0/50 vlan-tagging
set interfaces et-0/0/50 unit 0 vlan-id 1
set interfaces et-0/0/50 unit 0 family inet address 10.1.24.3/24
set interfaces et-0/0/51 vlan-tagging
set interfaces et-0/0/51 unit 0 vlan-id 1
set interfaces et-0/0/51 unit 0 family inet address 10.1.30.3/24
set interfaces lo0 unit 0 family inet address 10.0.0.3/32 primary
set routing-options router-id 10.0.0.3
set routing-options autonomous-system 65000
set routing-options forwarding-table export load-balancing-policy
set protocols bgp multihop
set protocols bgp group underlay type external
set protocols bgp group underlay accept-remote-nexthop
set protocols bgp group underlay advertise-peer-as
set protocols bgp group underlay family inet unicast loops 5
set protocols bgp group underlay export send-direct
set protocols bgp group underlay peer-as 65001
set protocols bgp group underlay multipath
set protocols bgp group underlay neighbor 10.1.12.9
set protocols bgp group underlay neighbor 10.1.24.10
set protocols bgp group underlay neighbor 10.1.30.10
set policy-options policy-statement load-balancing-policy then load-balance per-packet
set policy-options policy-statement send-direct term 10 from protocol direct
set policy-options policy-statement send-direct term 10 then accept

```

Configuring the Underlay Network for Leaf 1

Step-by-Step Procedure

To configure the underlay network for Leaf 1:

1. Configure Layer 2 interfaces and associate each interface with VXLAN v0001.

```
[edit interfaces]
user@switch# set et-0/0/49 vlan-tagging
user@switch# set et-0/0/49 unit 0 vlan-id 1
user@switch# set et-0/0/49 unit 0 family inet address 10.1.12.3/24
user@switch# set et-0/0/50 vlan-tagging
user@switch# set et-0/0/50 unit 0 vlan-id 1
user@switch# set et-0/0/50 unit 0 family inet address 10.1.24.3/24
user@switch# set et-0/0/51 vlan-tagging
user@switch# set et-0/0/51 unit 0 vlan-id 1
user@switch# set et-0/0/51 unit 0 family inet address 10.1.30.3/24
```

2. Specify an IP address for the loopback interface. This IP address serves as the source IP address in the outer header of any VXLAN-encapsulated packets.

```
[edit interfaces]
user@switch# set lo0 unit 0 family inet address 10.0.0.3/32 primary
```

3. Set the routing options.

```
[edit routing-options]
user@switch# set router-id 10.0.0.3
user@switch# set autonomous-system 65000
user@switch# set forwarding-table export load-balancing-policy
```

4. Configure multihop EBGP.

```
[edit protocols]
user@switch# set bgp multihop
```

5. Configure a BGP group that includes peers that handle underlay functions.

```
[edit protocols]
user@switch# set bgp group underlay type external
user@switch# set bgp group underlay accept-remote-nexthop
user@switch# set bgp group underlay advertise-peer-as
user@switch# set bgp group underlay family inet unicast loops 5
user@switch# set bgp group underlay export send-direct
user@switch# set bgp group underlay peer-as 65001
user@switch# set bgp group underlay multipath
user@switch# set bgp group underlay neighbor 10.1.12.9
user@switch# set bgp group underlay neighbor 10.1.24.10
user@switch# set bgp group underlay neighbor 10.1.30.10
```



NOTE: You must enter the `set protocols bgp group underlay accept-remote-nexthop` command so that the BGP routes appear in the routing table as direct routes. VXLAN requires direct next hops for equal-cost multipath (ECMP) load balancing.

6. Configure a policy that spreads traffic across multiple paths between the Juniper Networks switches.

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

7. Configure a policy that enables BGP to advertise direct interfaces such as loopback interface lo0.

```
[edit policy-options]
user@switch# set policy-statement send-direct term 10 from protocol direct
user@switch# set policy-statement send-direct term 10 then accept
```

Leaf 1: Overlay Network Configuration

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

```
set protocols bgp group evpn type internal
set protocols bgp group evpn local-address 10.0.0.3
set protocols bgp group evpn family evpn signaling
set protocols bgp group evpn multipath
set protocols bgp group evpn neighbor 10.0.0.4
set protocols bgp group evpn neighbor 10.0.0.7
set protocols bgp group evpn neighbor 10.0.0.8
set protocols bgp group evpn neighbor 10.0.0.9
set protocols bgp group evpn neighbor 10.0.0.10
set protocols evpn encapsulation vxlan
set protocols evpn multicast-mode ingress-replication
set policy-options policy-statement EVPN_VRF_IMPORT term switch_options_comm
from community switch_options_comm
set policy-options policy-statement EVPN_VRF_IMPORT term switch_options_comm
then accept
set policy-options community switch_options_comm members target:65000:2
set switch-options vtep-source-interface lo0.0
set switch-options route-distinguisher 10.0.0.3:1
set switch-options vrf-import EVPN_VRF_IMPORT
set switch-options vrf-target target:65000:2
set switch-options vrf-target auto
```

Configuring the Overlay Network for Leaf 1

Step-by-Step Procedure To configure the overlay network for Leaf 1:

1. Configure an IBGP group for the EVPN-VXLAN overlay network.

```
[edit protocols]
user@switch# set bgp group evpn type internal
user@switch# set bgp group evpn local-address 10.0.0.3
user@switch# set bgp group evpn family evpn signaling
user@switch# set bgp group evpn multipath
user@switch# set bgp group evpn neighbor 10.0.0.4
user@switch# set bgp group evpn neighbor 10.0.0.7
```

```

user@switch# set bgp group evpn neighbor 10.0.0.8
user@switch# set bgp group evpn neighbor 10.0.0.9
user@switch# set bgp group evpn neighbor 10.0.0.10

```

2. Configure VXLAN encapsulation for the data packets exchanged between the EVPN neighbors.

```

[edit protocols]
user@switch# set evpn encapsulation vxlan

```

3. Specify a tunnel type used for passing multicast traffic between Juniper Networks switches in an EVPN-VXLAN environment.

```

[edit protocols]
user@switch# set evpn multicast-mode ingress-replication

```

4. Set up a policy to handle Layer 2 EVPN traffic.

```

user@switch# set policy-statement EVPN_VRF_IMPORT term switch_options_comm
from community switch_options_comm
user@switch# set policy-statement EVPN_VRF_IMPORT term switch_options_comm
then accept
user@switch# set community switch_options_comm members target:65000:2

```

5. Configure options for the default routing instance (virtual switch type).

```

[edit switch-options]
user@switch# set vtep-source-interface lo0.0
user@switch# set route-distinguisher 10.0.0.3:1
user@switch# set vrf-import EVPN_VRF_IMPORT
user@switch# set vrf-target target:65000:2
user@switch# set vrf-target auto

```

Leaf 1: Customer Profile Configuration

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

```

set interfaces xe-0/0/0 ether-options 802.3ad ae0
set interfaces ae0 esi 00:00:00:ab:cd:00:01:00:00:01
set interfaces ae0 esi all-active
set interfaces ae0 aggregated-ether-options lacp active
set interfaces ae0 aggregated-ether-options lacp system-id 00:11:00:00:00:01
set interfaces ae0 unit 0 family ethernet-switching interface-mode trunk
set interfaces ae0 unit 0 family ethernet-switching vlan members [ 1 2 5 6 ]
set protocols evpn vni-options vni 1 vrf-target export target:10003:1
set protocols evpn vni-options vni 2 vrf-target export target:10003:2
set protocols evpn vni-options vni 5 vrf-target export target:10003:5
set protocols evpn vni-options vni 6 vrf-target export target:10003:6
set protocols evpn extended-vni-list [ 1 2 5 6 ]
set policy-options policy-statement EVPN_VRF_IMPORT term cust0001 from community
cust0001
set policy-options policy-statement EVPN_VRF_IMPORT term cust0001 then accept
set policy-options policy-statement EVPN_VRF_IMPORT term cust0002 from community
cust0002

```

```

set policy-options policy-statement EVPN_VRF_IMPORT term cust0002 then accept
set policy-options policy-statement EVPN_VRF_IMPORT term vni0001 from community
vni0001
set policy-options policy-statement EVPN_VRF_IMPORT term vni0001 then accept
set policy-options policy-statement EVPN_VRF_IMPORT term vni0002 from community
vni0002
set policy-options policy-statement EVPN_VRF_IMPORT term vni0002 then accept
set policy-options policy-statement EVPN_VRF_IMPORT term vni0005 from community
vni0005
set policy-options policy-statement EVPN_VRF_IMPORT term vni0005 then accept
set policy-options policy-statement EVPN_VRF_IMPORT term vni0006 from community
vni0006
set policy-options policy-statement EVPN_VRF_IMPORT term vni0006 then accept
set policy-options community cust0001 members target:10001:1
set policy-options community cust0002 members target:10001:2
set policy-options community vni0001 members target:10003:1
set policy-options community vni0002 members target:10003:2
set policy-options community vni0005 members target:10003:5
set policy-options community vni0006 members target:10003:6
set vlans v0001 vlan-id 1
set vlans v0001 vxlan vni 1
set vlans v0001 vxlan ingress-node-replication
set vlans v0002 vlan-id 2
set vlans v0002 vxlan vni 2
set vlans v0002 vxlan ingress-node-replication
set vlans v0005 vlan-id 5
set vlans v0005 vxlan vni 5
set vlans v0005 vxlan ingress-node-replication
set vlans v0006 vlan-id 6
set vlans v0006 vxlan vni 6
set vlans v0006 vxlan ingress-node-replication

```

Configuring the Customer Profiles for Leaf 1

Step-by-Step Procedure

To configure the customer profiles for the two customer groups:

1. Configure an aggregated Ethernet interface for the connection with the physical server. This interface is associated with VXLANs v0001, v0002, v0005, and v0006.

```
[edit interfaces]
```

```
user@switch# set xe-0/0/0 ether-options 802.3ad ae0
```

```
user@switch# set ae0 esi 00:00:00:ab:cd:00:01:00:00:01
```

```
user@switch# set ae0 esi all-active
```

```
user@switch# set ae0 aggregated-ether-options lacp active
```

```
user@switch# set ae0 aggregated-ether-options lacp system-id 00:11:00:00:00:01
```

```
user@switch# set ae0 unit 0 family ethernet-switching interface-mode trunk
```

```
user@switch# set ae0 unit 0 family ethernet-switching vlan members [ 1 2 5 6 ]
```

2. Configure a route target for each VNI.

```
[edit protocols]
```

```
user@switch# set evpn vni-options vni 1 vrf-target export target:10003:1
```

```
user@switch# set evpn vni-options vni 2 vrf-target export target:10003:2
```

```
user@switch# set evpn vni-options vni 5 vrf-target export target:10003:5
```

```
user@switch# set evpn vni-options vni 6 vrf-target export target:10003:6
```

3. Specify which VNIs are included in the EVPN-VXLAN domains.

```
[edit protocols]
user@switch# set evpn extended-vni-list [ 1 2 5 6 ]
```

4. Configure policies that accept and import overlay routes.

```
[edit policy-options]
user@switch# set policy-statement EVPN_VRF_IMPORT term cust0001 from
community cust0001
user@switch# set policy-statement EVPN_VRF_IMPORT term cust0001 then accept
user@switch# set policy-statement EVPN_VRF_IMPORT term cust0002 from
community cust0002
user@switch# set policy-statement EVPN_VRF_IMPORT term cust0002 then accept
user@switch# set policy-statement EVPN_VRF_IMPORT term vni0001 from
community vni0001
user@switch# set policy-statement EVPN_VRF_IMPORT term vni0001 then accept
user@switch# set policy-statement EVPN_VRF_IMPORT term vni0002 from
community vni0002
user@switch# set policy-statement EVPN_VRF_IMPORT term vni0002 then accept
user@switch# set policy-statement EVPN_VRF_IMPORT term vni0005 from
community vni0005
user@switch# set policy-statement EVPN_VRF_IMPORT term vni0005 then accept
user@switch# set policy-statement EVPN_VRF_IMPORT term vni0006 from
community vni0006
user@switch# set policy-statement EVPN_VRF_IMPORT term vni0006 then accept
```

5. Set up communities for the customer groups and VXLANs.

```
[edit policy-options]
user@switch# set community cust0001 members target:10001:1
user@switch# set community cust0002 members target:10001:2
user@switch# set community vni0001 members target:10003:1
user@switch# set community vni0002 members target:10003:2
user@switch# set community vni0005 members target:10003:5
user@switch# set community vni0006 members target:10003:6
```

6. Configure VXLANs v0001, v0002, v0005, and v0006.

```
[edit vlans]
user@switch# set v0001 vlan-id 1
user@switch# set v0001 vxlan vni 1
user@switch# set v0001 vxlan ingress-node-replication
user@switch# set v0002 vlan-id 2
user@switch# set v0002 vxlan vni 2
user@switch# set v0002 vxlan ingress-node-replication
user@switch# set v0005 vlan-id 5
user@switch# set v0005 vxlan vni 5
user@switch# set v0005 vxlan ingress-node-replication
user@switch# set v0006 vlan-id 6
user@switch# set v0006 vxlan vni 6
user@switch# set v0006 vxlan ingress-node-replication
```

Leaf 2: Underlay Network Configuration

CLI Quick Configuration To quickly configure this example, copy the following commands, paste them into a text file, remove any line breaks, change any details necessary to match your configuration,

copy and paste the commands into the CLI at the `[edit]` hierarchy level, and then enter `commit` from configuration mode.

```

set interfaces et-0/0/49 unit 0 family inet address 10.1.13.4/24
set interfaces et-0/0/50 unit 0 family inet address 10.1.25.4/24
set interfaces et-0/0/51 unit 0 family inet address 10.1.31.4/24
set interfaces lo0 unit 0 family inet address 10.0.0.4/32 primary
set routing-options router-id 10.0.0.4
set routing-options autonomous-system 65000
set routing-options forwarding-table export load-balancing-policy
set protocols bgp multihop
set protocols bgp group underlay type external
set protocols bgp group underlay accept-remote-nexthop
set protocols bgp group underlay advertise-peer-as
set protocols bgp group underlay family inet unicast loops 5
set protocols bgp group underlay export send-direct
set protocols bgp group underlay peer-as 65001
set protocols bgp group underlay multipath
set protocols bgp group underlay neighbor 10.1.13.9
set protocols bgp group underlay neighbor 10.1.25.10
set protocols bgp group underlay neighbor 10.1.31.10
set policy-options policy-statement load-balancing-policy then load-balance per-packet
set policy-options policy-statement send-direct term 10 from protocol direct
set policy-options policy-statement send-direct term 10 then accept

```

Configuring the Underlay Network for Leaf 2

Step-by-Step Procedure

To configure the underlay network for Leaf 2:

1. Configure Layer 2 and Layer 3 interfaces.


```

[edit interfaces]
set et-0/0/49 unit 0 family inet address 10.1.13.4/24
set et-0/0/50 unit 0 family inet address 10.1.25.4/24
set et-0/0/51 unit 0 family inet address 10.1.31.4/24

```
2. Specify an IP address for the loopback interface. This IP address serves as the source IP address in the outer header of any VXLAN-encapsulated packets.


```

[edit interfaces]
user@switch# set lo0 unit 0 family inet address 10.0.0.4/32 primary

```
3. Set the routing options.


```

[edit routing-options]
user@switch# set router-id 10.0.0.4
user@switch# set autonomous-system 65000
user@switch# set forwarding-table export load-balancing-policy

```
4. Configure multihop EBGp.


```

[edit protocols]
user@switch# set bgp multihop

```
5. Configure a BGP group that includes peers that handle underlay functions.


```

[edit protocols]
user@switch# set bgp group underlay type external

```

```

user@switch# set bgp group underlay accept-remote-nexthop
user@switch# set bgp group underlay advertise-peer-as
user@switch# set bgp group underlay family inet unicast loops 5
user@switch# set bgp group underlay export send-direct
user@switch# set bgp group underlay peer-as 65001
user@switch# set bgp group underlay multipath
user@switch# set bgp group underlay neighbor 10.1.13.9
user@switch# set bgp group underlay neighbor 10.1.25.10
user@switch# set bgp group underlay neighbor 10.1.31.10

```



NOTE: You must enter the set protocols bgp group underlay accept-remote-nexthop command so that the BGP routes appear in the routing table as direct routes. VXLAN requires direct next hops for equal-cost multipath (ECMP) load balancing.

6. Configure a policy that spreads traffic across multiple paths between the Juniper Networks switches.

```

[edit policy-options]
user@switch# set policy-statement load-balancing-policy then load-balance
per-packet

```

7. Configure a policy that enables BGP to advertise direct interfaces such as loopback interface lo0.

```

[edit policy-options]
user@switch# set policy-statement send-direct term 10 from protocol direct
user@switch# set policy-statement send-direct term 10 then accept

```

Leaf 2: Overlay Network Configuration

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

```

set protocols bgp group evpn type internal
set protocols bgp group evpn local-address 10.0.0.4
set protocols bgp group evpn family evpn signaling
set protocols bgp group evpn multipath
set protocols bgp group evpn neighbor 10.0.0.3
set protocols bgp group evpn neighbor 10.0.0.7
set protocols bgp group evpn neighbor 10.0.0.8
set protocols bgp group evpn neighbor 10.0.0.9
set protocols bgp group evpn neighbor 10.0.0.10
set protocols evpn encapsulation vxlan
set protocols evpn multicast-mode ingress-replication
set policy-options policy-statement EVPN_VRF_IMPORT term switch_options_comm
from community switch_options_comm
set policy-options policy-statement EVPN_VRF_IMPORT term switch_options_comm
then accept
set policy-options community switch_options_comm members target:65000:2

```

```

set switch-options vtep-source-interface lo0.0
set switch-options route-distinguisher 10.0.0.4:1
set switch-options vrf-import EVPN_VRF_IMPORT
set switch-options vrf-target target:65000:2
set switch-options vrf-target auto

```

Configuring the Overlay Network for Leaf 2

Step-by-Step Procedure

To configure the overlay network for Leaf 2:

1. Configure an IBGP group for the EVPN-VXLAN overlay network.

```

[edit protocols]
user@switch# set bgp group evpn type internal
user@switch# set bgp group evpn local-address 10.0.0.4
user@switch# set bgp group evpn family evpn signaling
user@switch# set bgp group evpn multipath
user@switch# set bgp group evpn neighbor 10.0.0.3
user@switch# set bgp group evpn neighbor 10.0.0.7
user@switch# set bgp group evpn neighbor 10.0.0.8
user@switch# set bgp group evpn neighbor 10.0.0.9
user@switch# set bgp group evpn neighbor 10.0.0.10

```

2. Configure VXLAN encapsulation for the data packets exchanged between the EVPN neighbors.

```

[edit protocols]
user@switch# set evpn encapsulation vxlan

```

3. Specify a tunnel type used for passing multicast traffic between Juniper Networks switches in an EVPN-VXLAN environment.

```

[edit protocols]
user@switch# set evpn multicast-mode ingress-replication

```

4. Set up a policy to handle Layer 2 EVPN traffic.

```

[edit policy-options]
user@switch# set policy-statement EVPN_VRF_IMPORT term switch_options_comm
from community switch_options_comm
user@switch# set policy-statement EVPN_VRF_IMPORT term switch_options_comm
then accept
user@switch# set community switch_options_comm members target:65000:2

```

5. Configure options for the default routing instance (virtual switch type).

```

[edit switch-options]
user@switch# set vtep-source-interface lo0.0
user@switch# set route-distinguisher 10.0.0.4:1
user@switch# set vrf-import EVPN_VRF_IMPORT
user@switch# set vrf-target target:65000:2
user@switch# set vrf-target auto

```

Leaf 2: Customer Profile Configuration

CLI Quick Configuration To quickly configure this example, copy the following commands, paste them into a text file, remove any line breaks, change any details necessary to match your configuration,

copy and paste the commands into the CLI at the [edit] hierarchy level, and then enter **commit** from configuration mode.

```
set interfaces xe-0/0/0 ether-options 802.3ad ae0
set interfaces ae0 esi 00:00:00:ab:cd:00:01:00:00:01
set interfaces ae0 esi all-active
set interfaces ae0 aggregated-ether-options lacp active
set interfaces ae0 aggregated-ether-options lacp system-id 00:11:00:00:00:01
set interfaces ae0 unit 0 family ethernet-switching interface-mode trunk
set interfaces ae0 unit 0 family ethernet-switching vlan members [ 1 2 5 6 ]
set protocols evpn vni-options vni 1 vrf-target export target:10003:1
set protocols evpn vni-options vni 2 vrf-target export target:10003:2
set protocols evpn vni-options vni 5 vrf-target export target:10003:5
set protocols evpn vni-options vni 6 vrf-target export target:10003:6
set protocols evpn extended-vni-list [ 1 2 5 6 ]
set policy-options policy-statement EVPN_VRF_IMPORT term cust0001 from community
  cust0001
set policy-options policy-statement EVPN_VRF_IMPORT term cust0001 then accept
set policy-options policy-statement EVPN_VRF_IMPORT term cust0002 from community
  cust0002
set policy-options policy-statement EVPN_VRF_IMPORT term cust0002 then accept
set policy-options policy-statement EVPN_VRF_IMPORT term vni0001 from community
  vni0001
set policy-options policy-statement EVPN_VRF_IMPORT term vni0001 then accept
set policy-options policy-statement EVPN_VRF_IMPORT term vni0002 from community
  vni0002
set policy-options policy-statement EVPN_VRF_IMPORT term vni0002 then accept
set policy-options policy-statement EVPN_VRF_IMPORT term vni0005 from community
  vni0005
set policy-options policy-statement EVPN_VRF_IMPORT term vni0005 then accept
set policy-options policy-statement EVPN_VRF_IMPORT term vni0006 from community
  vni0006
set policy-options policy-statement EVPN_VRF_IMPORT term vni0006 then accept
set policy-options community cust0001 members target:10001:1
set policy-options community cust0002 members target:10001:2
set policy-options community vni0001 members target:10003:1
set policy-options community vni0002 members target:10003:2
set policy-options community vni0005 members target:10003:5
set policy-options community vni0006 members target:10003:6
set vlans v0001 vlan-id 1
set vlans v0001 vxlan vni 1
set vlans v0001 vxlan ingress-node-replication
set vlans v0002 vlan-id 2
set vlans v0002 vxlan vni 2
set vlans v0002 vxlan ingress-node-replication
set vlans v0005 vlan-id 5
set vlans v0005 vxlan vni 5
set vlans v0005 vxlan ingress-node-replication
set vlans v0006 vlan-id 6
set vlans v0006 vxlan vni 6
set vlans v0006 vxlan ingress-node-replication
```


Configuring the Customer Profiles for Leaf 2

Step-by-Step Procedure

To configure profiles for the two customer groups:

1. Configure an aggregated Ethernet interface for the connection with the physical server. This interface is associated with VXLANs v0001, v0002, v0005, and v0006.

```
[edit interfaces]
user@switch# set xe-0/0/0 ether-options 802.3ad ae0
user@switch# set ae0 esi 00:00:00:ab:cd:00:01:00:00:01
user@switch# set ae0 esi all-active
user@switch# set ae0 aggregated-ether-options lacp active
user@switch# set ae0 aggregated-ether-options lacp system-id 00:11:00:00:00:01
user@switch# set ae0 unit 0 family ethernet-switching interface-mode trunk
user@switch# set ae0 unit 0 family ethernet-switching vlan members [ 1 2 5 6 ]
```

2. Configure a route target for each VNI.

```
[edit protocols]
user@switch# set evpn vni-options vni 1 vrf-target export target:10003:1
user@switch# set evpn vni-options vni 2 vrf-target export target:10003:2
user@switch# set evpn vni-options vni 5 vrf-target export target:10003:5
user@switch# set evpn vni-options vni 6 vrf-target export target:10003:6
```

3. Specify which VNIs are included in the EVPN-VXLAN domains.

```
[edit protocols]
user@switch# set evpn extended-vni-list [ 1 2 5 6 ]
```

4. Configure policies that accept and import overlay routes.

```
[edit policy-options]
user@switch# set policy-statement EVPN_VRF_IMPORT term cust0001 from
community cust0001
user@switch# set policy-statement EVPN_VRF_IMPORT term cust0001 then accept
user@switch# set policy-statement EVPN_VRF_IMPORT term cust0002 from
community cust0002
user@switch# set policy-statement EVPN_VRF_IMPORT term cust0002 then accept
user@switch# set policy-statement EVPN_VRF_IMPORT term vni0001 from
community vni0001
user@switch# set policy-statement EVPN_VRF_IMPORT term vni0001 then accept
user@switch# set policy-statement EVPN_VRF_IMPORT term vni0002 from
community vni0002
user@switch# set policy-statement EVPN_VRF_IMPORT term vni0002 then accept
user@switch# set policy-statement EVPN_VRF_IMPORT term vni0005 from
community vni0005
user@switch# set policy-statement EVPN_VRF_IMPORT term vni0005 then accept
user@switch# set policy-statement EVPN_VRF_IMPORT term vni0006 from
community vni0006
user@switch# set policy-statement EVPN_VRF_IMPORT term vni0006 then accept
```

5. Set up the communities.

```
[edit policy-options]
user@switch# set community cust0001 members target:10001:1
user@switch# set community cust0002 members target:10001:2
user@switch# set community vni0001 members target:10003:1
user@switch# set community vni0002 members target:10003:2
```

```

user@switch# set community vni0005 members target:10003:5
user@switch# set community vni0006 members target:10003:6

```

6. Configure VXLANs v0001, v0002, v0005, and v0006.

```

[edit vlans]
user@switch# set v0001 vlan-id 1
user@switch# set v0001 vxlan vni 1
user@switch# set v0001 vxlan ingress-node-replication
user@switch# set v0002 vlan-id 2
user@switch# set v0002 vxlan vni 2
user@switch# set v0002 vxlan ingress-node-replication
user@switch# set v0005 vlan-id 5
user@switch# set v0005 vxlan vni 5
user@switch# set v0005 vxlan ingress-node-replication
user@switch# set v0006 vlan-id 6
user@switch# set v0006 vxlan vni 6
user@switch# set v0006 vxlan ingress-node-replication

```

Leaf 3: Underlay Network Configuration

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

```

set interfaces et-0/0/48 unit 0 family inet address 10.1.4.7/24
set interfaces et-0/0/50 unit 0 family inet address 10.1.28.7/24
set interfaces et-0/0/51 unit 0 family inet address 10.1.34.7/24
set interfaces et-0/0/52 unit 0 family inet address 10.1.40.7/24
set interfaces lo0 unit 0 family inet address 10.0.0.7/32 primary
set routing-options router-id 10.0.0.7
set routing-options autonomous-system 65000
set routing-options forwarding-table export load-balancing-policy
set protocols bgp multihop
set protocols bgp group underlay type external
set protocols bgp group underlay accept-remote-nexthop
set protocols bgp group underlay advertise-peer-as
set protocols bgp group underlay family inet unicast loops 5
set protocols bgp group underlay export send-direct
set protocols bgp group underlay peer-as 65001
set protocols bgp group underlay multipath
set protocols bgp group underlay neighbor 10.1.4.9
set protocols bgp group underlay neighbor 10.1.28.10
set protocols bgp group underlay neighbor 10.1.34.10
set protocols bgp group underlay neighbor 10.1.40.9
set policy-options policy-statement load-balancing-policy then load-balance per-packet
set policy-options policy-statement send-direct term 10 from protocol direct
set policy-options policy-statement send-direct term 10 then accept

```

Configuring the Underlay Network for Leaf 3

Step-by-Step Procedure

To configure the underlay network for Leaf 3:

1. Configure Layer 3 interfaces.

```
[edit interfaces]
user@switch# set et-0/0/48 unit 0 family inet address 10.1.4.7/24
user@switch# set et-0/0/50 unit 0 family inet address 10.1.28.7/24
user@switch# set et-0/0/51 unit 0 family inet address 10.1.34.7/24
user@switch# set et-0/0/52 unit 0 family inet address 10.1.40.7/24
```

2. Specify an IP address for the loopback interface. This IP address serves as the source IP address in the outer header of any VXLAN-encapsulated packets.

```
[edit interfaces]
user@switch# set lo0 unit 0 family inet address 10.0.0.7/32 primary
```

3. Set the routing options.

```
[edit routing-options]
user@switch# set router-id 10.0.0.7
user@switch# set autonomous-system 65000
user@switch# set forwarding-table export load-balancing-policy
```

4. Configure multihop EBGP.

```
[edit protocols]
user@switch# set bgp multihop
```

5. Configure a BGP group that includes peers that handle underlay functions.

```
[edit protocols]
user@switch# set bgp group underlay type external
user@switch# set bgp group underlay accept-remote-nexthop
user@switch# set bgp group underlay advertise-peer-as
user@switch# set bgp group underlay family inet unicast loops 5
user@switch# set bgp group underlay export send-direct
user@switch# set bgp group underlay peer-as 65001
user@switch# set bgp group underlay multipath
user@switch# set bgp group underlay neighbor 10.1.4.9
user@switch# set bgp group underlay neighbor 10.1.28.10
user@switch# set bgp group underlay neighbor 10.1.34.10
user@switch# set bgp group underlay neighbor 10.1.40.9
```



NOTE: You must enter the `set protocols bgp group underlay accept-remote-nexthop` command so that the BGP routes appear in the routing table as direct routes. VXLAN requires direct next hops for equal-cost multipath (ECMP) load balancing.

6. Configure a policy that spreads traffic across multiple paths between the Juniper Networks switches.

```
[edit policy-options]
```

```
user@switch# set policy-statement load-balancing-policy then load-balance
per-packet
```

7. Configure a policy that enables BGP to advertise direct interfaces such as loopback interface lo0.

```
[edit policy-options]
user@switch# set policy-statement send-direct term 10 from protocol direct
user@switch# set policy-statement send-direct term 10 then accept
```

Leaf 3: Overlay Network Configuration

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

```
set protocols bgp group evpn type internal
set protocols bgp group evpn local-address 10.0.0.7
set protocols bgp group evpn family evpn signaling
set protocols bgp group evpn multipath
set protocols bgp group evpn neighbor 10.0.0.3
set protocols bgp group evpn neighbor 10.0.0.4
set protocols bgp group evpn neighbor 10.0.0.8
set protocols bgp group evpn neighbor 10.0.0.9
set protocols bgp group evpn neighbor 10.0.0.10
set protocols evpn encapsulation vxlan
set protocols evpn multicast-mode ingress-replication
set policy-options policy-statement EVPN_VRF_IMPORT term switch_options_comm
  from community switch_options_comm
set policy-options policy-statement EVPN_VRF_IMPORT term switch_options_comm
  then accept
set policy-options community switch_options_comm members target:65000:2
set switch-options vtep-source-interface lo0.0
set switch-options route-distinguisher 10.0.0.7:1
set switch-options vrf-import EVPN_VRF_IMPORT
set switch-options vrf-target target:65000:2
set switch-options vrf-target auto
```

Configuring the Overlay Network for Leaf 3

Step-by-Step Procedure To configure the overlay network for Leaf 3:

1. Configure an IBGP group for the EVPN-VXLAN overlay network.

```
[edit protocols]
user@switch# set bgp group evpn type internal
user@switch# set bgp group evpn local-address 10.0.0.7
user@switch# set bgp group evpn family evpn signaling
user@switch# set bgp group evpn multipath
user@switch# set bgp group evpn neighbor 10.0.0.3
user@switch# set bgp group evpn neighbor 10.0.0.4
user@switch# set bgp group evpn neighbor 10.0.0.8
user@switch# set bgp group evpn neighbor 10.0.0.9
user@switch# set bgp group evpn neighbor 10.0.0.10
```

2. Configure VXLAN encapsulation for the data packets exchanged between the EVPN neighbors.

```
[edit protocols]
user@switch# set evpn encapsulation vxlan
```

3. Specify a tunnel type used for passing multicast traffic between Juniper Networks switches in an EVPN-VXLAN environment.

```
[edit protocols]
user@switch# set evpn multicast-mode ingress-replication
```

4. Set up a policy to handle Layer 2 EVPN traffic.

```
[edit policy-options]
user@switch# set policy-statement EVPN_VRF_IMPORT term switch_options_comm
  from community switch_options_comm
user@switch# set policy-statement EVPN_VRF_IMPORT term switch_options_comm
  then accept
user@switch# set community switch_options_comm members target:65000:2
```

5. Configure options for the default routing instance (virtual switch type).

```
[edit switch-options]
user@switch# set vtep-source-interface lo0.0
user@switch# set route-distinguisher 10.0.0.7:1
user@switch# set vrf-import EVPN_VRF_IMPORT
user@switch# set vrf-target target:65000:2
user@switch# set vrf-target auto
```

Leaf 3: Customer Profile Configuration

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

```
set interfaces xe-0/0/0 ether-options 802.3ad ae0
set interfaces ae0 esi 00:00:00:ab:cd:00:02:00:00:01
set interfaces ae0 esi all-active
set interfaces ae0 aggregated-ether-options lacp active
set interfaces ae0 aggregated-ether-options lacp system-id 00:22:00:00:00:01
set interfaces ae0 unit 0 family ethernet-switching interface-mode trunk
set interfaces ae0 unit 0 family ethernet-switching vlan members [ 1 2 5 6 ]
set protocols evpn vni-options vni 1 vrf-target export target:10003:1
set protocols evpn vni-options vni 2 vrf-target export target:10003:2
set protocols evpn vni-options vni 5 vrf-target export target:10003:5
set protocols evpn vni-options vni 6 vrf-target export target:10003:6
set protocols evpn extended-vni-list [ 1 2 5 6 ]
set policy-options policy-statement EVPN_VRF_IMPORT term cust0001 from community
  cust0001
set policy-options policy-statement EVPN_VRF_IMPORT term cust0001 then accept
set policy-options policy-statement EVPN_VRF_IMPORT term cust0002 from community
  cust0002
set policy-options policy-statement EVPN_VRF_IMPORT term cust0002 then accept
set policy-options policy-statement EVPN_VRF_IMPORT term vni0001 from community
  vni0001
```

```

set policy-options policy-statement EVPN_VRF_IMPORT term vni0001 then accept
set policy-options policy-statement EVPN_VRF_IMPORT term vni0002 from community
vni0002
set policy-options policy-statement EVPN_VRF_IMPORT term vni0002 then accept
set policy-options policy-statement EVPN_VRF_IMPORT term vni0005 from community
vni0005
set policy-options policy-statement EVPN_VRF_IMPORT term vni0005 then accept
set policy-options policy-statement EVPN_VRF_IMPORT term vni0006 from community
vni0006
set policy-options policy-statement EVPN_VRF_IMPORT term vni0006 then accept
set policy-options community cust0001 members target:10001:1
set policy-options community cust0002 members target:10001:2
set policy-options community vni0001 members target:10003:1
set policy-options community vni0002 members target:10003:2
set policy-options community vni0005 members target:10003:5
set policy-options community vni0006 members target:10003:6
set vlans v0001 vlan-id 1
set vlans v0001 vxlan vni 1
set vlans v0001 vxlan ingress-node-replication
set vlans v0002 vlan-id 2
set vlans v0002 vxlan vni 2
set vlans v0002 vxlan ingress-node-replication
set vlans v0005 vlan-id 5
set vlans v0005 vxlan vni 5
set vlans v0005 vxlan ingress-node-replication
set vlans v0006 vlan-id 6
set vlans v0006 vxlan vni 6
set vlans v0006 vxlan ingress-node-replication

```

Step-by-Step Procedure

To configure profiles for the two customer groups:

1. Configure an aggregated Ethernet interface for the connection with the physical server. This interface is associated with VXLANs v0001, v0002, v0005, and v0006.

```
[edit interfaces]
```

```
user@switch# set xe-0/0/0 ether-options 802.3ad ae0
```

```
user@switch# set ae0 esi 00:00:00:ab:cd:00:02:00:00:01
```

```
user@switch# set ae0 esi all-active
```

```
user@switch# set ae0 aggregated-ether-options lacp active
```

```
user@switch# set ae0 aggregated-ether-options lacp system-id 00:22:00:00:00:01
```

```
user@switch# set ae0 unit 0 family ethernet-switching interface-mode trunk
```

```
user@switch# set ae0 unit 0 family ethernet-switching vlan members [ 1 2 5 6 ]
```

2. Configure a route target for each VNI.

```
[edit protocols]
```

```
user@switch# set evpn vni-options vni 1 vrf-target export target:10003:1
```

```
user@switch# set evpn vni-options vni 2 vrf-target export target:10003:2
```

```
user@switch# set evpn vni-options vni 5 vrf-target export target:10003:5
```

```
user@switch# set evpn vni-options vni 6 vrf-target export target:10003:6
```

3. Specify which VNIs are included in the EVPN-VXLAN domains.

```
[edit protocols]
```

```
user@switch# set evpn extended-vni-list [ 1 2 5 6 ]
```

4. Configure policies that accept and import overlay routes.

```
[edit policy-options]
user@switch# set policy-statement EVPN_VRF_IMPORT term cust0001 from
community cust0001
user@switch# set policy-statement EVPN_VRF_IMPORT term cust0001 then accept
user@switch# set policy-statement EVPN_VRF_IMPORT term cust0002 from
community cust0002
user@switch# set policy-statement EVPN_VRF_IMPORT term cust0002 then accept
user@switch# set policy-statement EVPN_VRF_IMPORT term vni0001 from
community vni0001
user@switch# set policy-statement EVPN_VRF_IMPORT term vni0001 then accept
user@switch# set policy-statement EVPN_VRF_IMPORT term vni0002 from
community vni0002
user@switch# set policy-statement EVPN_VRF_IMPORT term vni0002 then accept
user@switch# set policy-statement EVPN_VRF_IMPORT term vni0005 from
community vni0005
user@switch# set policy-statement EVPN_VRF_IMPORT term vni0005 then accept
user@switch# set policy-statement EVPN_VRF_IMPORT term vni0006 from
community vni0006
user@switch# set policy-statement EVPN_VRF_IMPORT term vni0006 then accept
```

5. Set up communities for the customer groups and VXLANs.

```
[edit policy-options]
user@switch# set community cust0001 members target:10001:1
user@switch# set community cust0002 members target:10001:2
user@switch# set community vni0001 members target:10003:1
user@switch# set community vni0002 members target:10003:2
user@switch# set community vni0005 members target:10003:5
user@switch# set community vni0006 members target:10003:6
```

6. Configure VXLANs v0001, v0002, v0005, and v0006.

```
[edit vlans]
user@switch# set v0001 vlan-id 1
user@switch# set v0001 vxlan vni 1
user@switch# set v0001 vxlan ingress-node-replication
user@switch# set v0002 vlan-id 2
user@switch# set v0002 vxlan vni 2
user@switch# set v0002 vxlan ingress-node-replication
user@switch# set v0005 vlan-id 5
user@switch# set v0005 vxlan vni 5
user@switch# set v0005 vxlan ingress-node-replication
user@switch# set v0006 vlan-id 6
user@switch# set v0006 vxlan vni 6
user@switch# set v0006 vxlan ingress-node-replication
```

Leaf 4: Underlay Network Configuration

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

```
set interfaces et-0/0/48 unit 0 family inet address 10.1.5.8/24
```

```
set interfaces et-0/0/50 unit 0 family inet address 10.1.29.8/24
set interfaces et-0/0/51 unit 0 family inet address 10.1.35.8/24
set interfaces et-0/0/52 unit 0 family inet address 10.1.41.8/24
set interfaces lo0 unit 0 family inet address 10.0.0.8/32 primary
set routing-options router-id 10.0.0.8
set routing-options autonomous-system 65000
set routing-options forwarding-table export load-balancing-policy
set protocols bgp multihop
set protocols bgp group underlay type external
set protocols bgp group underlay accept-remote-nexthop
set protocols bgp group underlay advertise-peer-as
set protocols bgp group underlay family inet unicast loops 5
set protocols bgp group underlay export send-direct
set protocols bgp group underlay peer-as 65001
set protocols bgp group underlay multipath
set protocols bgp group underlay neighbor 10.1.5.9
set protocols bgp group underlay neighbor 10.1.29.10
set protocols bgp group underlay neighbor 10.1.35.10
set protocols bgp group underlay neighbor 10.1.41.9
set policy-options policy-statement load-balancing-policy then load-balance per-packet
set policy-options policy-statement send-direct term 10 from protocol direct
set policy-options policy-statement send-direct term 10 then accept
```

Configuring the Underlay Network for Leaf 4

Step-by-Step Procedure

To configure the underlay network for Leaf 4:

1. Configure Layer 3 interfaces.

```
[edit interfaces]
set et-0/0/48 unit 0 family inet address 10.1.5.8/24
set et-0/0/50 unit 0 family inet address 10.1.29.8/24
set et-0/0/51 unit 0 family inet address 10.1.35.8/24
set et-0/0/52 unit 0 family inet address 10.1.41.8/24
```

2. Specify an IP address for the loopback interface. This IP address serves as the source IP address in the outer header of any VXLAN-encapsulated packets.

```
[edit interfaces]
user@switch# set lo0 unit 0 family inet address 10.0.0.8/32 primary
```

3. Set the routing options.

```
[edit routing-options]
user@switch# set router-id 10.0.0.8
user@switch# set autonomous-system 65000
user@switch# set forwarding-table export load-balancing-policy
```

4. Configure multihop EBGp.

```
[edit protocols]
user@switch# set bgp multihop
```

5. Configure a BGP group that includes peers that handle underlay functions.

```
[edit protocols]
user@switch# set bgp group underlay type external
user@switch# set bgp group underlay accept-remote-nexthop
```



```

user@switch# set bgp group underlay advertise-peer-as
user@switch# set bgp group underlay family inet unicast loops 5
user@switch# set bgp group underlay export send-direct
user@switch# set bgp group underlay peer-as 65001
user@switch# set bgp group underlay multipath
user@switch# set bgp group underlay neighbor 10.1.5.9
user@switch# set bgp group underlay neighbor 10.1.29.10
user@switch# set bgp group underlay neighbor 10.1.35.10
user@switch# set bgp group underlay neighbor 10.1.41.9

```



NOTE: You must enter the set protocols bgp group underlay accept-remote-nexthop command so that the BGP routes appear in the routing table as direct routes. VXLAN requires direct next hops for equal-cost multipath (ECMP) load balancing.

6. Configure a policy that spreads traffic across multiple paths between the Juniper Networks switches.

```

[edit policy-options]
user@switch# set policy-statement load-balancing-policy then load-balance
per-packet

```

7. Configure a policy that enables BGP to advertise direct interfaces such as loopback interface lo0.

```

[edit policy-options]
user@switch# set policy-statement send-direct term 10 from protocol direct
user@switch# set policy-statement send-direct term 10 then accept

```

Leaf 4: Overlay Network Configuration

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

```

set protocols bgp group evpn type internal
set protocols bgp group evpn local-address 10.0.0.8
set protocols bgp group evpn family evpn signaling
set protocols bgp group evpn multipath
set protocols bgp group evpn neighbor 10.0.0.3
set protocols bgp group evpn neighbor 10.0.0.4
set protocols bgp group evpn neighbor 10.0.0.7
set protocols bgp group evpn neighbor 10.0.0.9
set protocols bgp group evpn neighbor 10.0.0.10
set protocols evpn encapsulation vxlan
set protocols evpn multicast-mode ingress-replication
set policy-options policy-statement EVPN_VRF_IMPORT term switch_options_comm
from community switch_options_comm
set policy-options policy-statement EVPN_VRF_IMPORT term switch_options_comm
then accept
set policy-options community switch_options_comm members target:65000:2

```

```
set switch-options vtep-source-interface lo0.0
set switch-options route-distinguisher 10.0.0.8:1
set switch-options vrf-import EVPN_VRF_IMPORT
set switch-options vrf-target target:65000:2
set switch-options vrf-target auto
```

Configuring the Overlay Network for Leaf 4

Step-by-Step Procedure

To configure the overlay network for Leaf 4:

1. Configure an IBGP group for the EVPN-VXLAN overlay.

```
[edit protocols]
user@switch# set bgp group evpn type internal
user@switch# set bgp group evpn local-address 10.0.0.8
user@switch# set bgp group evpn family evpn signaling
user@switch# set bgp group evpn multipath
user@switch# set bgp group evpn neighbor 10.0.0.3
user@switch# set bgp group evpn neighbor 10.0.0.4
user@switch# set bgp group evpn neighbor 10.0.0.7
user@switch# set bgp group evpn neighbor 10.0.0.9
user@switch# set bgp group evpn neighbor 10.0.0.10
```

2. Configure VXLAN encapsulation for the data packets exchanged between the EVPN neighbors.

```
[edit protocols]
user@switch# set evpn encapsulation vxlan
```

3. Specify a tunnel type used for passing multicast traffic between Juniper Networks switches in an EVPN-VXLAN environment.

```
[edit protocols]
user@switch# set evpn multicast-mode ingress-replication
```

4. Set up a policy to handle Layer 2 EVPN traffic.

```
[edit policy-options]
user@switch# set policy-statement EVPN_VRF_IMPORT term switch_options_comm
  from community switch_options_comm
user@switch# set policy-statement EVPN_VRF_IMPORT term switch_options_comm
  then accept
user@switch# set community switch_options_comm members target:65000:2
```

5. Configure options for the default routing instance (virtual switch type).

```
[edit switch-options]
user@switch# set vtep-source-interface lo0.0
user@switch# set route-distinguisher 10.0.0.8:1
user@switch# set vrf-import EVPN_VRF_IMPORT
user@switch# set vrf-target target:65000:2
user@switch# set vrf-target auto
```

Leaf 4: Customer Profile Configuration

CLI Quick Configuration To quickly configure this example, copy the following commands, paste them into a text file, remove any line breaks, change any details necessary to match your configuration,

copy and paste the commands into the CLI at the [edit] hierarchy level, and then enter **commit** from configuration mode.

```

set interfaces xe-0/0/0 ether-options 802.3ad ae0
set interfaces ae0 esi 00:00:00:ab:cd:00:02:00:00:01
set interfaces ae0 esi all-active
set interfaces ae0 aggregated-ether-options lacp active
set interfaces ae0 aggregated-ether-options lacp system-id 00:22:00:00:00:01
set interfaces ae0 unit 0 family ethernet-switching interface-mode trunk
set interfaces ae0 unit 0 family ethernet-switching vlan members [ 1 2 5 6 ]
set protocols evpn vni-options vni 1 vrf-target export target:10003:1
set protocols evpn vni-options vni 2 vrf-target export target:10003:2
set protocols evpn vni-options vni 5 vrf-target export target:10003:5
set protocols evpn vni-options vni 6 vrf-target export target:10003:6
set protocols evpn extended-vni-list [ 1 2 5 6 ]
set policy-options policy-statement EVPN_VRF_IMPORT term cust0001 from community
  cust0001
set policy-options policy-statement EVPN_VRF_IMPORT term cust0001 then accept
set policy-options policy-statement EVPN_VRF_IMPORT term cust0002 from community
  cust0002
set policy-options policy-statement EVPN_VRF_IMPORT term cust0002 then accept
set policy-options policy-statement EVPN_VRF_IMPORT term vni0001 from community
  vni0001
set policy-options policy-statement EVPN_VRF_IMPORT term vni0001 then accept
set policy-options policy-statement EVPN_VRF_IMPORT term vni0002 from community
  vni0002
set policy-options policy-statement EVPN_VRF_IMPORT term vni0002 then accept
set policy-options policy-statement EVPN_VRF_IMPORT term vni0005 from community
  vni0005
set policy-options policy-statement EVPN_VRF_IMPORT term vni0005 then accept
set policy-options policy-statement EVPN_VRF_IMPORT term vni0006 from community
  vni0006
set policy-options policy-statement EVPN_VRF_IMPORT term vni0006 then accept
set policy-options community cust0001 members target:10001:1
set policy-options community cust0002 members target:10001:2
set policy-options community vni0001 members target:10003:1
set policy-options community vni0002 members target:10003:2
set policy-options community vni0005 members target:10003:5
set policy-options community vni0006 members target:10003:6
set vlans v0001 vlan-id 1
set vlans v0001 vxlan vni 1
set vlans v0001 vxlan ingress-node-replication
set vlans v0002 vlan-id 2
set vlans v0002 vxlan vni 2
set vlans v0002 vxlan ingress-node-replication
set vlans v0005 vlan-id 5
set vlans v0005 vxlan vni 5
set vlans v0005 vxlan ingress-node-replication
set vlans v0006 vlan-id 6
set vlans v0006 vxlan vni 6
set vlans v0006 vxlan ingress-node-replication

```

Step-by-Step Procedure

To configure profiles for the two customer groups:

1. Configure an aggregated Ethernet interface for the connection with the physical server. This interface is associated with VXLANs v0001, v0002, v0005, and v0006.

```
[edit interfaces]
user@switch# set xe-0/0/0 ether-options 802.3ad ae0
user@switch# set ae0 esi 00:00:00:ab:cd:00:02:00:00:01
user@switch# set ae0 esi all-active
user@switch# set ae0 aggregated-ether-options lacp active
user@switch# set ae0 aggregated-ether-options lacp system-id 00:22:00:00:00:01
user@switch# set ae0 unit 0 family ethernet-switching interface-mode trunk
user@switch# set ae0 unit 0 family ethernet-switching vlan members [ 1 2 5 6 ]
```

2. Configure a route target for each VNI.

```
[edit protocols]
user@switch# set evpn vni-options vni 1 vrf-target export target:10003:1
user@switch# set evpn vni-options vni 2 vrf-target export target:10003:2
user@switch# set evpn vni-options vni 5 vrf-target export target:10003:5
user@switch# set evpn vni-options vni 6 vrf-target export target:10003:6
```

3. Specify which VNIs are included in the EVPN-VXLAN domains.

```
[edit protocols]
user@switch# set evpn extended-vni-list [ 1 2 5 6 ]
```

4. Configure policies that accept and import overlay routes.

```
[edit policy-options]
user@switch# set policy-statement EVPN_VRF_IMPORT term cust0001 from
community cust0001
user@switch# set policy-statement EVPN_VRF_IMPORT term cust0001 then accept
user@switch# set policy-statement EVPN_VRF_IMPORT term cust0002 from
community cust0002
user@switch# set policy-statement EVPN_VRF_IMPORT term cust0002 then accept
user@switch# set policy-statement EVPN_VRF_IMPORT term vni0001 from
community vni0001
user@switch# set policy-statement EVPN_VRF_IMPORT term vni0001 then accept
user@switch# set policy-statement EVPN_VRF_IMPORT term vni0002 from
community vni0002
user@switch# set policy-statement EVPN_VRF_IMPORT term vni0002 then accept
user@switch# set policy-statement EVPN_VRF_IMPORT term vni0005 from
community vni0005
user@switch# set policy-statement EVPN_VRF_IMPORT term vni0005 then accept
user@switch# set policy-statement EVPN_VRF_IMPORT term vni0006 from
community vni0006
user@switch# set policy-statement EVPN_VRF_IMPORT term vni0006 then accept
```

5. Set up communities for the customer groups and VXLANs.

```
[edit policy-options]
user@switch# set community cust0001 members target:10001:1
user@switch# set community cust0002 members target:10001:2
user@switch# set community vni0001 members target:10003:1
user@switch# set community vni0002 members target:10003:2
user@switch# set community vni0005 members target:10003:5
user@switch# set community vni0006 members target:10003:6
```

6. Configure VXLANs v0001, v0002, v0005, and v0006.

```
[edit vlans]
user@switch# set v0001 vlan-id 1
user@switch# set v0001 vxlan vni 1
user@switch# set v0001 vxlan ingress-node-replication
user@switch# set v0002 vlan-id 2
user@switch# set v0002 vxlan vni 2
user@switch# set v0002 vxlan ingress-node-replication
user@switch# set v0005 vlan-id 5
user@switch# set v0005 vxlan vni 5
user@switch# set v0005 vxlan ingress-node-replication
user@switch# set v0006 vlan-id 6
user@switch# set v0006 vxlan vni 6
user@switch# set v0006 vxlan ingress-node-replication
```

Verification

Confirm that the IRB interfaces are working properly:

- [Verifying the Configuration of the IRB Interfaces on page 93](#)
- [Verifying the Configuration of the Routing Instances on page 95](#)
- [Verifying that Dynamic MAC Addresses Are Installed on page 96](#)

Verifying the Configuration of the IRB Interfaces

Purpose Verify the configuration of the IRB interfaces on Spine 1 and Spine 2.

Action From operational mode, enter the **show interfaces irb** command.

```
user@switch> show interfaces irb
Physical interface: irb, Enabled, Physical link is Up
  Interface index: 641, SNMP ifIndex: 503
  Type: Ethernet, Link-level type: Ethernet, MTU: 1514
  Device flags : Present Running
  Interface flags: SNMP-Traps
  Link type : Full-Duplex
  Link flags : None
  Current address: 0c:86:10:d6:9d:fe, Hardware address: 0c:86:10:d6:9d:fe
  Last flapped : Never
    Input packets : 0
    Output packets: 0

Logical interface irb.1 (Index 923) (SNMP ifIndex 520)
  Flags: Up SNMP-Traps 0x4000 Encapsulation: ENET2
  Bandwidth: 1000mbps
  Routing Instance: default-switch Bridging Domain: v0001
  Input packets : 0
  Output packets: 9
  Protocol inet, MTU: 1550
    Flags: Sendbcst-pkt-to-re, Is-Primary
    Addresses, Flags: Is-Preferred Is-Primary
      Destination: 10.2.1/24, Local: 10.2.1.9, Broadcast: 10.2.1.255

Logical interface irb.2 (Index 924) (SNMP ifIndex 525)
  Flags: Up SNMP-Traps 0x4000 Encapsulation: ENET2
  Bandwidth: 1000mbps
  Routing Instance: default-switch Bridging Domain: v0002
  Input packets : 0
  Output packets: 9
  Protocol inet, MTU: 1550
    Flags: Sendbcst-pkt-to-re
    Addresses, Flags: Is-Preferred Is-Primary
      Destination: 10.2.2/24, Local: 10.2.2.9, Broadcast: 10.2.2.255

Logical interface irb.5 (Index 927) (SNMP ifIndex 535)
  Flags: Up SNMP-Traps 0x4000 Encapsulation: ENET2
  Bandwidth: 1000mbps
  Routing Instance: default-switch Bridging Domain: v0005
  Input packets : 0
  Output packets: 1
  Protocol inet, MTU: 1550
    Flags: Sendbcst-pkt-to-re, Is-Primary
    Addresses, Flags: Is-Preferred Is-Primary
      Destination: 10.2.5/24, Local: 10.2.5.9, Broadcast: 10.2.5.255

Logical interface irb.6 (Index 928) (SNMP ifIndex 536)
  Flags: Up SNMP-Traps 0x4000 Encapsulation: ENET2
  Bandwidth: 1000mbps
  Routing Instance: default-switch Bridging Domain: v0006
  Input packets : 0
  Output packets: 1
  Protocol inet, MTU: 1550
    Flags: Sendbcst-pkt-to-re
    Addresses, Flags: Is-Preferred Is-Primary
      Destination: 10.2.6/24, Local: 10.2.6.9, Broadcast: 10.2.6.255
```

Meaning The sample output from Spine 2 verifies the following:

- IRB interfaces irb.1, irb.2, irb.5, and irb.6 are configured.
- The physical interface upon which the IRB interfaces are configured is up and running.
- Each IRB interface is properly mapped to its respective VXLAN.
- The configuration of each IRB interface properly reflects the IP address and destination (virtual gateway address) assigned to it.

Verifying the Configuration of the Routing Instances

Purpose Verify that the routing instances for customer groups 1 and 2 are properly configured on Spine 1 and Spine 2.

Action From operational mode, enter the **show route instance *routing-instance-name* extensive** command for customer groups 1 and 2.

```
user@switch> show route instance cust0001 extensive
cust0001:
  Router ID: 127.9.0.1
  Type: vrf                               State: Active
  Interfaces:
    lo0.1
    irb.2
    irb.1
  Route-distinguisher: 10.0.0.9:2001
  Vrf-import: [ cust0001_vrf_imp ]
  Vrf-export: [ __vrf-export-cust0001-internal__ ]
  Vrf-export-target: [ target:10001:1 ]
  Fast-reroute-priority: low
  Tables:
    cust0001.inet.0      : 10 routes (10 active, 0 holddown, 0 hidden)
    cust0001.iso.0       : 0 routes (0 active, 0 holddown, 0 hidden)
    cust0001.inet6.0     : 0 routes (0 active, 0 holddown, 0 hidden)
    cust0001.mdt.0       : 0 routes (0 active, 0 holddown, 0 hidden)
```

```
user@switch> show route instance cust0002 extensive
cust0002:
  Router ID: 127.9.0.2
  Type: vrf                               State: Active
  Interfaces:
    lo0.2
    irb.6
    irb.5
  Route-distinguisher: 10.0.0.9:2002
  Vrf-import: [ cust0002_vrf_imp ]
  Vrf-export: [ __vrf-export-cust0002-internal__ ]
  Vrf-export-target: [ target:10001:2 ]
  Fast-reroute-priority: low
  Tables:
    cust0002.inet.0      : 10 routes (10 active, 0 holddown, 0 hidden)
    cust0002.iso.0       : 0 routes (0 active, 0 holddown, 0 hidden)
    cust0002.inet6.0     : 0 routes (0 active, 0 holddown, 0 hidden)
    cust0002.mdt.0       : 0 routes (0 active, 0 holddown, 0 hidden)
```

Meaning In the sample output from Spine 2, the routing instances for customer groups 1 and 2 shows the loopback interface and IRB interfaces that are associated with each group. The output also shows the actual route distinguisher and import and export policy configurations.

Verifying that Dynamic MAC Addresses Are Installed

Purpose Verify that for VXLANs v0001, v0002, v0003, and v0004, a dynamic MAC address is installed in the Ethernet switching tables on Leaf 1, Leaf 2, Leaf 3, and Leaf 4.

Action From operational mode, enter the **show ethernet-switching table** command.

```
user@switch> show ethernet-switching table
```

MAC flags (S - static MAC, D - dynamic MAC, L - locally learned, P - Persistent static

SE - statistics enabled, NM - non configured MAC, R - remote PE MAC, O - ovsdb MAC)

Ethernet switching table : 3 entries, 3 learned

Routing instance : default-switch

Vlan	MAC	MAC	Logical	Active
name	address	flags	interface	source
v0001	00:00:5e:00:01:01	DR	esi.7126	
05:00:00:fd:e9:00:00:00:01:00				
v0001	0c:86:10:d6:9d:fe	D	vtep.32769	10.0.0.9
v0001	dc:38:e1:6b:46:03	D	vtep.32770	
10.0.0.10				

MAC flags (S - static MAC, D - dynamic MAC, L - locally learned, P - Persistent static

SE - statistics enabled, NM - non configured MAC, R - remote PE MAC, O - ovsdb MAC)

Ethernet switching table : 3 entries, 3 learned

Routing instance : default-switch

Vlan	MAC	MAC	Logical	Active
name	address	flags	interface	source
v0002	00:00:5e:00:01:01	DR	esi.7127	
05:00:00:fd:e9:00:00:00:02:00				
v0002	0c:86:10:d6:9d:fe	D	vtep.32769	10.0.0.9
v0002	dc:38:e1:6b:46:03	D	vtep.32770	
10.0.0.10				

MAC flags (S - static MAC, D - dynamic MAC, L - locally learned, P - Persistent static

SE - statistics enabled, NM - non configured MAC, R - remote PE MAC, O - ovsdb MAC)

Ethernet switching table : 3 entries, 3 learned

Routing instance : default-switch

Vlan	MAC	MAC	Logical	Active
name	address	flags	interface	source
v0005	00:00:5e:00:01:01	DR	esi.7130	
05:00:00:fd:e9:00:00:00:05:00				
v0005	0c:86:10:d6:9d:fe	D	vtep.32769	10.0.0.9
v0005	dc:38:e1:6b:46:03	D	vtep.32770	
10.0.0.10				

MAC flags (S - static MAC, D - dynamic MAC, L - locally learned, P - Persistent static

SE - statistics enabled, NM - non configured MAC, R - remote PE MAC, O - ovsdb MAC)

Ethernet switching table : 3 entries, 3 learned

Routing instance : default-switch

Vlan	MAC	MAC	Logical	Active
name	address	flags	interface	source
v0006	00:00:5e:00:01:01	DR	esi.7131	
05:00:00:fd:e9:00:00:00:06:00				
v0006	0c:86:10:d6:9d:fe	D	vtep.32769	10.0.0.9
v0006	dc:38:e1:6b:46:03	D	vtep.32770	
10.0.0.10				

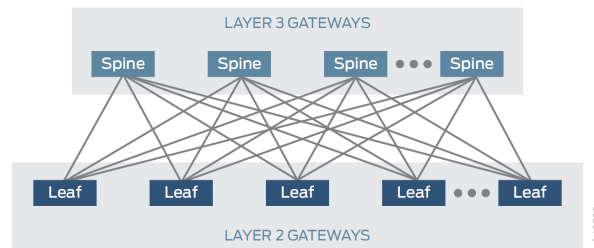
Meaning The sample output from Leaf 1 indicates that it has learned the MAC address 00:00:5e:00:01:01 for a virtual gateway, which is the next-hop of the ESI. The output also indicates that Leaf 1 learned the IRB MAC addresses for Spine 1 and Spine 2, which function as virtual tunnel endpoints (VTEPs).

Example: Configuring EVPN-VXLAN In a Collapsed IP Fabric Topology Within a Data Center

Ethernet VPN (EVPN) is a control plane technology that enables hosts (physical servers and virtual machines [VMs]) to be placed anywhere in a network and remain connected to the same logical Layer 2 overlay network. Virtual Extensible LAN (VXLAN) is a tunneling protocol that creates the data plane for the Layer 2 overlay network.

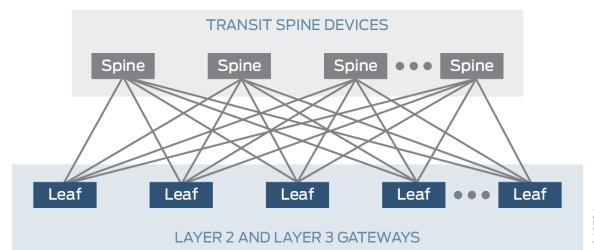
The physical underlay network over which EVPN-VXLAN is commonly deployed is a two-layer IP fabric, which includes spine and leaf devices as shown in [Figure 14 on page 99](#). The spine devices—for example, QFX10000 switches—provide connectivity between the leaf devices, and the leaf devices—for example, QFX5100 switches—provide connectivity to attached hosts. In the overlay network, the leaf devices function as Layer 2 gateways that handle traffic within a VXLAN, and the spine devices function as Layer 3 gateways that handle traffic between VXLANs through the use of integrated routing and bridging (IRB) interfaces. For more information about configuring EVPN-VXLAN in a two-layer IP fabric, see “[Example: Configuring IRB Interfaces in an EVPN-VXLAN Environment to Provide Layer 3 Connectivity for Hosts in a Data Center](#)” on page 53.

Figure 14: Two-Layer IP Fabric



You can also deploy EVPN-VXLAN over a physical underlay network in which the IP fabric is collapsed into a single layer of QFX10000 switches that function as leaf devices. In this collapsed fabric, which is shown in [Figure 15 on page 99](#), the leaf devices serve as both Layer 2 and Layer 3 gateways. In this topology, transit spine devices provide Layer 3 routing functionality only.

Figure 15: Collapsed IP Fabric



This example describes how to configure EVPN-VXLAN, in particular, the Layer 3 gateway, on a leaf device in a collapsed IP fabric topology.

- [Requirements on page 99](#)
- [Overview and Topology on page 100](#)
- [Configuration on page 103](#)
- [Verification on page 107](#)

Requirements

This example uses the following hardware and software components:

- Two routers that function as transit spine devices.
- Three QFX10000 switches running Junos OS Release 15.1X53-D60 or later software. These switches are leaf devices that provide both Layer 2 and Layer 3 gateway functionality.



NOTE: This example focuses on the configuration of the Layer 2 overlay network on a leaf device. The transit spine devices used in this example provide Layer 3 functionality only. As a result, this example does not include the configuration of these devices.

Further, this example provides the configuration for leaf 1 only. The configuration for leaf 1 essentially serves as a template for the configuration of the other leaf devices. For the configuration of the other leaf devices, where appropriate, you can replace leaf 1-specific information with the information specific to the device you are configuring, add additional commands, and so on.

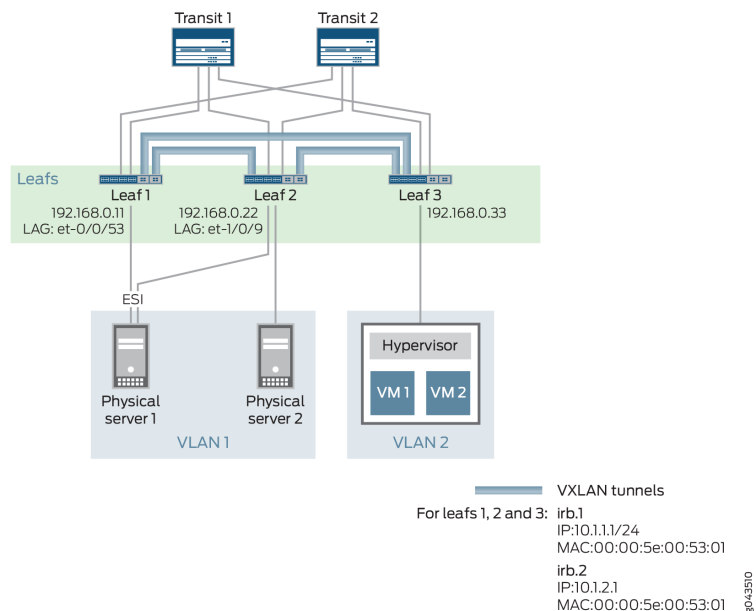
- Two physical (bare-metal) servers and one server with VMs that are supported by a hypervisor.

Overview and Topology

The collapsed IP fabric topology shown in [Figure 16 on page 100](#) includes two transit spine devices, a collapsed IP fabric, which includes three leaf devices that function as both Layer 2 and Layer 3 gateways, two physical servers, and one virtualized server on which VMs and a hypervisor are installed. Physical server 1 is connected to leaf 1 and leaf 2 through a link aggregation group (LAG) interface. On both leaf devices, the interface is assigned the same Ethernet segment identifier (ESI) and set to multihoming active-active mode.

All leaf devices are in the same autonomous system (65200).

Figure 16: Collapsed IP Fabric Topology Within a Data Center



In this topology, an application on physical server 1 needs to communicate with VM 1 on the virtualized server. Physical servers 1 and 2 are included in VLAN 1, and the virtualized server is included in VLAN 2. For communication between VLANs 1 and 2 to occur, two IRB interfaces—`irb.1`, which is associated with VLAN 1, and `irb.2`, which is associated with VLAN 2—must be configured on each leaf device.

The most significant difference between the configuration of an EVPN-VXLAN overlay network deployed over a collapsed IP fabric and the configuration of an overlay network deployed over a two-layer IP fabric is the configuration of the Layer 3 gateway. Therefore, this example focuses on the EVPN-VXLAN configuration, in particular, the Layer 3 gateway configuration on the leaf devices.

For the collapsed IP fabric topology, you can configure the IRB interfaces within an EVPN instance using one of the following methods:

- **Method 1**—For each IRB interface on a particular leaf device, for example, leaf 1, the following is specified:
 - A unique IP address.
 - The *same* MAC address.

For example:

<code>irb.1</code>	IP address: 10.1.1.1/24	MAC address: 00:00:5e:00:53:01
<code>irb.2</code>	IP address: 10.1.2.1/24	MAC address: 00:00:5e:00:53:01

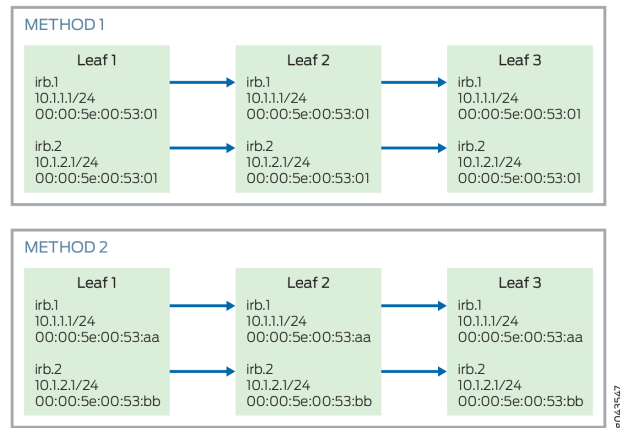
- **Method 2**—For each IRB interface on leaf 1, the following is specified:
 - A unique IP address.
 - A *unique* MAC address.

For example:

<code>irb.1</code>	IP address: 10.1.1.1/24	MAC address: 00:00:5e:00:53:aa
<code>irb.2</code>	IP address: 10.1.2.1/24	MAC address: 00:00:5e:00:53:bb

Regardless of the method that you use to configure the IRB interfaces on leaf 1, if `irb.1` and `irb.2` are also configured on other leaves in the collapsed IP fabric, for example, leaves 2 and 3, you must specify the same configurations that you specified on leaf 1 for those IRB interfaces on leaves 2 and 3. For example, [Figure 17 on page 102](#) shows the configurations for `irb.1` and `irb.2` on leaves 1, 2, and 3 for both methods.

Figure 17: Method 1 and 2 IRB Interface Configurations on Multiple Leaf Devices



NOTE: In this example, method 1 is used to configure the IRB interfaces.

As shown in this example, with the same MAC address configured for each IRB interface on each leaf device, each host uses the same MAC address when sending inter-VLAN traffic regardless of where the host is located or which leaf device receives the traffic. For example, in the topology shown in [Figure 16 on page 100](#), multi-homed physical server 1 in VLAN 1 sends a packet to VM 1 in VLAN 2. If leaf 1 is down, leaf 2 continues to forward the inter-VLAN traffic even without the configuration of a redundant default gateway MAC address.

Note that the configuration of IRB interfaces used in this example does not include a virtual gateway address (VGA) and a corresponding MAC address that establishes redundant default gateway functionality, which is mentioned above. By configuring the same MAC address for each IRB interface on each leaf device, hosts use the local leaf device configured with the common MAC address as the default Layer 3 gateway. Therefore, you eliminate the need to advertise a redundant default gateway and dynamically synchronize the MAC addresses of the redundant default gateway throughout the EVPN control plane. As a result, when configuring each leaf device, you must disable the advertisement of the redundant default gateway by including the **default-gateway do-not-advertise** configuration statement in the **[edit protocols evpn]** hierarchy level in your configuration.



NOTE: Although the IRB interface configuration used in this example does not include a VGA, you can configure it as needed to make EVPN-VXLAN work properly in your collapsed IP fabric topology. If you configure a VGA for each IRB interface, you specify the same IP address for each VGA on each leaf device instead of configuring the same MAC address for each IRB interface on each leaf device as is shown in this example.

When it comes to handling the replication of broadcast, unknown unicast, and multicast (BUM) traffic, note that the configuration on leaf 1:

- includes the **set protocols evpn multicast-mode ingress-replication** command. This command causes leaf 1, which is a hardware VTEP, to handle the replication and sending of BUM traffic instead of a multicast client in the EVPN-VXLAN topology.
- Retains the QFX10000 switch's default setting of disabled for ingress node replication for EVPN-VXLAN. With this feature disabled, if a QFX10000 switch that functions as a VTEP receives a BUM packet intended, for example, for a physical server in a VLAN with the VNI of 1001, the VTEP replicates and sends the packet only to VTEPs on which the VNI of 1001 is configured. If this feature is enabled, the VTEP replicates and sends this packet to all VTEPs in its database, including those that do not have VNI 1001 configured. To prevent a VTEP from needlessly flooding BUM traffic throughout an EVPN-VXLAN overlay network, we strongly recommend that if not already disabled, you disable ingress node replication on each of the leaf devices by specifying the **delete vlans *vlan-name* vxlan ingress-node-replication** command.

Configuration

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

```
Leaf 1 set interfaces et-0/0/53 ether-options 802.3ad ae202
set interfaces ae202 esi 00:11:22:33:44:55:66:77:88:99
set interfaces ae202 esi all-active
set interfaces ae202 aggregated-ether-options lacp active
set interfaces ae202 aggregated-ether-options lacp system-id 00:00:00:04:04:04
set interfaces ae202 unit 0 family ethernet-switching interface-mode trunk
set interfaces ae202 unit 0 family ethernet-switching vlan members 1-2
set interfaces irb unit 1 family inet address 10.1.1.1/24
set interfaces irb unit 1 mac 00:00:5e:00:53:01
set interfaces irb unit 2 family inet address 10.1.2.1/24
set interfaces irb unit 2 mac 00:00:5e:00:53:01
set interfaces lo0 unit 0 family inet address 192.168.0.11/32
set interfaces lo0 unit 1 family inet address 192.168.10.11/32
set protocols bgp group overlay-evpn type internal
set protocols bgp group overlay-evpn local-address 192.168.0.11
set protocols bgp group overlay-evpn family evpn signaling
set protocols bgp group overlay-evpn local-as 65200
set protocols bgp group overlay-evpn multipath
set protocols bgp group overlay-evpn neighbor 192.168.0.22
set protocols bgp group overlay-evpn neighbor 192.168.0.33
set protocols evpn encapsulation vxlan
set protocols evpn extended-vni-list 1001
set protocols evpn extended- vni-list 1002
set protocols evpn multicast-mode ingress-replication
set protocols evpn default-gateway do-not-advertise
set protocols evpn vni-options vni 1001 vrf-target export target:1:1001
set protocols evpn vni-options vni 1002 vrf-target export target:1:1002
set policy-options community comm-leaf_esi members target 9999:9999
set policy-options community com1001 members target:1:1001
```

```

set policy-options community com1002 members target:1:1002
set policy-options policy-statement LEAF-IN term import_leaf_esi from community
  comm-leaf_esi
set policy-options policy-statement LEAF-IN term import_leaf_esi then accept
set policy-options policy-statement vrf-1-to-200 term import_vni1001 from community
  com1001
set policy-options policy-statement vrf-1-to-200 term import_vni1001 then accept
set policy-options policy-statement vrf-1-to-200 term import_vni1002 from community
  com1002
set policy-options policy-statement vrf-1-to-200 term import_vni1002 then accept
set routing-instances VRF_1 instance-type vrf
set routing-instances VRF_1 interface irb.1
set routing-instances VRF_1 interface irb.2
set routing-instances VRF_1 interface lo0.1
set routing-instances VRF_1 route-distinguisher 192.168.0.11:1
set routing-instances VRF_1 vrf-target target:1:1
set switch-options vtep-source-interface lo0.0
set switch-options route-distinguisher 192.168.0.11:5000
set switch-options vrf-import LEAF-IN
set switch-options vrf-import vrf-1-to-200
set switch-options vrf-target target:9999:9999
set vlans bd1 vlan-id 1
set vlans bd1 l3-interface irb.1
set vlans bd1 vxlan vni 1001
delete vlans bd1 vxlan ingress-node-replication
set vlans bd2 vlan-id 2
set vlans bd2 l3-interface irb.2
set vlans bd2 vxlan vni 1002
delete vlans bd2 vxlan ingress-node-replication

```

Configuring EVPN-VXLAN on Leaf 1

Step-by-Step Procedure

1. Enable physical server 1 to be multihomed to leaf 1 and leaf 2 by configuring an aggregated Ethernet interface, specifying an ESI for the interface, and setting the mode so that the connections to both leaf devices are active.



NOTE: When configuring the ae202 interface on leaf 2, you must specify the same ESI (00:11:22:33:44:55:66:77:88:99) that is specified for the same interface on leaf 1.

[edit]

```

user@switch# set interfaces et-0/0/53 ether-options 802.3ad ae202
user@switch# set interfaces ae202 esi 00:11:22:33:44:55:66:77:88:99
user@switch# set interfaces ae202 esi all-active
user@switch# set interfaces ae202 aggregated-ether-options lacp active
user@switch# set interfaces ae202 aggregated-ether-options lacp system-id
  00:00:00:04:04:04
user@switch# set interfaces ae202 unit 0 family ethernet-switching interface-mode
  trunk
user@switch# set interfaces ae202 unit 0 family ethernet-switching vlan members
  1-2

```


2. Configure two IRB interfaces, each with unique IP addresses and the same MAC address.

```
[edit]
user@switch# set interfaces irb unit 1 family inet address 10.1.1.1/24
user@switch# set interfaces irb unit 1 mac 00:00:5e:00:53:01
user@switch# set interfaces irb unit 2 family inet address 10.1.2.1/24
user@switch# set interfaces irb unit 2 mac 00:00:5e:00:53:01
```

3. Configure a loopback interface (lo0.0) for the leaf device and a logical loopback address (lo0.1) for the EVPN routing instance (VRF-1).

```
[edit]
user@switch# set interfaces lo0 unit 0 family inet address 192.168.0.11/32
user@switch# set interfaces lo0 unit 1 family inet address 192.168.10.11/32
```

4. Set up the IBGP overlay network.

```
[edit]
user@switch# set protocols bgp group overlay-evpn type internal
user@switch# set protocols bgp group overlay-evpn local-address 192.168.0.11
user@switch# set protocols bgp group overlay-evpn family evpn signaling
user@switch# set protocols bgp group overlay-evpn local-as 65200
user@switch# set protocols bgp group overlay-evpn multipath
user@switch# set protocols bgp group overlay-evpn neighbor 192.168.0.22
user@switch# set protocols bgp group overlay-evpn neighbor 192.168.0.33
```

5. Set up the EVPN-VXLAN domain, which entails determining which VNIs are included in the domain, specifying that leaf 1, which is a hardware VTEP, handles the replication and sending of BUM traffic, disabling the advertisement of the redundant default gateway throughout the EVPN control plane, and specifying a route target for each VNI.

```
[edit]
user@switch# set protocols evpn encapsulation vxlan
user@switch# set protocols evpn extended-vni-list 1001
user@switch# set protocols evpn extended-vni-list 1002
user@switch# set protocols evpn multicast-mode ingress-replication
user@switch# set protocols evpn default-gateway do-not-advertise
user@switch# set protocols evpn vni-options vni 1001 vrf-target export target:1:1001
user@switch# set protocols evpn vni-options vni 1002 vrf-target export target:1:1002
```

6. Set up communities for the VNIs, and create policies that import and accept the overlay routes.

```
[edit]
user@switch# set policy-options community comm-leaf_esi members target
9999:9999
user@switch# set policy-options community com1001 members target:1:1001
user@switch# set policy-options community com1002 members target:1:1002
user@switch# set policy-options policy-statement LEAF-IN term import_leaf_esi
from community comm-leaf_esi
user@switch# set policy-options policy-statement LEAF-IN term import_leaf_esi
then accept
user@switch# set policy-options policy-statement vrf-1-to-200 term import_vni1001
from community com1001
user@switch# set policy-options policy-statement vrf-1-to-200 term import_vni1001
then accept
```

```

user@switch# set policy-options policy-statement vrf-1-to-200 term import_vni1002
from community com1002
user@switch# set policy-options policy-statement vrf-1-to-200 term import_vni1002
then accept

```

7. Set up an EVPN routing instance.

```

[edit]
user@switch# set routing-instances VRF_1 instance-type vrf
user@switch# set routing-instances VRF_1 interface irb.1
user@switch# set routing-instances VRF_1 interface irb.2
user@switch# set routing-instances VRF_1 interface lo0.1
user@switch# set routing-instances VRF_1 route-distinguisher 192.168.0.11:1
user@switch# set routing-instances VRF_1 vrf-target target:1:1

```



NOTE: In the above EVPN routing instance configuration, a unique logical loopback interface (lo0.1) is specified, and an IP address for the interface is specified using the `set interfaces lo0 unit logical-unit-number family inet address ip-address/prefix` command. All items configured in the above routing instance except for the logical loopback interface are required for EVPN. However, the configuration of a logical loopback interface and associated IP address are required to ensure that VXLAN control packets are properly processed.

8. Configure the switch options to use loopback interface lo0.0 as the source interface of the VTEP, set a route distinguisher, and import the route targets for the three communities into the EVPN (MAC) table.

```

[edit]
user@switch# set switch-options vtep-source-interface lo0.0
user@switch# set switch-options route-distinguisher 192.168.0.11:5000
user@switch# set switch-options vrf-import LEAF-IN
user@switch# set switch-options vrf-import vrf-1-to-200
user@switch# set switch-options vrf-target target:9999:9999

```

9. Configure VLANs to which IRB interfaces and VXLAN VNIs are associated.

```

[edit]
user@switch# set vlans bd1 vlan-id 1
user@switch# set vlans bd1 l3-interface irb.1
user@switch# set vlans bd1 vxlan vni 1001
user@switch# set vlans bd2 vlan-id 2
user@switch# set vlans bd2 l3-interface irb.2
user@switch# set vlans bd2 vxlan vni 1002

```

10. If not already disabled, disable ingress node replication to prevent leaf 1 from needlessly flooding BUM traffic throughout the EVPN-VXLAN overlay network.

```

[edit]
user@switch# delete vlans bd1 vxlan ingress-node-replication
user@switch# delete vlans bd2 vxlan ingress-node-replication

```

Verification

The section describes the following verifications for this example:

- [Verifying the IRB Interfaces on page 107](#)
- [Verifying the VTEP Interfaces on page 107](#)
- [Verifying the EVPN Routing Instance on page 107](#)

Verifying the IRB Interfaces

Purpose Verify that the IRB interfaces are up and running.

Action Display the status of the IRB interfaces:

```
user@leaf1> show interfaces irb terse
```

Interface	Admin	Link	Proto	Local	Remote
irb	up	up			
irb.1	up	up	inet	10.1.1.1/24	
irb.2	up	up	inet	10.1.2.1/24	

Meaning The IRB interfaces are up and running.

Verifying the VTEP Interfaces

Purpose Verify the status of the VTEP interfaces.

Action Display the status of the VTEP interfaces:

```
user@leaf1> show interfaces vtep terse
```

Interface	Admin	Link	Proto	Local	Remote
vtep	up	up			
vtep.32769	up	up	eth-switch		
vtep.32770	up	up	eth-switch		
vtep.32771	up	up	eth-switch		

Meaning The interfaces for each of the VTEPs is up. Therefore, the VTEP interfaces are functioning normally.

Verifying the EVPN Routing Instance

Purpose Verify the routing table for VRF_1.

Action Verify the routing table for the EVPN routing instance VRF_1.

```

user@leaf1> show route table VRF_1.inet.0
VRF_1.inet.0: 5 destinations, 5 routes (5 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

10.1.1.0/24      *[Direct/0] 00:07:38
                  > via irb.1
10.1.1.1/32      *[Local/0] 00:07:38
                  Local via irb.110.1.2.0/24      *[Direct/0] 00:07:38
                  > via irb.2
10.1.2.1/32      *[Local/0] 00:07:38
                  Local via irb.2
192.168.10.11/32 *[Direct/0] 00:07:38
                  > via lo0.1

```

Meaning The EVPN routing instance is functioning correctly.

Preventing Traffic Loss in an EVPN/VXLAN Environment With GRES and NSR

The graceful Routing Engine switchover (GRES) feature in Junos OS enables QFX10000 switches to continue forwarding packets, even if one Routing Engine fails. To preserve routing during a switchover, GRES is combined with nonstop active routing (NSR). Although GRES preserves interface and kernel information, in an EVPN/VXLAN environment, a QFX10000 switch could experience Layer 3 traffic loss if traffic is flowing during the switchover or reboot of the Routing Engine.

To prevent traffic flows from being dropped in such a scenario, configure a hold-time interval that prevents IP phantom addresses from being added to the routing tables. During this interval, the routes are maintained in the kernel. After the interval expires, the phantom routes are added to the appropriate routing tables before they are deleted from the kernel.



NOTE: The recommended hold-time value in an EVPN/VXLAN environment is 300 seconds (5 minutes).

- Enable GRES
- Enable NSR

To specify a hold-time interval that prevents phantom routes from being added to the routing table during a switchover until the interval expires:

1. Specify a hold-time interval of 300.

```

[edit routing-options]
user@swtich# set nsr-phantom-holdtime seconds 300.

```



NOTE: The hold-time interval range is 0 through 10000.

Related Documentation • *Understanding Graceful Routing Engine Switchover*

PART 3

Configuration Statements and Operational Commands

- [EVPN-VXLAN Configuration Statements on page 113](#)
- [EVPN-VXLAN Operational Commands on page 137](#)

CHAPTER 3

EVPN-VXLAN Configuration Statements

- [designated-forwarder-election-hold-time \(evpn\) on page 114](#)
- [encapsulation \(Logical Interface\) on page 115](#)
- [evpn on page 119](#)
- [extended-vni-list on page 120](#)
- [ingress-node-replication \(EVPN\) on page 121](#)
- [ip-prefix-routes on page 122](#)
- [ip-prefix-support on page 124](#)
- [multicast-mode \(EVPN\) on page 125](#)
- [no-default-gateway-ext-comm on page 126](#)
- [nsr-phantom-holdtime on page 127](#)
- [proxy-macip-advertisement on page 128](#)
- [route-distinguisher on page 129](#)
- [vni-options on page 131](#)
- [vrf-export on page 132](#)
- [vrf-import on page 133](#)
- [vrf-target on page 134](#)
- [vxlan on page 135](#)

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

Syntax	<code>designated-forwarder-election-hold-time <i>seconds</i> { encapsulation-type extended-vni-list extended-vni-all no-default-gateway-ext-comm }</code>
Hierarchy Level	[edit routing-instances protocols evpn] [edit protocols evpn]
Release Information	Statement introduced in Junos OS Release 14.1X53-D15 for QFX Series switches.
Description	A designated forwarder (DF) is required when customer edge devices (CEs) are multihomed to more than one provider edge (PE) device. 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	<ul style="list-style-type: none">• EVPN Multihoming Overview on page 27

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 <i>interface-name</i> unit <i>logical-unit-number</i>], [edit logical-systems <i>logical-system-name</i> interfaces <i>interface-name</i> unit <i>logical-unit-number</i>], [edit interfaces rlsq <i>number</i> unit <i>logical-unit-number</i>]
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.
Description	Configure a logical link-layer encapsulation type.
Options	<p>atm-ccc-cell-relay—Use ATM cell-relay encapsulation.</p> <p>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.</p> <p>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.</p> <p>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.</p> <p>atm-nlpid—Use ATM NLPID encapsulation. When you use this encapsulation type, you can configure the inet family only.</p> <p>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.</p> <p>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.</p> <p>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.</p> <p>atm-tcc-snap—Use ATM SNAP encapsulation on translational cross-connect (TCC) circuits.</p>

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 device over a time-division multiplexing (TDM) link. This encapsulation type enables the PE device 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—You use PPP over Ethernet encapsulation to configure an underlying Ethernet interface for a dynamic PPPoE logical interface.

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.

Required Privilege	interface—To view this statement in the configuration.
Level	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 and 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-time seconds {
    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)
    control-word (EVPN)
    interface interface-name {
        ignore-encapsulation-mismatch;
        interface-mac-limit limit {
            packet-action drop;
        }
        no-mac-learning;
        static-mac mac-address;
    }
    interface-mac-limit 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 QFX 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	<ul style="list-style-type: none">• Implementing EVPN-VXLAN for Data Centers on page 20• <i>Configuring EVPN Routing Instances</i>• <i>Tracing EVPN Traffic and Operations</i>• Implementing EVPN-VXLAN for Data Centers on page 20

extended-vni-list

Syntax	extended-vni-list xx
Hierarchy Level	For QFX Series switches: protocols For MX Series Routers: Routing-instance { Protocols { evpn } }
Release Information	Statement introduced in Junos OS Release 14.1X53-D30.
Description	The extended-vni-list establishes which VXLAN Network Identifiers (VNI) 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.
Required Privilege Level	routing—To view this statement in the configuration. routing-control—To add this statement to the configuration.
Related Documentation	<ul style="list-style-type: none">• extended-vni-list on page 120• show configuration protocols evpn on page 138

ingress-node-replication (EVPN)

Syntax	ingress-node-replication;
Hierarchy Level	[edit vlans] { vxlan }
Release Information	Statement introduced in Junos OS Release 14.1X53-D30 for QFX Series switches.
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 provider edge (PE) instead of the MPLS ES-label. This is called local bias for EVPN-VXLAN. Each EVPN PE tracks the IP address of its peer multihomed EVPN PE that share 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 multidestination traffic. For egress traffic, there is no forwarding of any multidestination packets to the same multihomed 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	<ul style="list-style-type: none"> • Understanding EVPN with VXLAN Data Plane Encapsulation on page 14

ip-prefix-routes

Syntax `ip-prefix-routes {
 advertise direct-nexthop;
 encapsulation vxlan;
 <export routing-policy-name>;
 vni number;
}`

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

Release Information Statement introduced in Junos OS Release 15.1x53-D60 for QFX10000 Series switches.

Description In an Ethernet VPN (EVPN) Virtual Extensible LAN (VXLAN) environment, enable the switch to advertise the IP prefix associated with a specified customer domain as a pure type-5 route to remote data centers. You use this feature when the Layer 2 domain does not exist at the remote data centers. A pure type-5 route advertises the summary IP prefix and includes a BGP extended community called a router MAC, which is used to carry the MAC address of the sending switch and to provide next-hop reachability information for the prefix. In contrast, a standard type-5 route requires a gateway IP address as a next-hop overlay and a supporting type-2 route to provide recursive route resolution. The data packets are sent as Layer 2 frames that are encapsulated in the VXLAN header. On QFX10000 switches, only fully resolved next-hop—that is, EVPN pure type-5—routes are currently supported.



CAUTION: Pure type-5 routing for EVPN/VXLAN was introduced in Junos OS Release 15.1x53-D30 for QFX10002 switches only. In that release, this statement is `ip-prefix-support forwarding-mode symmetric`. Starting with Junos OS Release 15.1x53-D60, the statement is `ip-prefix-routes advertise direct-nexthop`. Any configuration with the original `ip-prefix-support` statement is automatically upgraded to the new `ip-prefix-routes` statement when you upgrade to Junos OS Release 15.1x53-D60 or later.



NOTE: Pure type-5 routing is supported on all QFX10000 switches starting in Junos OS Release 15.1x53-D60.

Options The `advertise direct-nexthop`, `encapsulation vxlan` and `vni number` options are required. The `export routing-policy-name` option is optional.

advertise direct-nexthop—Enable the switch to send IP prefix information using an EVPN pure type-5 route, which includes a router MAC extended community used to send the MAC address of the switch. This router MAC extended community provides

next-hop reachability without requiring an overlay next-hop or supporting type-2 route.

encapsulation vxlan—Specify to encapsulate forwarded traffic in VXLAN for transmission to the remote data center.

export *routing-policy-name*—(Optional) Specify the name of the routing policy configured at the **[edit policy-options policy-statement *policy-statement-name*]** hierarchy level to apply to the routes for the specified customer domain. Applying an export policy allows you to further control the IP prefixes to advertise or to suppress through EVPN type-5 routes for each customer. You can apply a separate export routing policy to one or more customer domains. This allows each customer to each have its own policy.

vni *number*—Specify the identifier associated with a customer domain. Each customer domain must have a unique identifier.

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

Related Documentation	• Understanding EVPN Pure Route Type-5 on QFX Series Switches on page 24
	• <i>policy-statement</i>

ip-prefix-support

Syntax ip-prefix-support {
 encapsulation vxlan;
 <export *routing-policy-name*>;
 forwarding-mode symmetric;
 vni *number*;
 }

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

Release Information Statement introduced in Junos OS Release 15.1x53-D30 on QFX10002 switches.

Description In an Ethernet VPN (EVPN) Virtual Extensible LAN (VXLAN) environment, enable the provider edge (PE) switch to forward the IP prefix associated with a specified customer domain in scenarios where the Layer 2 domain does not extend across data centers. Such routes are referred to as EVPN pure type-5 routes. On QFX switches, only fully resolved next-hop routes are supported. The data packets are sent as Layer 2 frames that are encapsulated in the VXLAN header. A unique VXLAN numerical identifier is associated with each customer domain.

When the advertisement of the type-5 route is enabled, you can optionally use an export routing policy configured at the **[edit policy-options policy-statement *policy-name*]** hierarchy level to further control the IP prefixes to advertise or suppress through EVPN type-5 routes for each customer. The policy is applied to the routes that reside in the customer's inet.0 routing table by specifying a *routing-instance-name* configured at the **[edit routing-instances]** hierarchy level. This *routing-instance-name* refers to a Layer 3 virtual routing and forwarding (VRF) domain for a specific customer. The corresponding name of the table is *instance.name.inet.0*. This table is created by default when you configure a routing instance.



CAUTION: This statement is deprecated starting with Junos OS Release 15.1x53-D60 and has been replaced with **ip-prefix-routes**. The statement **ip-prefix-support forwarding-mode symmetric** is now **ip-prefix-routes advertise direct-nexthop**. Any configuration with the original **ip-prefix-support** statement is automatically upgraded to the new **ip-prefix-routes** statement when you upgrade to Junos OS Release 15.1x53-D60 or later.

Options The **forwarding-mode symmetric**, **encapsulation vxlan** and **vni *number*** options are required. The **export *policy-name*** option is optional.

encapsulation vxlan—Specify to encapsulate forwarded traffic in VXLAN for transmission to the other data center.

export *routing-policy-name*—(Optional) Specify the name of the routing policy configured at the **[edit policy-options policy-statement *policy-name*]** hierarchy level to apply to the routes for the specified customer domain. Applying an export policy allows you

to further control the IP prefixes to advertise or to suppress for each customer. You can apply a separate export routing policy to one or more customer domains. This allows each customer to each have its own policy.

forwarding-mode symmetric—Enable the receiver to resolve the next-hop route using only information carried in the type-5 route.

vni number—Specify the identifier associated with a customer domain. Each customer domain must have a unique identifier.

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

Related Documentation

- [Understanding EVPN Pure Route Type-5 on QFX Series Switches on page 24](#)
- *policy-statement*

multicast-mode (EVPN)

Syntax multicast-mode ingress-replication;

Hierarchy Level [edit protocols [evpn](#)],
[edit routing-instances *routing-instance-name* protocols [evpn](#)],

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

Description Configure the multicast server mode for delivering traffic and packets for Ethernet VPN (EVPN). This statement is required for a VXLAN EVPN instance.



NOTE: If you configure the **multicast-mode** statement, then you must also configure the **encapsulation vxlan** statement.

Options **ingress-replication**—Use ingress replication as the multicast mode for delivering broadcast, unknown unicast, and multicast (BUM) traffic and multicast packets across routers and switches.

Default: ingress-replication

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


Related Documentation

- [Understanding EVPN with VXLAN Data Plane Encapsulation on page 14](#)
- *Example: Configuring an EVPN Control Plane and VXLAN Data Plane on MX Series Routers and QFX5100 Switches*

no-default-gateway-ext-comm

Syntax	no-default-gateway-ext-comm;
Hierarchy Level	[edit routing-instances <i>name</i> protocols evpn]
Release Information	Statement introduced in Junos OS Release 15.1
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 Ethernet VPNs (EVPNs).
Required Privilege Level	interface—To view this statement in the configuration. interface-control—To add this statement to the configuration.
Related Documentation	<ul style="list-style-type: none">• EVPN Multihoming Overview on page 27

nsr-phantom-holdtime

Syntax	<code>nsr-phantom-holdtime seconds;</code>
Hierarchy Level	[edit routing-options]
Release Information	Statement introduced in Junos OS Release 15.1x53-D60 for QFX10000 switches.
Description	<p>Specify to hold phantom IP addresses, that is, prevent these routes from being added to routing tables, for a specified period of time. During this hold-time interval, these routes are maintained in the kernel. After the hold time expires, the routes are added to the routing tables.</p> <p>We strongly recommend that you configure this statement before you perform a graceful Routing Engine switchover (GRES) when nonstop routing (NSR) is enabled. Doing so prevents traffic loss because routes are added to the routing tables after the hold-time interval expires, but before they are deleted from the kernel during a switchover.</p>
Options	<p>seconds—Specify the interval of time to prevent phantom IP addresses from being added to the routing tables. After this interval expires, the routes are added to the routing tables.</p>
<div>  <p>BEST PRACTICE: In an EVPN/VXLAN environment with NSR and GRES enabled, the recommended hold-time value is 300 (5 minutes)</p> </div>	
Range: 0 through 10000	
Required Privilege Level	<p>routing—To view this statement in the configuration.</p> <p>routing-control—To add this statement to the configuration.</p>
Related Documentation	<ul style="list-style-type: none"> • <i>Understanding Graceful Routing Engine Switchover</i> • Preventing Traffic Loss in an EVPN/VXLAN Environment With GRES and NSR on page 108 • <i>Example: Configuring Nonstop Active Routing on Switches</i>

proxy-macip-advertisement

Syntax	<code>proxy-macip-advertisement;</code>
Hierarchy Level	[edit interfaces irb unit <i>logical-unit-number</i>]
Release Information	Statement introduced in Junos OS Release 15.1X53-D60 for QFX Series switches.
Description	<p>Configure a QFX10000 switch that functions as a Layer 3 gateway in an Ethernet VPN-Virtual Extensible LAN (EVPN-VXLAN) with integrated routing and bridging (IRB) interfaces that advertise the MAC and IP routes (MAC+IP type 2 routes) for hosts in the topology.</p> <p>In an EVPN-VXLAN topology, leaf devices typically function as Layer 2 gateways. As such, these devices can advertise only the MAC routes (EVPN type 2 routes) for the attached hosts. Since the Layer 2 gateways are unable to resolve the MAC-to-IP bindings for the hosts, each of the spine devices, which typically function as Layer 3 gateways, must rely on the Address Resolution Protocol (ARP) and the Neighbor Discovery Protocol (NDP) to discover and install the bindings.</p> <p>With the proxy advertisement feature enabled, after receiving a host MAC route advertisement from a Layer 2 gateway, and ARP and NDP resolve the MAC-to-IP bindings, the QFX10000 switch in turn advertises the host MAC and IP routes along with the next hop, which is set to the Layer 2 gateway to which the host is attached. Upon receipt of this advertisement, Layer 2 and 3 gateways in the topology install the MAC-to-IP bindings along with the associated next hops. When any of these gateways receives a packet with a destination MAC that matches an address in its MAC table, the gateway can check the next hop associated with the MAC address and forward the packet directly to the Layer 2 gateway to which the host is attached. This resulting packet flow eliminates the need for the packet to be forwarded first to a Layer 3 gateway, which then forwards the packet to the Layer 2 gateway.</p> <p>Enabling this feature in an EVPN-VXLAN topology that includes both Layer 2 and Layer 3 gateways is mandatory, while enabling the feature in a topology that includes only Layer 3 gateways is optional.</p>
Required Privilege Level	interface—To view this statement in the configuration. interface-control—To add this statement to the configuration.

route-distinguisher

Syntax	<code>route-distinguisher (as-number:id ip-address:id);</code>
Hierarchy Level	<p>[edit logical-systems <i>logical-system-name</i> routing-instances <i>routing-instance-name</i>], [edit logical-systems <i>logical-system-name</i> routing-instances <i>routing-instance-name</i> protocols l2vpn mesh-group <i>mesh-group-name</i>], [edit logical-systems <i>logical-system-name</i> routing-instances <i>routing-instance-name</i> protocols vpls mesh-group <i>mesh-group-name</i>], [edit routing-instances <i>routing-instance-name</i>], [edit routing-instances <i>routing-instance-name</i> protocols l2vpn mesh-group <i>mesh-group-name</i>], [edit routing-instances <i>routing-instance-name</i> protocols vpls mesh-group <i>mesh-group-name</i>]</p>
Release Information	<p>Statement introduced before Junos OS Release 7.4.</p> <p>Statement introduced in Junos OS Release 11.1 for EX Series switches.</p> <p>Support at [edit routing-instances <i>routing-instance-name</i> protocols vpls mesh-group <i>mesh-group-name</i>] hierarchy level introduced in Junos OS Release 11.2.</p> <p>Statement introduced in Junos OS Release 12.3 for ACX Series routers.</p> <p>Support at [edit routing-instances <i>routing-instance-name</i> protocols l2vpn mesh-group <i>mesh-group-name</i>] hierarchy level introduced in Junos OS Release 13.2.</p> <p>Statement introduced in Junos OS Release 14.1X53-D30 for QFX Series switches.</p>
Description	<p>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 (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. If the instance type is vrf, the route-distinguisher statement is required.</p> <p>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.</p> <p>For other types of VPNs, we recommend that you use a unique RD for each provider edge (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.</p> <p>For Layer 2 VPNs and VPLSs, if you configure mesh groups, the RD in each mesh group must be unique.</p>



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 a 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

Syntax	<pre>vni-options vni <i>vxlan-network-identifier</i> { designated-forwarder-election-hold-time <i>seconds</i>; vrf-target { community; auto import <i>community-name</i>; export <i>community-name</i>; } }</pre>
Hierarchy Level	[edit protocols evpn] [edit routing-instances <i>routing-instance-name</i> protocols evpn]
Release Information	Statement introduced in Junos OS Release 14.1X53-D30 for QFX Series switches.
Description	Configure a designated forwarder election hold time and specific route targets (RTs) for each VXLAN network identifier (VNI).
Required Privilege Level	routing—To view this statement in the configuration. routing-control—To add this statement to the configuration.
Related Documentation	<ul style="list-style-type: none">• extended-vni-list on page 120• EVPN Multihoming Overview on page 27

vrf-export

Syntax	<code>vrf-export [<i>policy-names</i>];</code>
Hierarchy Level	<code>[edit logical-systems <i>logical-system-name</i> routing-instances <i>routing-instance-name</i>],</code> <code>[edit logical-systems <i>logical-system-name</i> routing-instances <i>routing-instance-name</i> protocols</code> <code> vpls mesh-group <i>mesh-group-name</i>]</code> <code>[edit routing-instances <i>routing-instance-name</i>]</code> <code>[edit routing-instances <i>routing-instance-name</i> protocols vpls mesh-group <i>mesh-group-name</i>]</code> <code>[edit switch-options]</code>
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	<p>Specify how routes are exported from the local PE router's VRF table (<i>routing-instance-name</i>.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.</p> <p>You can configure multiple export policies on the PE router or PE switch.</p>
Default	If the instance-type is vrf , vrf-export is a required statement. The default action is to reject.
Options	<i>policy-names</i> —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	<ul style="list-style-type: none">• Implementing EVPN-VXLAN for Data Centers on page 20• <i>instance-type</i>• <i>Configuring Policies for the VRF Table on PE Routers in VPNs</i>

vrf-import

Syntax	<code>vrf-import [<i>policy-names</i>];</code>
Hierarchy Level	<p>[edit logical-systems <i>logical-system-name</i> routing-instances <i>routing-instance-name</i>],</p> <p>[edit logical-systems <i>logical-system-name</i> routing-instances <i>routing-instance-name</i> protocols vpls mesh-group <i>mesh-group-name</i>]</p> <p>[edit routing-instances <i>routing-instance-name</i>]</p> <p>[edit routing-instances <i>routing-instance-name</i> protocols vpls mesh-group <i>mesh-group-name</i>]</p> <p>[edit switch-options]</p>
Release Information	<p>Statement introduced before Junos OS Release 7.4.</p> <p>Statement introduced in Junos OS Release 11.1 for EX Series switches.</p> <p>Statement introduced in Junos OS Release 14.1X53-D30 for QFX Series switches.</p>
Description	<p>Specify how routes are imported into the virtual routing and forwarding (VRF) table (<i>routing-instance-name</i>.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.</p> <p>You can configure multiple import policies on the PE router or switch.</p>
Default	If the instance type is vrf , vrf-import is a required statement. The default action is to accept.
Options	<i>policy-names</i> —Names for the import policies.
Required Privilege Level	<p>routing—To view this statement in the configuration.</p> <p>routing-control—To add this statement to the configuration.</p>
Related Documentation	<ul style="list-style-type: none"> • Implementing EVPN-VXLAN for Data Centers on page 20 • <i>instance-type</i> • <i>Configuring Policies for the VRF Table on PE Routers in VPNs</i>

vrf-target

Syntax	<pre>vrf-target { community; auto import community-name; export community-name; }</pre>
Hierarchy Level	<p>[edit logical-systems <i>logical-system-name</i> routing-instances <i>routing-instance-name</i>], [edit logical-systems <i>logical-system-name</i> routing-instances <i>routing-instance-name</i> protocols l2vpn mesh-group <i>mesh-group-name</i>], [edit logical-systems <i>logical-system-name</i> routing-instances <i>routing-instance-name</i> protocols vpls mesh-group <i>mesh-group-name</i>], [edit routing-instances <i>routing-instance-name</i>], [edit routing-instances <i>routing-instance-name</i> protocols l2vpn mesh-group <i>mesh-group-name</i>], [edit routing-instances <i>routing-instance-name</i> protocols vpls mesh-group <i>mesh-group-name</i>], [edit switch-options]</p>
Release Information	<p>Statement introduced before Junos OS Release 7.4.</p> <p>Statement introduced in Junos OS Release 11.1 for EX Series switches.</p> <p>Statement introduced in Junos OS Release 12.3 for ACX Series routers.</p> <p>Statement introduced in Junos OS Release 14.1X53-D30 for QFX Series switches. auto option was also added at this time.</p>
Description	<p>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.</p> <p>You can still create more complex policies by explicitly configuring VRF import and export policies using the import and export options.</p>
Options	<p>community—Community name.</p> <p>auto—Automatically derives the route target (RT) for supported QFX Series switches.</p> <p>import community-name—Allowed communities accepted from neighbors.</p> <p>export community-name—Allowed communities sent to neighbors.</p>
Required Privilege Level	<p>routing—To view this statement in the configuration.</p> <p>routing-control—To add this statement to the configuration.</p>
Related Documentation	<ul style="list-style-type: none"> • Implementing EVPN-VXLAN for Data Centers on page 20 • Configuring Policies for the VRF Table on PE Routers in VPNs

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

vxlan

Syntax	<pre> vxlan { encapsulate-inner-vlan ingress-node-replication multicast-group ovsdb-managed unreachable-vtep-aging-timer vni } </pre>
Hierarchy Level	[edit vlans]
Release Information	<p>Statement introduced in Junos OS Release 14.1X53-D10.</p> <p>ingress-node-replication option added for QFX Series switches in Junos OS Release 14.1X53-D30.</p>
Description	Configure support for Virtual Extensible LANs (VXLANs) on a Juniper Networks device.
Options	The remaining statements are explained separately.
Required Privilege Level	<p>routing—To view this statement in the configuration.</p> <p>routing-control—To add this statement to the configuration.</p>
Related Documentation	<ul style="list-style-type: none"> • Understanding VXLANs on page 3 • <i>Manually Configuring VXLANs on QFX Series Switches</i> • <i>Examples: Manually Configuring VXLANs on QFX Series Switches</i>

CHAPTER 4

EVPN-VXLAN Operational Commands

- `show configuration protocols evpn`
- `show evpn ip-prefix-database`
- `show evpn l3-context`
- `show route table`

show configuration protocols evpn

Syntax	show configuration protocols evpn
Release Information	Statement introduced in Junos OS Release 14.1X53-D15 for QFX Series switches.
Description	An Ethernet VPN (EVPN) enables you to connect a group of dispersed customer sites using a Layer 2 virtual bridge. This command displays configured EVPN information.
Required Privilege Level	view
Related Documentation	<ul style="list-style-type: none"> • extended-vni-list on page 120 • vni-options on page 131
List of Sample Output	command-name (optional-text) on page 138
Output Fields	Table 5 on page 138 describes the output fields for the command show-protocols-evpn .

Table 5: Output Fields for Command show protocols evpn

Field Name	Field Description
encapsulation	Encapsulation method will always be vxlan .
extended-vni-list	Lists the VXLAN Network Identifiers (VNIs) that are part of the EVPN/VXLAN MP-BGP domain. These VNIs are configured with the command extended-vni-list .
vni-routing-options	Displays route targets (RTs) for each VXLAN Network Identifier (VNI) instance configured with the vni-options command.

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-routing-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 evpn ip-prefix-database

Syntax	<code>show evpn ip-prefix-database</code> <code><extensive></code> <code><direction (imported exported)></code> <code><esi number></code> <code><ethernet-tag number></code> <code><family (inet inet6)></code> <code><gateway ip-address></code> <code><l3-context routing-instance-name></code> <code><nexthop ip-address></code> <code><prefix ip-prefix></code>
Release Information	Command introduced in Junos OS Release 15.1X53-D30 for QFX10002 switches. Support added for QFX10008 and QFX10016 switches in Junos OS Release 15.1x53-D60.
Description	Display Ethernet VPN (EVPN) database information for imported and exported IP prefixes.
Options	none —Display standard information for all EVPN imported and exported IP prefixes. extensive —Display the specified level of output. direction (imported exported) —(Optional) Display the specified subset of IP prefixes. esi number —(Optional) Display the IP prefix associated with the specified Ethernet segment identifier. ethernet-tag number —(Optional) Display the IP prefix associated with the specified Ethernet tag. family (inet inet6) —(Optional) Display IP prefixes for the specified protocol family. gateway ip-address —(Optional) Display the IP Prefix associated with the overlay gateway IP address. l3-context routing-instance-name —(Optional) Display EVPN IP prefix information for the specified Layer 3 virtual routing and forwarding (VRF) instance. nexthop ip-address —(Optional) Display EVPN IP prefix information for the specified underlay next-hop IP address. prefix ip-prefix —(Optional) Display EVPN database information for the specified IP prefix.
Required Privilege Level	view
Related Documentation	<ul style="list-style-type: none">• show evpn l3-context on page 147• show route table on page 150
List of Sample Output	show evpn ip-prefix-database on page 141 show evpn ip-prefix-database extensive on page 142

Output Fields Table 6 on page 141 lists the output fields for the **show evpn ip-prefix-database** command. Output fields are listed in the approximate order in which they appear.

Table 6: show evpn ip-prefix-database Output Fields

Field Name	Field Description	Level of Output
L3 context	Name of virtual routing and forwarding (VRF) instance.	All levels
EVPN Exported Prefixes Prefix	List of exported IP prefixes.	All levels
EVPN route status	For exported prefixes only, status of route: Created .	level-of-output none
EVPN imported Prefixes	List of imported EVPN IP prefixes.	All levels
Etag	Ethernet tag	level-of-output none
Route distinguisher	IP address identifier for the IP prefix route.	All levels
VNI	Virtual network identifier for the Layer 3 virtual and routing forwarding (VRF) for the customer or tenant domain.	All levels
Router MAC	MAC address associated with the IP prefix.	All levels
Nexthop/Overlay GW/ESI	For imported IP prefixes, next-hop IP address.	level-of-output none
Change flags	Trace flags.	level-of-output extensive
Advertisement mode	For exported IP prefixes, type of next-hop address.	level-of-output extensive
Encapsulation	For exported IP prefixes, type of encapsulation	level-of-output extensive
Remote Advertisements	For imported IP prefixes, route distinguisher identifier, MAC address, virtual network identifier, and BGP next-hop address to remote destination.	level-of-output extensive

Sample Output

show evpn ip-prefix-database

```

user@host > show evpn ip-prefix-database
L3 context: VRF-100

IPv4->EVPN Exported Prefixes
Prefix
100.1.0.0/22
100.1.4.0/22
100.1.8.0/22
100.1.12.0/22

EVPN route status
Created
Created
Created
Created

```

```

100.1.16.0/22                                     Created

IPv6->EVPN Exported Prefixes
Prefix                                             EVPN route status
1234:100:1::/64                                  Created
1234:100:1:4::/64                                Created
1234:100:1:8::/64                                Created
1234:100:1:12::/64                               Created
1234:100:1:16::/64                               Created

EVPN->IPv4 Imported Prefixes
Prefix                                             Etag      IP route status
100.2.0.0/22                                     0          Created
  Route distinguisher  VNI/Label  Router MAC  Nexthop/Overlay GW/ESI
    10.255.2.1:100    9101      00:05:86:e1:03:f0  10.255.2.1
    10.255.2.2:100    9101      00:05:86:d0:a6:f0  10.255.2.2
100.2.4.0/22                                     0          Created
  Route distinguisher  VNI/Label  Router MAC  Nexthop/Overlay GW/ESI
    10.255.2.1:100    9101      00:05:86:e1:03:f0  10.255.2.1
    10.255.2.2:100    9101      00:05:86:d0:a6:f0  10.255.2.2
100.2.8.0/22                                     0          Created
  Route distinguisher  VNI/Label  Router MAC  Nexthop/Overlay GW/ESI
    10.255.2.1:100    9101      00:05:86:e1:03:f0  10.255.2.1
    10.255.2.2:100    9101      00:05:86:d0:a6:f0  10.255.2.2
100.2.12.0/22                                    0          Created
  Route distinguisher  VNI/Label  Router MAC  Nexthop/Overlay GW/ESI
    10.255.2.1:100    9101      00:05:86:e1:03:f0  10.255.2.1
    10.255.2.2:100    9101      00:05:86:d0:a6:f0  10.255.2.2
100.2.16.0/22                                    0          Created
  Route distinguisher  VNI/Label  Router MAC  Nexthop/Overlay GW/ESI
    10.255.2.1:100    9101      00:05:86:e1:03:f0  10.255.2.1
    10.255.2.2:100    9101      00:05:86:d0:a6:f0  10.255.2.2

EVPN->IPv6 Imported Prefixes
Prefix                                             Etag      IP route status
1234:100:2::/64                                  0          Created
  Route distinguisher  VNI/Label  Router MAC  Nexthop/Overlay GW/ESI
    10.255.2.1:100    9101      00:05:86:e1:03:f0  10.255.2.1
    10.255.2.2:100    9101      00:05:86:d0:a6:f0  10.255.2.2
1234:100:2:4::/64                                0          Created
  Route distinguisher  VNI/Label  Router MAC  Nexthop/Overlay GW/ESI
    10.255.2.1:100    9101      00:05:86:e1:03:f0  10.255.2.1
    10.255.2.2:100    9101      00:05:86:d0:a6:f0  10.255.2.2
1234:100:2:8::/64                                0          Created
  Route distinguisher  VNI/Label  Router MAC  Nexthop/Overlay GW/ESI
    10.255.2.1:100    9101      00:05:86:e1:03:f0  10.255.2.1
    10.255.2.2:100    9101      00:05:86:d0:a6:f0  10.255.2.2
1234:100:2:12::/64                               0          Created
  Route distinguisher  VNI/Label  Router MAC  Nexthop/Overlay GW/ESI
    10.255.2.1:100    9101      00:05:86:e1:03:f0  10.255.2.1
    10.255.2.2:100    9101      00:05:86:d0:a6:f0  10.255.2.2
1234:100:2:16::/64                               0          Created
  Route distinguisher  VNI/Label  Router MAC  Nexthop/Overlay GW/ESI
    10.255.2.1:100    9101      00:05:86:e1:03:f0  10.255.2.1
    10.255.2.2:100    9101      00:05:86:d0:a6:f0  10.255.2.2

```

show evpn ip-prefix-database extensive

```

user@host> show evpn ip-prefix-database extensive
L3 context: VRF-100

```

IPv4->EVPN Exported Prefixes

Prefix: 100.1.0.0/22
EVPN route status: Created
Change flags: 0x0
Advertisement mode: Direct nexthop
Encapsulation: VXLAN
VNI: 9100
Router MAC: 00:05:86:28:90:f0

Prefix: 100.1.4.0/22
EVPN route status: Created
Change flags: 0x0
Advertisement mode: Direct nexthop
Encapsulation: VXLAN
VNI: 9100
Router MAC: 00:05:86:28:90:f0

Prefix: 100.1.8.0/22
EVPN route status: Created
Change flags: 0x0
Advertisement mode: Direct nexthop
Encapsulation: VXLAN
VNI: 9100
Router MAC: 00:05:86:28:90:f0

Prefix: 100.1.12.0/22
EVPN route status: Created
Change flags: 0x0
Advertisement mode: Direct nexthop
Encapsulation: VXLAN
VNI: 9100
Router MAC: 00:05:86:28:90:f0

Prefix: 100.1.16.0/22
EVPN route status: Created
Change flags: 0x0
Advertisement mode: Direct nexthop
Encapsulation: VXLAN
VNI: 9100
Router MAC: 00:05:86:28:90:f0

IPv6->EVPN Exported Prefixes

Prefix: 1234:100:1::/64
EVPN route status: Created
Change flags: 0x0
Advertisement mode: Direct nexthop
Encapsulation: VXLAN
VNI: 9100
Router MAC: 00:05:86:28:90:f0

Prefix: 1234:100:1:4::/64
EVPN route status: Created
Change flags: 0x0
Advertisement mode: Direct nexthop
Encapsulation: VXLAN
VNI: 9100
Router MAC: 00:05:86:28:90:f0

Prefix: 1234:100:1:8::/64

EVPN route status: Created
Change flags: 0x0
Advertisement mode: Direct nexthop
Encapsulation: VXLAN
VNI: 9100
Router MAC: 00:05:86:28:90:f0

Prefix: 1234:100:1:12::/64
EVPN route status: Created
Change flags: 0x0
Advertisement mode: Direct nexthop
Encapsulation: VXLAN
VNI: 9100
Router MAC: 00:05:86:28:90:f0

Prefix: 1234:100:1:16::/64
EVPN route status: Created
Change flags: 0x0
Advertisement mode: Direct nexthop
Encapsulation: VXLAN
VNI: 9100
Router MAC: 00:05:86:28:90:f0

EVPN->IPv4 Imported Prefixes

Prefix: 100.2.0.0/22, Ethernet tag: 0
IP route status: Created
Change flags: 0x0
Remote advertisements:
Route Distinguisher: 10.255.2.1:100
VNI: 9101
Router MAC: 00:05:86:e1:03:f0
BGP nexthop address: 10.255.2.1
Route Distinguisher: 10.255.2.2:100
VNI: 9101
Router MAC: 00:05:86:d0:a6:f0
BGP nexthop address: 10.255.2.2

Prefix: 100.2.4.0/22, Ethernet tag: 0
IP route status: Created
Change flags: 0x0
Remote advertisements:
Route Distinguisher: 10.255.2.1:100
VNI: 9101
Router MAC: 00:05:86:e1:03:f0
BGP nexthop address: 10.255.2.1
Route Distinguisher: 10.255.2.2:100
VNI: 9101
Router MAC: 00:05:86:d0:a6:f0
BGP nexthop address: 10.255.2.2

Prefix: 100.2.8.0/22, Ethernet tag: 0
IP route status: Created
Change flags: 0x0
Remote advertisements:
Route Distinguisher: 10.255.2.1:100
VNI: 9101
Router MAC: 00:05:86:e1:03:f0
BGP nexthop address: 10.255.2.1
Route Distinguisher: 10.255.2.2:100
VNI: 9101


```
Router MAC: 00:05:86:d0:a6:f0
BGP nexthop address: 10.255.2.2

Prefix: 100.2.12.0/22, Ethernet tag: 0
IP route status: Created
Change flags: 0x0
Remote advertisements:
  Route Distinguisher: 10.255.2.1:100
  VNI: 9101
  Router MAC: 00:05:86:e1:03:f0
  BGP nexthop address: 10.255.2.1
  Route Distinguisher: 10.255.2.2:100
  VNI: 9101
  Router MAC: 00:05:86:d0:a6:f0
  BGP nexthop address: 10.255.2.2

Prefix: 100.2.16.0/22, Ethernet tag: 0
IP route status: Created
Change flags: 0x0
Remote advertisements:
  Route Distinguisher: 10.255.2.1:100
  VNI: 9101
  Router MAC: 00:05:86:e1:03:f0
  BGP nexthop address: 10.255.2.1
  Route Distinguisher: 10.255.2.2:100
  VNI: 9101
  Router MAC: 00:05:86:d0:a6:f0
  BGP nexthop address: 10.255.2.2

EVPN->IPv6 Imported Prefixes

Prefix: 1234:100:2::/64, Ethernet tag: 0
IP route status: Created
Change flags: 0x0
Remote advertisements:
  Route Distinguisher: 10.255.2.1:100
  VNI: 9101
  Router MAC: 00:05:86:e1:03:f0
  BGP nexthop address: 10.255.2.1
  Route Distinguisher: 10.255.2.2:100
  VNI: 9101
  Router MAC: 00:05:86:d0:a6:f0
  BGP nexthop address: 10.255.2.2

Prefix: 1234:100:2:4::/64, Ethernet tag: 0
IP route status: Created
Change flags: 0x0
Remote advertisements:
  Route Distinguisher: 10.255.2.1:100
  VNI: 9101
  Router MAC: 00:05:86:e1:03:f0
  BGP nexthop address: 10.255.2.1
  Route Distinguisher: 10.255.2.2:100
  VNI: 9101
  Router MAC: 00:05:86:d0:a6:f0
  BGP nexthop address: 10.255.2.2

Prefix: 1234:100:2:8::/64, Ethernet tag: 0
IP route status: Created
Change flags: 0x0
Remote advertisements:
```

Route Distinguisher: 10.255.2.1:100
VNI: 9101
Router MAC: 00:05:86:e1:03:f0
BGP nexthop address: 10.255.2.1
Route Distinguisher: 10.255.2.2:100
VNI: 9101
Router MAC: 00:05:86:d0:a6:f0
BGP nexthop address: 10.255.2.2

Prefix: 1234:100:2:12::/64, Ethernet tag: 0
IP route status: Created
Change flags: 0x0
Remote advertisements:
Route Distinguisher: 10.255.2.1:100
VNI: 9101
Router MAC: 00:05:86:e1:03:f0
BGP nexthop address: 10.255.2.1
Route Distinguisher: 10.255.2.2:100
VNI: 9101
Router MAC: 00:05:86:d0:a6:f0
BGP nexthop address: 10.255.2.2

Prefix: 1234:100:2:16::/64, Ethernet tag: 0
IP route status: Created
Change flags: 0x0
Remote advertisements:
Route Distinguisher: 10.255.2.1:100
VNI: 9101
Router MAC: 00:05:86:e1:03:f0
BGP nexthop address: 10.255.2.1
Route Distinguisher: 10.255.2.2:100
VNI: 9101
Router MAC: 00:05:86:d0:a6:f0
BGP nexthop address: 10.255.2.2

show evpn l3-context

Syntax	show evpn l3-context <brief extensive> <routing-instance-name>
Release Information	Statement introduced in Junos OS Release 15.1x53-D30 for QFX10002 switches. Support added for QFX10008 and QFX10016 switches in Junos OS Release 15.1x53-D60.
Description	Display EVPN database information for Layer 3 virtual routing and forwarding (VRF) instances.
Options	<p>none—Display standard EVPN database information for all Layer 3 VRF instances.</p> <p>brief extensive—(Optional) Display the specified level of output.</p> <p>routing-instance-name—(Optional) Display EVPN database information for the specified Layer 3 VRF instance.</p>
Additional Information	
Required Privilege Level	view
Related Documentation	<ul style="list-style-type: none"> • show evpn ip-prefix-database on page 140
List of Sample Output	show evpn l3-context on page 148 show evpn l3-context extensive on page 148
Output Fields	Table 7 on page 147 lists the output fields for the show evpn l3-context command. Output fields are listed in the approximate order in which they appear.

Table 7: show evpn l3-context Output Fields

Field Name	Field Description	Level of Output
L3 context	Name of VRF instance.	All levels
Type	Status of VRF instance	All levels
Fwd	Type of forwarding route	level-of-output none
Encap	Type of encapsulation.	level-of-output none
VNI	Virtual network identifier for VRF instance.	All levels
Router MAC	MAC address of device	All levels
Advertisement mode	Type of forwarding route.	level-of-output Extensive

Table 7: show evpn l3-context Output Fields (*continued*)

Field Name	Field Description	Level of Output
IPv4 source VTEP address	IPv4 source address for the Virtual Extensible LAN (VXLAN) tunnel.	level-of-output Extensive
IPv6 source VTEP address	IPv6 source address for the Virtual Extensible LAN (VXLAN) tunnel.	level-of-output Extensive
IP->EVPN export policy	Name of export routing policy applied to forwarded IP routes.	level-of-output extensive
Flags	Enabled trace flags.	level-of-output extensive
Change flags	Changed trace flags.	level-of-output extensive
Composite nexthop	Status of next-hop route: Enabled or Disabled .	level-of-output extensive
Route distinguisher	Route distinguisher identifier of the VRF instance	level-of-output extensive
Reference count		level-of-output extensive

Sample Output

show evpn l3-context

```

user@DC1_SPINE1_RE> show evpn l3-context
L3 context          Type Fwd  Encap VNI/Label Router MAC/GW intf
VRF-100             Cfg  Sym  VXLAN 9100    00:05:86:28:90:f0
VRF-200             Cfg  Sym  VXLAN 9200    00:05:86:28:90:f0

```

show evpn l3-context extensive

```

user@DC1_SPINE1_RE> show evpn l3-context extensive
L3 context: VRF-100
Type: Configured
Advertisement mode: Direct nexthop, Router MAC: 00:05:86:28:90:f0
Encapsulation: VXLAN, VNI: 9100
IPv4 source VTEP address: 10.255.1.1
IPv6 source VTEP address: abcd::128:102:242:145
IP->EVPN export policy: EVPN-TYPE5-EXPORT-VRF-100
Flags: 0x19 <Configured IRB-MAC New>
Change flags: 0xe <Export-Policy Fwd-Mode Encap>
Composite nexthop support: Enabled
Route Distinguisher: 10.255.1.1:100
Reference count: 21

L3 context: VRF-200
Type: Configured
Advertisement mode: Direct nexthop, Router MAC: 00:05:86:28:90:f0

```

Encapsulation: VXLAN, VNI: 9200
IPv4 source VTEP address: 10.255.1.1
IPv6 source VTEP address: abcd::128:102:242:145
IP->EVPN export policy: EVPN-TYPE5-EXPORT-VRF-200
Flags: 0x19 <Configured IRB-MAC New>
Change flags: 0xe <Export-Policy Fwd-Mode Encap>
Composite nexthop support: Enabled
Route Distinguisher: 10.255.1.1:200
Reference count: 21

show route table

List of Syntax	Syntax on page 150 Syntax (EX Series Switches and QFX Series Switches) on page 150
Syntax	show route table <i>routing-table-name</i> <brief detail extensive terse> <logical-system (all <i>logical-system-name</i>)>
Syntax (EX Series Switches and QFX Series Switches)	show route table <i>routing-table-name</i> <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. Statement introduced in Junos OS Release 14.1X53-D15 for QFX Series switches. Show route table evpn statement introduced in Junos OS Release 15.1X53-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 <i>logical-system-name</i>) —(Optional) Perform this operation on all logical systems or on a particular logical system. <i>routing-table-name</i> —Display route entries for all routing tables whose name begins 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	<ul style="list-style-type: none"> show route summary
List of Sample Output	show route table bgp.l2.vpn on page 161 show route table bgp.l3vpn.0 on page 161 show route table bgp.l3vpn.0 detail on page 161 show route table bgp.rtarget.0 (When Proxy BGP Route Target Filtering Is Configured) on page 162 show route table bgp.evpn.0 on page 163 show route table evpna.evpn.0 on page 163 show route table inet.0 on page 163 show route table inet.3 on page 164 show route table inet6.0 on page 164 show route table inet6.3 on page 164 show route table inetflow detail on page 165 show route table l2circuit.0 on page 165 show route table mpls on page 165 show route table mpls extensive on page 166

[show route table mpls.0 on page 166](#)
[show route table mpls.0 detail \(PTX Series\) on page 166](#)
[show route table mpls.0 extensive \(PTX Series\) on page 167](#)
[show route table mpls.0 \(RSVP Route—Transit LSP\) on page 168](#)
[show route table vpls_1 detail on page 168](#)
[show route table vpn-a on page 168](#)
[show route table vpn-a.mdt.0 on page 169](#)
[show route table VPN-A detail on page 169](#)
[show route table VPN-AB.inet.0 on page 170](#)
[show route table VPN_blue.mvpn-inet6.0 on page 170](#)
[show route table vrf1.mvpn.0 extensive on page 170](#)
[show route table MVPN.mvpn.0 on page 171](#)
[show route table inetflow detail on page 171](#)
[show route table bgp.evpn.0 extensive |no-more \(EVPN\) on page 174](#)

Output Fields [Table 8 on page 151](#) describes the output fields for the **show route table** command. Output fields are listed in the approximate order in which they appear.

Table 8: show route table Output Fields

Field Name	Field Description
<i>routing-table-name</i>	Name of the routing table (for example, inet.0).
Restart complete	<p>All protocols have restarted for this routing table.</p> <p>Restart state:</p> <ul style="list-style-type: none"> • Pending:<i>protocol-name</i>—List of protocols that have not yet completed graceful restart for this routing table. • Complete—All protocols have restarted for this routing table. <p>For example, if the output shows-</p> <ul style="list-style-type: none"> • LDP.inet.0 : 5 routes (4 active, 1 holddown, 0 hidden) Restart Pending: OSPF LDP VPN <p>This indicates that OSPF, LDP, and VPN protocols did not restart for LDP.inet.0 routing table.</p> <ul style="list-style-type: none"> • vpls_1.l2vpn.0: 1 destinations, 1 routes (1 active, 0 holddown, 0 hidden) Restart Complete <p>This indicates that all protocols have restarted for vpls_1.l2vpn.0 routing table.</p>
<i>number destinations</i>	Number of destinations for which there are routes in the routing table.
<i>number routes</i>	<p>Number of routes in the routing table and total number of routes in the following states:</p> <ul style="list-style-type: none"> • 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 8: show route table Output Fields (*continued*)

Field Name	Field Description
<i>route-destination</i> (entry, announced)	<p>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:</p> <ul style="list-style-type: none"> • 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). <ul style="list-style-type: none"> • 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): <ul style="list-style-type: none"> • 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	<p>(Next-to-the-last-hop routing device 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.</p> <ul style="list-style-type: none"> • S=0 route indicates that a packet with an incoming label stack depth of 2 or more exits this routing device 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).
[<i>protocol, preference</i>]	<p>Protocol from which the route was learned and the preference value for the route.</p> <ul style="list-style-type: none"> • +—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. <p>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.</p>

Table 8: 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 9 on page 156 .
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: <ul style="list-style-type: none"> • 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 routing device is performing unequal-cost load balancing. This information is available when you enable BGP multipath load balancing.
Label-switched-path <i>lsp-path-name</i>	Name of the LSP used to reach the next hop.
Label operation	MPLS label and operation occurring at this routing device. 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 routing device 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 10 on page 158 .

Table 8: show route table Output Fields (*continued*)

Field Name	Field Description
Local AS	AS number of the local routing device.
Age	How long the route has been known.
AIGP	Accumulated interior gateway protocol (AIGP) BGP attribute.
Metric	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	<p>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 KRT for installing the route into the Packet Forwarding Engine, to a resolve tree, a L2 VC, or even a VPN. For example, <i>n-Resolve inet</i> indicates that the specified route is used for route resolution for next hops found in the routing table.</p> <ul style="list-style-type: none"> <i>n</i>—An index used by Juniper Networks customer support only.
AS path	<p>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:</p> <ul style="list-style-type: none"> I—IGP. E—EGP. Recorded—The AS path is recorded by the sample process (sampled). ?—Incomplete; typically, the AS path was aggregated. <p>When AS path numbers are included in the route, the format is as follows:</p> <ul style="list-style-type: none"> []—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. <p>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.</p>

Table 8: show route table Output Fields (*continued*)

Field Name	Field Description
validation-state	<p>(BGP-learned routes) Validation status of the route:</p> <ul style="list-style-type: none"> • 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	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, 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, 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 11 on page 160 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 routing device uses this first label when sending traffic toward the advertising PE routing device.
status vector	Layer 2 VPN and VPLS network layer reachability information (NLRI).

Table 8: 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 might be displayed for a route. Neither of these flags is displayed at the same time as the Stale (ordinary GR stale) flag.
Accepted LongLivedStaleImport	<p>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 might be displayed for a route. Neither of these flags is displayed at the same time as the Stale (ordinary GR stale) flag.</p> <p>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</p>
ImportAccepted LongLivedStaleImport	<p>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</p> <p>The LongLivedStaleImport flag indicates that the route was marked LLGR-stale when it was received from a peer, or by import policy.</p>
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 9 on page 156](#) describes all possible values for the Next-hop Types output field.

Table 9: 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 9: 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 (mcrtr)	Regular multicast next hop.
Router	<p>A specific node or set of nodes to which the routing device forwards packets that match the route prefix.</p> <p>To qualify as next-hop type router, the route must meet the following criteria:</p> <ul style="list-style-type: none"> • 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 10 on page 158 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 10: 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 EGBP path is available.

Table 10: 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: <ul style="list-style-type: none"> • 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 11 on page 160 describes the possible values for the Communities output field.

Table 11: Communities Output Field Values

Value	Description
<i>area-number</i>	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.
<i>link-bandwidth-number</i>	Link-bandwidth number: from 0 through 4,294,967,295 (bytes per second).
<i>local AS number</i>	Local AS number: from 1 through 65,535.
<i>options</i>	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.
<i>ospf-route-type</i>	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.l2vpn

```
user@host> show route table bgp.l2vpn
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.0
```

```

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 inet.3

```

user@host> show route table inet.3
inet.3: 5 destinations, 5 routes (5 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

22.0.0.5/32      *[LDP/9] 00:25:43, metric 10, tag 200
                  to 1.2.94.2 via lt-1/2/0.49
                  > to 1.2.3.2 via lt-1/2/0.23

```

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: 11 destinations, 11 routes (11 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

0          *[MPLS/0] 00:45:09, metric 1
           Receive
1          *[MPLS/0] 00:45:09, metric 1
           Receive
2          *[MPLS/0] 00:45:09, metric 1
           Receive
100000     *[L2VPN/7] 00:43:04
           > via so-0/1/0.1, Pop
100001     *[L2VPN/7] 00:43:03
           > via so-0/1/0.2, Pop      Offset: 4
100002     *[LDP/9] 00:43:22, metric 1
           via so-0/1/2.0, Pop
           > via so-0/1/3.0, Pop
100002(S=0) *[LDP/9] 00:43:22, metric 1
           via so-0/1/2.0, Pop
           > via so-0/1/3.0, Pop
100003     *[LDP/9] 00:43:22, metric 1
           > via so-0/1/2.0, Swap 100002
           via so-0/1/3.0, Swap 100002
100004     *[LDP/9] 00:43:16, metric 1
           via so-0/1/2.0, Swap 100049
           > via so-0/1/3.0, Swap 100049
so-0/1/0.1 *[L2VPN/7] 00:43:04
           > via so-0/1/2.0, Push 100001, Push 100049(top)
           via so-0/1/3.0, Push 100001, Push 100049(top)
so-0/1/0.2 *[L2VPN/7] 00:43:03
           via so-0/1/2.0, Push 100000, Push 100049(top) Offset: -4
           > via so-0/1/3.0, Push 100000, Push 100049(top) Offset: -4

```

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
Forwarding nexthops: 1
Node path count: 1
Nexthop: 3.0.0.1 via ge-0/0/1.0

```

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

In the sample output, the 1 in [RSVP/7/1] indicates the secondary preference value. The secondary preference value becomes significant when multiple RSVP LSPs of different types are signaled to the destination. The possible values of RSVP secondary preferences are:

1—Normal Point-to-Point RSVP-TE LSP

2—Point-to-Multipoint (P2MP) RSVP-TE LSP

3—Dynamic RSVP-TE 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.12vpn.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.13vpn.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 MVPN.mvpn.0

Starting in Junos OS Release 15.1, multicast routes on the locally originated type 7 customer multicast routes are added exclusively by PIM. The functionality of the BGP-MVPN service (which, internally, depends on contributions of state from both the MVPN and PIM protocol components of Junos OS) remains unchanged. MVPN, however, no longer appears as the originator of the locally advertised route. Routes advertised by remote PEs are, as usual, always learned locally from their respective [BGP/...] protocol.

```

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

7:10.255.2.202:65535:65000:128:::192.168.90.2:128:ffff::1/432
    *[PIM/70] 00:02:37, metric2 1
    Indirect
5:100:32:192.168.1.9:32:239.1.1.1/240
    *[PIM/105] 01:51:21
    Multicast (IPv4)
7:100:1:100.32.192.168.5:32:237.1.1.1/240
    *[PIM/105] 01:51:21
    Multicast (IPv4)

```

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.13vpn.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.13vpn.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

```

show route table bgp.evpn.0 extensive |no-more (EVPN)

```

show route table bgp.evpn.0 extensive | no-more
bgp.evpn.0: 6 destinations, 6 routes (6 active, 0 holddown, 0 hidden)
2:1000:10::100::00:aa:aa:aa:aa:aa/304 (1 entry, 0 announced)
    *BGP
        Preference: 170/-101
        Route Distinguisher: 1000:10
        Next hop type: Indirect
        Address: 0x9420fd0
        Next-hop reference count: 12
        Source: 1.2.3.4
        Protocol next hop: 1.2.3.4
        Indirect next hop: 0x2 no-forward INH Session ID: 0x0
        State: Local AS: 17 Peer AS:17 Age:21:12 Metric2:1 Validation State:
unverified
        Task: BGP_17.1.2.3.4+50756
        AS path: I
        Communities: target:1111:8388708 encapsulation0:0:0:0:3
        Import Accepted
        Route Label: 100
        ESI: 00:00:00:00:00:00:00:00:00:00

```

```

Localpref: 100
Router ID: 1.2.3.4
Secondary Tables: default-switch.evpn.0
Indirect next hops: 1
    Protocol next hop: 1.2.3.4 Metric: 1
    Indirect next hop: 0x2 no-forward INH Session ID: 0x0
    Indirect path forwarding next hops: 1
        Next hop type: Router
        Next hop: 10.10.10.1 via xe-0/0/1.0
        Session Id: 0x2
    1.2.3.4/32 Originating RIB: inet.0
        Metric: 1                      Node path count: 1
        Forwarding nexthops: 2
        Nexthop: 10.92.78.102 via em0.0

2:1000:10::200::00:bb:bb:bb:bb:bb/304 (1 entry, 0 announced)
  *BGP   Preference: 170/-101
        Route Distinguisher: 1000:10
        Next hop type: Indirect
        Address: 0x9420fd0
        Next-hop reference count: 12
        Source: 1.2.3.4
        Protocol next hop: 1.2.3.4
        Indirect next hop: 0x2 no-forward INH Session ID: 0x0
        State: Local AS:17 Peer AS:17 Age:19:43 Metric2:1 Validation
State:unverified
        Task: BGP_17.1.2.3.4+50756
        AS path: I
        Communities: target:2222:22 encapsulation0:0:0:0:3
        Import Accepted
        Route Label: 200
        ESI: 00:00:00:00:00:00:00:00:00:00
        Localpref: 100
        Router ID: 1.2.3.4
        Secondary Tables: default-switch.evpn.0
        Indirect next hops: 1
            Protocol next hop: 1.2.3.4 Metric: 1
            Indirect next hop: 0x2 no-forward INH Session ID: 0x0
            Indirect path forwarding next hops: 1
                Next hop type: Router
                Next hop: 10.10.10.1 via xe-0/0/1.0
                Session Id: 0x2
            1.2.3.4/32 Originating RIB: inet.0
                Metric: 1                      Node path count: 1
                Forwarding nexthops: 2
                Nexthop: 10.92.78.102 via em0.0

2:1000:10::300::00:cc:cc:cc:cc:cc/304 (1 entry, 0 announced)
  *BGP   Preference: 170/-101
        Route Distinguisher: 1000:10
        Next hop type: Indirect
        Address: 0x9420fd0
        Next-hop reference count: 12
        Source: 1.2.3.4
        Protocol next hop: 1.2.3.4
        Indirect next hop: 0x2 no-forward INH Session ID: 0x0
        State: Local AS:17 Peer AS:17 Age:17:21 Metric2:1 Validation State:
unverified Task: BGP 17,1,2,3,4+50756
        AS path: I
        Communities: target:3333:33 encapsulation0:0:0:0:3
        Import Accepted

```

```

Route Label: 300
ESI: 00:00:00:00:00:00:00:00:00
Localpref: 100
Router ID: 1.2.3.4
Secondary Tables: default-switch.evpn.0
Indirect next hops: 1
    Protocol next hop: 1.2.3.4 Metric: 1
    Indirect next hop: 0x2 no-forward INH Session ID: 0x0
    Indirect path forwarding next hops: 1
        Next hop type: Router
        Next hop: 10.10.10.1 via xe-0/0/1.0
        Session Id: 0x2
    1.2.3.4/32 Originating RIB: inet.0
        Metric: 1                      Node path count: 1
        Forwarding nexthops: 2
        Nexthop: 10.92.78.102 via em0.0

3:1000:10::100::1.2.3.4/304 (1 entry, 0 announced)
*BGP   Preference: 170/-101
      Route Distinguisher: 1000:10
      PMSI: Flags 0x0: Label 100: Type INGRESS-REPLICATION 1.2.3.4
      Next hop type: Indirect
      Address: 0x9420fd0
      Next-hop reference count: 12
      Source: 1.2.3.4
      Protocol next hop: 1.2.3.4
      Indirect next hop: 0x2 no-forward INH Session ID: 0x0
      State: Local AS:17 Peer AS:17 Age:37:01 Metric2:1 Validation State:
unverified Task: BGP 17.1.2.3.4+50756
      AS path: I
      Communities: target:1111:8388708 encapsulation0:0:0:0:3
      Import Accepted
      Localpref: 100
      Router ID: 1.2.3.4
      Secondary Tables: default-switch.evpn.0
      Indirect next hops: 1
          Protocol next hop: 1.2.3.4 Metric: 1
          Indirect next hop: 0x2 no-forward INH Session ID: 0x0
          Indirect path forwarding next hops: 1
              Next hop type: Router
              Next hop: 10.10.10.1 via xe-0/0/1.0
              Session Id: 0x2
          1.2.3.4/32 Originating RIB: inet.0
              Metric: 1                      Node path count: 1
              Forwarding nexthops: 2
              Nexthop: 10.92.78.102 via em0.0

3:1000:10::200::1.2.3.4/304 (1 entry, 0 announced)
*BGP   Preference: 170/-101
      Route Distinguisher: 1000:10
      PMSI: Flags 0x0: Label 200: Type INGRESS-REPLICATION 1.2.3.4
      Next hop type: Indirect
      Address: 0x9420fd0
      Next-hop reference count: 12
      Source: 1.2.3.4
      Protocol next hop: 1.2.3.4
      Indirect next hop: 0x2 no-forward INH Session ID: 0x0
      State: Local AS: 17 Peer AS: 17 Age:35:22 Metric2:1 Validation
State:unverified Task: BGP 17.1.2.3.4+50756
      AS path:I Communities: target:2222:22 encapsulation):0:0:0:0:3

```



```

Import Accepted
  Localpref: 100
  Router ID: 1.2.3.4
  Secondary Tables: default-switch.evpn.0
  Indirect next hops: 1
    Protocol next hop: 1.2.3.4 Metric: 1
    Indirect next hop: 0x2 no-forward INH Session ID: 0x0
    Indirect path forwarding next hops: 1
      Next hop type: Router
      Next hop: 10.10.10.1 via xe-0/0/1.0
      Session Id: 0x2
    1.2.3.4/32 Originating RIB: inet.0
      Metric: 1
      Forwarding nexthops: 2
      Nexthop: 10.92.78.102 via em0.0
      Node path count: 1

3:1000:10::300::1.2.3.4/304 (1 entry, 0 announced)
  *BGP Preference: 170/-101
    Route Distinguisher: 1000:10
    PMSI: Flags 0x0: Label 300: Type INGRESS-REPLICATION 1.2.3.4
    Next hop type: Indirect
    Address: 0x9420fd0
    Next-hop reference count: 12
    Source: 1.2.3.4
    Protocol next hop: 1.2.3.4
    Indirect next hop: 0x2 no-forward INH Session ID: 0x0
    State: Local AS: 17 Peer AS: 17 Age 35:22 Metric2:1 Validation State:
unverified Task: BGP 17.1.2.3.4+5075
  6 AS path: I Communities: target:3333:33 encapsulation0:0:0:0:3
Import Accepted Localpref:100
  Router ID: 1.2.3.4
  Secondary Tables: default-switch.evpn.0
  Indirect next hops: 1
    Protocol next hop: 1.2.3.4 Metric: 1
    Indirect next hop: 0x2 no-forward INH Session ID: 0x0
    Indirect path forwarding next hops: 1
      Next hop type: Router
      Next hop: 10.10.10.1 via xe-0/0/1.0
      Session Id: 0x2
    1.2.3.4/32 Originating RIB: inet.0
      Metric: 1
      Forwarding nexthops: 2
      Nexthop: 10.92.78.102 via em0.0
      Node path count: 1

```

