

# Junos® OS

---

## OVSDB and VXLAN User Guide for MX Series Routers and EX9200 Switches

Published  
2021-12-14

Juniper Networks, Inc.  
1133 Innovation Way  
Sunnyvale, California 94089  
USA  
408-745-2000  
[www.juniper.net](http://www.juniper.net)

Juniper Networks, the Juniper Networks logo, Juniper, and Junos are registered trademarks of Juniper Networks, Inc. in the United States and other countries. All other trademarks, service marks, registered marks, 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 OVSDb and VXLAN User Guide for MX Series Routers and EX9200 Switches*  
Copyright © 2021 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 <https://support.juniper.net/support/eula/>. By downloading, installing or using such software, you agree to the terms and conditions of that EULA.

# Table of Contents

## 1

[About This Guide | viii](#)

## [OVSDB and VXLAN](#)

[Understanding OVSDB | 2](#)

[OVSDB Support on Juniper Networks Devices | 2](#)

[Understanding the Junos OS Implementation of OVSDB and VXLAN in a VMware NSX for vSphere Environment | 3](#)

[Understanding the OVSDB Protocol Running on Juniper Networks Devices | 6](#)

[Understanding How to Set Up OVSDB Connections on a Juniper Networks Device | 7](#)

[Understanding How Layer 2 BUM and Layer 3 Routed Multicast Traffic Are Handled with OVSDB | 8](#)

[Understanding How to Manually Configure OVSDB-Managed VXLANs | 10](#)

[OVSDB Schema for Physical Devices | 13](#)

### [Configuring OVSDB and VXLAN | 17](#)

[OVSDB and VXLAN Configuration Workflows for VMware NSX Environment | 17](#)

[OVSDB and VXLAN Configuration Workflow for QFX Series Switches | 18](#)

[OVSDB and VXLAN Configuration Workflow for MX Series Routers and EX9200 Switches | 20](#)

[Installing OVSDB on Juniper Networks Devices | 22](#)

[Creating and Installing an SSL Key and Certificate on a Juniper Networks Device for Connection with SDN Controllers | 23](#)

[Setting Up the OVSDB Protocol on Juniper Networks Devices that Support Manual Configuration of VXLANs | 24](#)

[Configuring OVSDB-Managed VXLANs | 26](#)

[VMware NSX Configuration for Juniper Networks Devices Functioning as Virtual Tunnel Endpoints | 29](#)

[Creating a Gateway | 30](#)

[Creating a Gateway Service | 30](#)

[Creating a Logical Switch Port | 31](#)

[Example: Setting Up Inter-VXLAN Unicast Routing and OVSDB Connections in a Data Center | 33](#)

Requirements	33
Overview and Topology	34
Configuration	38
Verification	45

#### Example: Setting Up Inter-VXLAN Unicast and Multicast Routing and OVSDDB Connections in a Data Center | 48

Requirements	49
Overview and Topology	50
Configuration	54
Verification	63

#### Example: Configuring VXLAN to VPLS Stitching with OVSDDB | 66

Requirements	66
Overview	67
Configuration	68
Verification	78

#### Example: Configuring Inter-VXLAN Traffic Routing from One Bridge Domain to Another Using an MX Series Router as a Layer 3 Gateway | 95

Requirements	95
Overview and Topology	95
Configuration	98
Verification	102

#### Example: Passing Traffic Between Data Centers with DCI in an OVSDDB-Managed Network with MX Series Routers | 105

Requirements	105
Overview and Topology	106
Configuration	108
Verification	113

### OVSDDB Configuration Statements | 118

bridge-domains	118
controller (OVSDDB)	121
inactivity-probe-duration	123
ingress-node-replication	124

interfaces (OVSDB) | 126

maximum-backoff-duration | 127

ovsdb | 129

ovsdb-managed | 130

port (OVSDB) | 132

protocol (OVSDB) | 133

traceoptions (OVSDB) | 135

routing-instances (Multiple Routing Entities) | 137

interface-mode | 140

unit | 142

vlan-id-list (Interface in Bridge Domain) | 153

## **OVSDB Monitoring Commands | 155**

show bridge domain | 155

show ovsdb controller | 158

show ovsdb interface | 161

show ovsdb logical-switch | 164

show ovsdb mac | 167

show ovsdb statistics interface | 173

show ovsdb virtual-tunnel-end-point | 175

show vpls mac-table | 178

Verifying That a Logical Switch and Corresponding Junos OS OVSDB-Managed VXLAN Are Working Properly | 186

## **Troubleshooting OVSDB | 188**

Troubleshooting a Nonoperational Logical Switch and Corresponding Junos OS OVSDB-Managed VXLAN | 188

## **VXLAN (Without a Controller)**

Using VXLAN Without a Controller | 193

## Understanding VXLANs | 193

VXLAN Benefits | 194

How Does VXLAN Work? | 195

VXLAN Implementation Methods | 196

Using QFX5100, QFX5110, QFX5120, QFX5200, QFX5210, EX4300-48MP, and EX4600 Switches with VXLANs | 196

Changing the UDP Port on QFX5100, QFX5110, QFX5200, QFX5210, and EX4600 Switches | 197

Controlling Transit Multicast Traffic on QFX5100, QFX5110, QFX5200, QFX5210, and EX4600 Switches | 198

Using an MX Series Router, EX9200 Switch, or QFX10000 Switch as a VTEP | 198

Manual VXLANs Require PIM | 199

Load Balancing VXLAN Traffic | 199

VLAN IDs for VXLANs | 200

Enabling QFX5120 Switches to Tunnel Traffic on Core-Facing Layer 3 Tagged and IRB Interfaces | 200

Using ping and traceroute with a VXLAN | 201

Supported VXLAN Standards | 201

## PIM NSR and Unified ISSU Support for VXLAN Overview | 202

### Example: Manually Configuring VXLANs on MX Series Routers | 203

Requirements | 204

Overview | 204

Configuring VXLAN on MX Series Routers | 206

Verification | 215

## VXLAN Configuration Statements | 219

encapsulate-inner-vlan | 219

multicast-group | 221

ovsdb-managed | 222

unreachable-vtep-aging-timer | 223

vni | 225

vxlan | 226

## VXLAN Monitoring Commands | 229

Monitoring a Remote VTEP Interface | 229

show bridge mac-table | 231

show vpls mac-table | 239

Verifying VXLAN Reachability | 246

Verifying That a Local VXLAN VTEP Is Configured Correctly | 248

Verifying MAC Learning from a Remote VTEP | 249

Understanding Overlay ping and traceroute Packet Support | 250

Example: Troubleshooting a VXLAN Overlay Network By Using Overlay Ping and Traceroute for MX Series Routers | 254

Requirements | 255

Overview and Topology | 255

Configuration | 259

Verification | 259

ping overlay | 268

traceroute overlay | 275

# About This Guide

The Open vSwitch Database (OVSDB) management protocol provides a control plane through which MX Series routers and EX9200 switches in the physical underlay can exchange control and statistical information with VMware NSX controllers in the virtual overlay. Virtual Extensible LAN (VXLAN) provides a data plane through which Layer 2 data packets can be tunneled over a Layer 3 transport network. Use this guide to learn how OVSDB-VXLAN is implemented on MX Series routers and EX9200 switches and to configure, monitor, and troubleshoot OVSDB-VXLAN on these Juniper Networks devices.

You can also use this guide to learn about and configure manual VXLAN, which enables you to manually create VXLANs on MX Series routers instead of using a controller. If you use this approach, you must also configure Protocol Independent Multicast (PIM), which enables two MX Series routers to create VXLAN tunnels between themselves.



# 1

PART

## OVSDB and VXLAN

---

[Understanding OVSDB | 2](#)

[Configuring OVSDB and VXLAN | 17](#)

[OVSDB Configuration Statements | 118](#)

[OVSDB Monitoring Commands | 155](#)

[Troubleshooting OVSDB | 188](#)

---

# Understanding OVSDB

## IN THIS CHAPTER

- [OVSDB Support on Juniper Networks Devices | 2](#)
- [Understanding the Junos OS Implementation of OVSDB and VXLAN in a VMware NSX for vSphere Environment | 3](#)
- [Understanding the OVSDB Protocol Running on Juniper Networks Devices | 6](#)
- [Understanding How to Set Up OVSDB Connections on a Juniper Networks Device | 7](#)
- [Understanding How Layer 2 BUM and Layer 3 Routed Multicast Traffic Are Handled with OVSDB | 8](#)
- [Understanding How to Manually Configure OVSDB-Managed VXLANs | 10](#)
- [OVSDB Schema for Physical Devices | 13](#)

## OVSDB Support on Juniper Networks Devices

The following Juniper Networks devices support the *Open vSwitch Database (OVSDB)* management protocol:

- EX9200 Line of Ethernet Switches
- MX80, MX104, MX240, MX480, MX960, MX2010, and MX2020 Universal Routing Platforms
- QFX Series Switches

Starting with Junos OS Release 14.1X53-D30 for QFX5100 switches, 15.1X53-D20 for QFX10002 switches, 15.1X53-D30 for QFX10008 switches, 15.1X53-D60 for QFX10016 switches, 15.1X53-D210 for QFX5110 and QFX5200 switches, 16.1R1 for EX9200 switches and MX routers, and 18.1R1 for QFX5210 switches, the OVSDB software (jsdn) package is included in the Junos OS software (jinstall) package. As a result, if you have one of the listed releases or a later release, you no longer need to install the separate jsdn package on the Juniper Networks devices.

**Release History Table**

Release	Description
14.1X53-D30	Starting with Junos OS Release 14.1X53-D30 for QFX5100 switches, 15.1X53-D20 for QFX10002 switches, 15.1X53-D30 for QFX10008 switches, 15.1X53-D60 for QFX10016 switches, 15.1X53-D210 for QFX5110 and QFX5200 switches, 16.1R1 for EX9200 switches and MX routers, and 18.1R1 for QFX5210 switches, the OVSDb software (jsdn) package is included in the Junos OS software (jinstall) package. As a result, if you have one of the listed releases or a later release, you no longer need to Install the separate jsdn package on the Juniper Networks devices.

## Understanding the Junos OS Implementation of OVSDb and VXLAN in a VMware NSX for vSphere Environment

Some Juniper Networks devices support Virtual Extensible LAN (VXLAN) and the Open vSwitch Database (OVSDb) management protocol. (See ["OVSDb Support on Juniper Networks Devices" on page 2.](#)) Support for VXLAN and OVSDb enables the Juniper Networks devices in a physical network to be integrated into a virtual network.

The implementation of VXLAN and OVSDb on Juniper Networks devices is supported in a VMware NSX for NSX for vSphere environment for the data center. [Table 1 on page 3](#) outlines the components that compose this environment and products that are typically deployed for each component.

**Table 1: NSX for vSphere Components and Related Products**

Component	Products
Cloud management platform (CMP)	CloudStack OpenStack Custom CMP
Network virtualization platform	NSX for vSphere

**Table 1: NSX for vSphere Components and Related Products** *(Continued)*

Component	Products
Hypervisor	Kernel-based Virtual Machine (KVM) Red Hat VMware ESXi Xen <b>NOTE:</b> Juniper Networks supports only KVM and ESXi.
Virtual switch	Open vSwitch (OVS) NSX vSwitch
SDN controller	NSX for vSphere controller
Overlay protocol	VXLAN
Media access control (MAC) learning protocol	OVSDB

Figure 1 on page 5 shows a high-level view of the NSX for vSphere platform architecture, while Figure 2 on page 5 provides a more detailed representation of the components in the virtual and physical networks.

Figure 1: High-Level View of NSX for vSphere Architecture

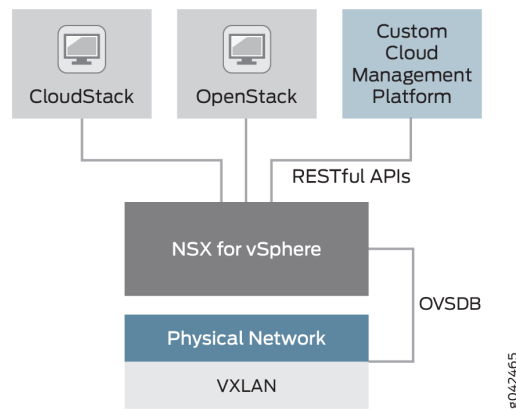
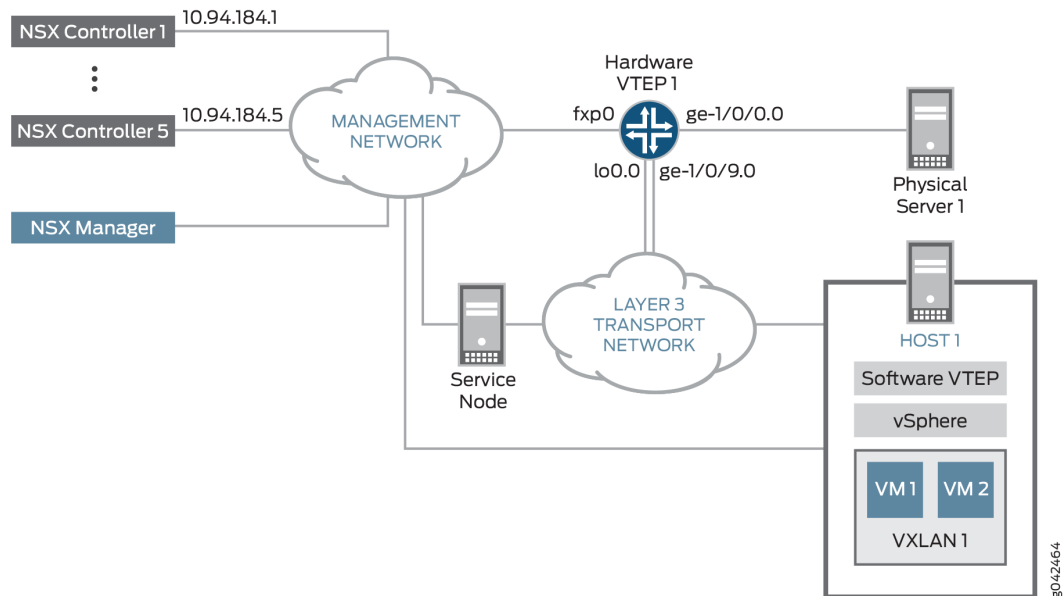


Figure 2: Integration of Juniper Networks Device into NSX for vSphere Environment



In the data center topology shown in Figure 2 on page 5, the physical and virtual servers need to communicate. To facilitate this communication, a Juniper Networks device that supports VXLAN is strategically deployed so that it serves as a *gateway*, which is also known as a hardware *virtual tunnel*

*endpoint (VTEP)*, at the edge of the physical network. Working in conjunction with the software VTEP, which is deployed at the edge of the virtual network, the hardware VTEP encapsulates packets from resources on Physical Server 1 with a VXLAN header, and after the packets traverse the Layer 3 transport network, the software VTEP removes the VXLAN header from the packets and forwards the packets to the appropriate virtual machines (VMs). In essence, the encapsulation and de-encapsulation of packets by the hardware and software VTEPs enable the components in the physical and virtual networks to coexist without one needing to understand the workings of the other.

The same Juniper Networks device that acts as a hardware VTEP in [Figure 2 on page 5](#) implements OVSDB, which enables this device to learn the MAC addresses of Physical Server 1 and other physical servers, and publish the addresses in the OVSDB schema, which was defined for physical devices. In the virtual network, one or more NSX controllers collect the MAC addresses of Host 1 and other virtual servers, and publish the addresses in the OVSDB schema. Using the OVSDB schema, components in the physical and virtual networks can exchange MAC addresses, as well as statistical information, enabling the components to learn about and reach each other in their respective networks.

## RELATED DOCUMENTATION

[Understanding the OVSDB Protocol Running on Juniper Networks Devices | 6](#)

[OVSDB Schema for Physical Devices | 13](#)

## Understanding the OVSDB Protocol Running on Juniper Networks Devices

The Juniper Networks Junos OS implementation of the Open vSwitch Database (OVSDB) management protocol provides a means through which Juniper Networks devices that support OVSDB can communicate with software-defined networking (SDN) controllers. Juniper Networks devices exchange control and statistical information with the SDN controllers, thereby enabling *virtual machine (VM)* traffic from the entities in a virtualized network to be forwarded to entities in a physical network and vice versa.

The Junos OS implementation of OVSDB includes an OVSDB server and an OVSDB client, both of which run on each Juniper Networks device that supports OVSDB.

The OVSDB server on a Juniper Networks device can communicate with an OVSDB client on an SDN controller. To establish a connection between a Juniper Networks device and an SDN controller, you must specify information about the SDN controller (IP address) and the connection (port over which the connection occurs and the communication protocol to be used) on each Juniper Networks device. After the configuration is successfully committed, the connection is established between the management

port of the Juniper Networks device and the SDN controller port that you specify in the Junos OS configuration.

The OVSDB server stores and maintains an OVSDB database schema, which is defined for physical devices. This schema contains control and statistical information provided by the OVSDB client on the Juniper Networks devices and on SDN controllers. This information is stored in various tables in the schema. The OVSDB client monitors the schema for additions, deletions, and modifications to this information, and the information is used for various purposes, such as learning the media access control (MAC) addresses of virtual hosts and physical servers.

The schema provides a means through which the Juniper Networks devices and the SDN controllers can exchange information. For example, the Juniper Networks devices capture MAC routes to entities in the physical network and push this information to a table in the schema so that SDN controllers with connections to these Juniper Networks devices can access the MAC routes. Conversely, SDN controllers capture MAC routes to entities in the virtualized network and push this information to a table in the schema so that Juniper Networks devices with connections to the SDN controllers can access the MAC routes.

Some of the OVSDB table names include the words *local* or *remote*, for example, *unicast MACs local table* and *unicast MACs remote table*. Information in *local* tables is learned by a Juniper Networks device that functions as a hardware *virtual tunnel endpoint (VTEP)*, while information in *remote* tables is learned from other software or hardware VTEPs.

## Understanding How to Set Up OVSDB Connections on a Juniper Networks Device

The Juniper Networks Junos OS implementation of the Open vSwitch Database (OVSDB) management protocol provides a means through which Juniper Networks devices that support OVSDB can communicate with software-defined networking (SDN) controllers. A Juniper Networks device exchanges control and statistical data with each SDN controller to which it is connected.

You can connect a Juniper Networks device to more than one SDN controller for redundancy.

In a VMware NSX environment, one cluster of NSX controllers typically includes three or five controllers. To implement the OVSDB management protocol on a Juniper Networks device, you must explicitly configure a connection to one SDN controller, using the Junos OS CLI. If the SDN controller to which you explicitly configure a connection is in a cluster, the controller pushes information about other controllers in the same cluster to the device, and the device establishes connections with the other controllers. However, you can also explicitly configure connections with the other controllers in the cluster, using the Junos OS CLI.

To implement the OVSDB management protocol on a Juniper Networks device in a Contrail environment, you must configure a connection to a Contrail controller, using the Junos OS CLI.

Connections to all SDN controllers are made on the management interface of the Juniper Networks device. To set up a connection between a Juniper Networks device and an SDN controller, you need to configure the following parameters on the Juniper Networks device:

- IP address of the SDN controller.
- The protocol that secures the connection. *Secure Sockets Layer (SSL)* is the supported protocol.

**NOTE:** The SSL connection requires a private key and certificates, which must be stored in the `/var/db/certs` directory of the Juniper Networks device. See ["Creating and Installing an SSL Key and Certificate on a Juniper Networks Device for Connection with SDN Controllers" on page 23](#).

- Number of the port over which the connection is made. The port number of the default port is 6632.

Optionally, you can configure the following connection timers on the Juniper Networks device:

- Inactivity probe duration—The maximum amount of time, in milliseconds, that the connection can be inactive before an inactivity probe is sent. The default value is 0 milliseconds, which means that an inactivity probe is never sent.
- Maximum backoff duration—If an attempt to connect to an SDN controller fails, the maximum amount of time, in milliseconds, before the device can make the next attempt. The default value is 1000 milliseconds.

## RELATED DOCUMENTATION

[Setting Up the OVSDB Protocol on Juniper Networks Devices that Support Manual Configuration of VXLANs | 24](#)

*Setting Up OVSDB on Juniper Networks Devices That Support the Dynamic Configuration of VXLANs*

## Understanding How Layer 2 BUM and Layer 3 Routed Multicast Traffic Are Handled with OVSDB

The Juniper Networks Junos OS implementation of the Open vSwitch Database (OVSDB) management protocol provides a means through which software-defined networking (SDN) controllers and Juniper Networks devices that support OVSDB can communicate.



This topic explains how a Juniper Networks device with Virtual Extensible LAN (VXLAN) and OVSDB management protocol capabilities handles the following types of traffic:

- (This scenario applies to all Juniper Networks devices that support VXLAN and OVSDB.) Layer 2 *broadcast, unknown unicast, and multicast (BUM)* traffic that originates in an OVSDB-managed VXLAN and is forwarded to interfaces within the same VXLAN.

**NOTE:** You must explicitly configure the replication of unknown unicast traffic in a Contrail environment.

- (This scenario applies only to Juniper Networks devices that can function as a Layer 3 VXLAN gateway in an OVSDB-VXLAN environment.) Layer 3 multicast traffic that is received by an *integrated routing and bridging (IRB)* interface in an OVSDB-managed VXLAN and is forwarded to interfaces in another OVSDB-managed VXLAN.

By default, Layer 2 BUM traffic that originates in an OVSDB-managed VXLAN is handled by one or more software *virtual tunnel endpoints (VTEPs)*, service nodes, or top-of-rack service nodes (TSNs) in the same VXLAN. (In this topic, software VTEPs, service nodes, and TSNs are known collectively as *replicators*.) The table for remote multicast media access control (MAC) addresses in the OVSDB schema for physical devices contains only one entry that has the keyword *unknown-dst* as the MAC string and a list of replicators.

Given the previously described table entry, Layer 2 BUM traffic received on an interface in the OVSDB-managed VXLAN is forwarded to one of the replicators. The replicator to which a BUM packet is forwarded is determined by the Juniper Networks device on which the OVSDB-managed VXLAN is configured. On receiving the BUM packet, the entity replicates the packet and forwards the replicas to all interfaces within the VXLAN.

Instead of using replicators, you can optionally enable ingress node replication to handle Layer 2 BUM traffic on Juniper Networks devices that support OVSDB.

**NOTE:** Ingress node replication is supported on all Juniper Networks devices that support OVSDB except the QFX Series switches.

With ingress node replication enabled, on receiving a Layer 2 BUM packet on an interface in an OVSDB-managed VXLAN, the Juniper Networks device replicates the packet and then forwards the replicas to all software VTEPs included in the unicast MACs remote table in the OVSDB schema. The software VTEPs then forward the replicas to all *virtual machines (VMs)*, except service VMs, or nodes, on the same host.

**NOTE:** When Juniper Networks devices replicate Layer 2 BUM packets to a large number of remote software VTEPs, the performance of the Juniper Networks devices can be impacted.

On IRB interfaces that forward Layer 3 multicast traffic from one OVSDB-managed VXLAN to another, ingress node replication is automatically implemented. With ingress node replication, the Juniper Networks device replicates a Layer 3 multicast packet and then the IRB interface forwards the replicas to all hardware and software VTEPs, but not to service nodes, in the other OVSDB-managed VXLAN. For the routing of Layer 3 multicast traffic from one OVSDB-managed VXLAN to another, ingress node replication is the only option and does not need to be configured.

## RELATED DOCUMENTATION

[Configuring OVSDB-Managed VXLANs | 26](#)

*Understanding BFD in a VMware NSX Environment with OVSDB and VXLAN*

# Understanding How to Manually Configure OVSDB-Managed VXLANs

## IN THIS SECTION

- [Understanding How to Manually Configure OVSDB-Managed VXLANs On Juniper Networks Devices | 11](#)

The Juniper Networks Junos operating system (Junos OS) implementation of the Open vSwitch Database (OVSDB) management protocol provides a means through which VMware NSX controllers and Juniper Networks devices that support OVSDB can communicate.

In a Junos OS environment, the concept of an OVSDB-managed Layer 2 broadcast domain in which data flows are limited to that domain is known as a *VXLAN*. In an NSX environment, the same concept is known as a *logical switch*. Understanding the different terminology in turn enables you to better understand the configuration tasks required for setting up OVSDB-managed VXLANs.

The following sections explain what you need to do to configure OVSDB-managed VXLANs properly for each Juniper Networks device that supports OVSDB and VXLAN:

## Understanding How to Manually Configure OVSDB-Managed VXLANs On Juniper Networks Devices

For each VXLAN that you plan to implement, you must first configure a logical switch, using NSX Manager or the NSX API. Based on the name and the VXLAN network identifier (VNI) that you specify, NSX automatically generates a universally unique identifier (UUID) for the logical switch. You must retain the UUID of the logical switch for later use.

Next, on the Juniper Networks device, you must manually configure the corresponding VXLAN, including the same VNI specified for the logical switch, using the Junos OS CLI. For the name of the VXLAN, you must specify the UUID for the logical switch.

When configuring a logical switch and a corresponding VXLAN, it is important that the UUID and VNI in both configurations are the same. If these elements are not the same, the logical switch and VXLAN cannot become operational, which means they cannot exchange MAC addresses learned in the NSX and Junos OS environments, respectively.

[Table 2 on page 12](#) provides a summary of the procedure that you must perform for each OVSDB-managed VXLAN on each Juniper Networks device, where to get more information about the configuration task, and the configuration statements that you must use to configure the VXLAN.

Table 2: Summary of Configuration Tasks for Manually Configuring An OVSDb-Managed VXLAN

Juniper Networks Device That Supports OVSDb and VXLAN	Configure Logical Switch, Using NSX Manager or the NSX API?	Where to Find More Configuration Information	Manually Configure Corresponding VXLAN on Juniper Networks Device?	Junos OS Statement to Configure the OVSDb-Managed VXLAN	Where to Find More Configuration Information
MX Series routers	Yes	See the documentation that accompanies NSX Manager or the NSX API.	Yes	<p>ovsdb-managed statement in the [edit bridge-domains <i>domain-name</i> vxlan] hierarchy.</p> <p>For the name of the VXLAN, specify the UUID for the logical switch configured in NSX Manager or in the NSX API.</p>	<a href="#">"Configuring OVSDb-Managed VXLANs" on page 26</a>
EX9200 switch	Yes	See the documentation that accompanies NSX Manager or the NSX API.	Yes	<p>ovsdb-managed statement in the [edit vlans <i>vlan-name</i> vxlan] hierarchy.</p> <p>For the name of the VXLAN, specify the UUID for the logical switch configured in NSX Manager or in the NSX API.</p>	<a href="#">"Configuring OVSDb-Managed VXLANs" on page 26</a>

RELATED DOCUMENTATION

| [show ovssdb logical-switch](#) | 164

OVSSDB Schema for Physical Devices

An Open vSwitch Database (OVSSDB) server runs on a Juniper Networks device that supports the OVSSDB management protocol. When this device is connected to one or more VMware NSX controllers, the connections provide a means through which the Juniper Networks device and the controllers can communicate.

In an NSX for vSphere environment, Juniper Networks devices that support OVSSDB and NSX controllers exchange control and statistical data. This data is stored in the OVSSDB database schema defined for physical devices. The schema resides in the OVSSDB server. The schema includes several tables. Juniper Networks devices and NSX controllers, both of which have OVSSDB clients, can add rows to the tables as well as monitor the tables for the addition, deletion, and modification of rows.

For example, the OVSSDB client on a Juniper Networks device or on an NSX controller can collect MAC routes learned by entities in the physical or virtual networks, respectively, and publish the routes to the appropriate table in the schema. By using the MAC routes and other information provided in the table, Juniper Networks devices in the physical network and entities in the virtual network can determine where to forward virtual machine (VM) traffic.

Some of the OVSSDB table names include the words *local* or *remote*—for example, the *unicast MACs local table* and the *unicast MACs remote table*. Information in *local* tables is learned by a Juniper Networks device that functions as a hardware virtual tunnel endpoint (VTEP), whereas information in *remote* tables is learned by other software or hardware VTEPs.

[Table 3 on page 13](#) describes the tables in the schema, the physical or virtual entity that is the source of the data provided in the table, and the command that you can enter in the CLI of the Juniper Networks device to get similar information.

Table 3: OVSSDB Schema Tables

Table Name	Description	Source of Information	Command
Global table	Includes the top-level configuration for the Juniper Networks device.	Juniper Networks device	–

Table 3: OVSDb Schema Tables *(Continued)*

Table Name	Description	Source of Information	Command
Manager table	Includes information for each NSX controller that is connected to the Juniper Networks device.	<ul style="list-style-type: none"> <li>• Juniper Networks device</li> <li>• NSX controller</li> </ul>	<a href="#">show ovssdb controller</a>
Physical switch table	Includes information about the Juniper Networks device on which a hardware VTEP is implemented. This table includes information only for the device on which the table resides.	Juniper Networks device	–
Physical port table	Includes information about OVSDb-managed interfaces.	Juniper Networks device	<a href="#">show ovssdb interface</a>
Logical switch table	Includes information about logical switches, which you configure in NSX Manager or in the NSX API, and the corresponding Virtual Extensible LANs (VXLANs), which are configured on the Juniper Networks device.	Juniper Networks device	<a href="#">show ovssdb logical-switch</a>
Logical binding statistics table	Includes statistics for OVSDb-managed interfaces.	Juniper Networks device	<a href="#">show ovssdb statistics interface</a>

Table 3: OVSDb Schema Tables *(Continued)*

Table Name	Description	Source of Information	Command
Physical locator table	Includes information about Juniper Networks devices configured as hardware VTEPs, software VTEPs, and service nodes.	Juniper Networks device	<a href="#">show ovssdb virtual-tunnel-end-point</a>
Physical locator set table	Lists service nodes for a logical switch.	Juniper Networks device	–
Unicast MACs remote table	Contains reachability information, including unicast MAC addresses, for entities in the virtual network.	NSX controller	<a href="#">show ovssdb mac</a>
Unicast MACs local table	Contains reachability information, including unicast MAC addresses, for entities in the physical network.	Juniper Networks device that is configured as a hardware VTEP.	<a href="#">show ovssdb mac</a>
Multicast MACs remote table	Includes only one row. In this row, the MAC column includes the keyword <code>unknown dst</code> along with a list of software VTEPs that host a cluster of service nodes, which handle multicast traffic.	NSX controller	<a href="#">show ovssdb mac</a>

Table 3: OVSDb Schema Tables *(Continued)*

Table Name	Description	Source of Information	Command
Multicast MACs local table	Includes one row for each logical switch. In this row, the MAC column includes the keyword <code>unknown dst</code> and a list of hardware VTEPs, which are identified by the IP address assigned to the hardware VTEP loopback interface (lo0). These hardware VTEPs can terminate or originate a VXLAN tunnel.	Juniper Networks device	<a href="#">show ovssdb mac</a>

## RELATED DOCUMENTATION

[Understanding the OVSDb Protocol Running on Juniper Networks Devices](#) | 6

[Understanding How to Set Up OVSDb Connections on a Juniper Networks Device](#) | 7



# Configuring OVSDB and VXLAN

## IN THIS CHAPTER

- [OVSDB and VXLAN Configuration Workflows for VMware NSX Environment | 17](#)
- [Installing OVSDB on Juniper Networks Devices | 22](#)
- [Creating and Installing an SSL Key and Certificate on a Juniper Networks Device for Connection with SDN Controllers | 23](#)
- [Setting Up the OVSDB Protocol on Juniper Networks Devices that Support Manual Configuration of VXLANs | 24](#)
- [Configuring OVSDB-Managed VXLANs | 26](#)
- [VMware NSX Configuration for Juniper Networks Devices Functioning as Virtual Tunnel Endpoints | 29](#)
- [Example: Setting Up Inter-VXLAN Unicast Routing and OVSDB Connections in a Data Center | 33](#)
- [Example: Setting Up Inter-VXLAN Unicast and Multicast Routing and OVSDB Connections in a Data Center | 48](#)
- [Example: Configuring VXLAN to VPLS Stitching with OVSDB | 66](#)
- [Example: Configuring Inter-VXLAN Traffic Routing from One Bridge Domain to Another Using an MX Series Router as a Layer 3 Gateway | 95](#)
- [Example: Passing Traffic Between Data Centers with DCI in an OVSDB-Managed Network with MX Series Routers | 105](#)

## OVSDB and VXLAN Configuration Workflows for VMware NSX Environment

### IN THIS SECTION

- [OVSDB and VXLAN Configuration Workflow for QFX Series Switches | 18](#)
- [OVSDB and VXLAN Configuration Workflow for MX Series Routers and EX9200 Switches | 20](#)

The workflow that you use to configure Open vSwitch Database (OVSDB) and Virtual Extensible LAN (VXLAN) in a VMware NSX environment depends on the Juniper Networks device that you are configuring. This topic provides more information about the following workflows:

## OVSDB and VXLAN Configuration Workflow for QFX Series Switches

[Table 4 on page 18](#) provides a high-level workflow of the tasks that you must perform to configure OVSDB and VXLAN on QFX Series switches. You must perform the tasks in [Table 4 on page 18](#) for each Juniper Networks switch that you plan to deploy in an OVSDB environment. In general, the successful completion of a task in this workflow depends on the successful completion of the previous task, so it is important to adhere to the task sequence provided in [Table 4 on page 18](#).

**Table 4: OVSDB and VXLAN Configuration Workflow for QFX Series Switches**

Sequence	Task	For More Information
1	Create and install a Secure Sockets Layer (SSL) key and certificate.	<a href="#">"Creating and Installing an SSL Key and Certificate on a Juniper Networks Device for Connection with SDN Controllers" on page 23.</a>
2	Enter the <code>set switch-options ovldb-managed</code> configuration mode command on the Juniper Networks switch.	–
3	Explicitly configure a connection to at least one VMware NSX controller.	<i>Setting Up OVSDB on Juniper Networks Devices That Support the Dynamic Configuration of VXLANs.</i>
4	Specify that each physical interface associated with a VXLAN is to be managed by OVSDB.	<i>Setting Up OVSDB on Juniper Networks Devices That Support the Dynamic Configuration of VXLANs.</i>
5	Configure a logical switch for each OVSDB-managed VXLAN that you plan to implement.	See the VMware documentation that accompanies NSX Manager or the NSX API.

Table 4: OVSDB and VXLAN Configuration Workflow for QFX Series Switches (*Continued*)

Sequence	Task	For More Information
6	<ul style="list-style-type: none"> <li>For each Juniper Network switch on which OVSDB-managed VXLANs and interfaces are configured, create a gateway.</li> <li>For each OVSDB-managed interface that you configure, create a gateway service.</li> <li>For each logical interface that you plan to implement for a VXLAN, configure a logical switch port.</li> </ul> <p><b>NOTE:</b> On QFX Series switches, when multiple logical interfaces are bound to an OVSDB-managed physical interface, keep in mind that all of the logical interfaces must be either access interfaces that handle untagged packets or trunk interfaces that handle tagged packets. An OVSDB-managed physical interface does not support a mix of access and trunk interfaces.</p>	<p>For general information about configuring gateways, gateway services, and logical switch ports, see the VMware documentation that accompanies NSX Manager or the NSX API.</p> <p>For key NSX Manager configuration details that help you configure gateways, gateway services, and logical switch ports so they function properly with their physical counterparts, see <a href="#">"VMware NSX Configuration for Juniper Networks Devices Functioning as Virtual Tunnel Endpoints" on page 29</a>.</p>
7	<p>Configure the loopback interface (lo0) on the Juniper Networks switch for VXLAN by entering the following configuration mode commands:</p> <ul style="list-style-type: none"> <li>set interfaces lo0 unit 0 family inet address <i>ip-address</i> primary</li> <li>set switch-options vtep-source-Interface lo0.0</li> </ul>	–

After you successfully complete task 6 in [Table 4 on page 18](#), the Juniper Networks switch dynamically creates a VXLAN for each logical switch that you configured in task 5. The Juniper Networks switch also dynamically creates and associates interfaces with each VXLAN. The dynamically created interface configuration is based on the gateway service and logical switch ports that you configured in task 6. For more information, see *Understanding Dynamically Configured VXLANs in an OVSDB Environment*.

For OVSDB-VXLAN scenarios in which Juniper Networks switches are commonly deployed, see the following topics:

- *Example: Setting Up a VXLAN Layer 2 Gateway and OVSDb Connections in a VMware NSX Environment (Trunk Interfaces Supporting Untagged Packets)*
- *Example: Setting Up a VXLAN Layer 2 Gateway and OVSDb Connections in a VMware NSX Environment (Trunk Interfaces Supporting Tagged Packets)*

## OVSDb and VXLAN Configuration Workflow for MX Series Routers and EX9200 Switches

[Table 5 on page 20](#) provides a high-level workflow of the tasks that you must perform to configure OVSDb and VXLAN on MX Series routers and EX9200 switches. You must perform the tasks in [Table 5 on page 20](#) for each Juniper Networks device that you plan to deploy in an OVSDb environment. In general, the successful completion of a task in this workflow depends on the successful completion of the previous task, so it is important to adhere to the task sequence provided in [Table 5 on page 20](#).

**Table 5: OVSDb and VXLAN Configuration Workflow for MX Series Routers and EX9200 Switches**

Sequence	Task	For More Information
1	Create and install an SSL key and certificate.	<a href="#">"Creating and Installing an SSL Key and Certificate on a Juniper Networks Device for Connection with SDN Controllers" on page 23.</a>
2	Explicitly configure a connection to at least one NSX controller.	<a href="#">"Setting Up the OVSDb Protocol on Juniper Networks Devices that Support Manual Configuration of VXLANs" on page 24.</a>
3	Specify that each physical interface associated with a VXLAN is to be managed by OVSDb.	<a href="#">"Setting Up the OVSDb Protocol on Juniper Networks Devices that Support Manual Configuration of VXLANs" on page 24.</a>
4	Configure a logical switch for each OVSDb-managed VXLAN that you plan to implement.	See the VMware documentation that accompanies NSX Manager or the NSX API.
5	Configure OVSDb-managed VXLANs.	<a href="#">"Configuring OVSDb-Managed VXLANs" on page 26.</a>

**Table 5: OVSDDB and VXLAN Configuration Workflow for MX Series Routers and EX9200 Switches**  
*(Continued)*

Sequence	Task	For More Information
6	<p>For each Juniper Network device on which OVSDDB-managed VXLANs and interfaces will be configured, create a gateway.</p> <p>For each OVSDDB-managed interface that you configure, create a gateway service.</p> <p>For each logical interface that you plan to implement for a VXLAN, configure a logical switch port.</p>	<p>For general information about configuring gateways, gateway services, and logical switch ports, see the VMware documentation that accompanies NSX Manager or the NSX API.</p> <p>For key NSX Manager configuration details that help you configure gateways, gateway services, and logical switch ports, so that they function properly with their physical counterparts, see <a href="#">"VMware NSX Configuration for Juniper Networks Devices Functioning as Virtual Tunnel Endpoints"</a> on page 29.</p>
7	<p>Configure the loopback interface (lo0) on the Juniper Networks device for VXLAN by entering the following configuration mode commands:</p> <ul style="list-style-type: none"> <li>• <code>set interfaces lo0 unit 0 family inet address <i>ip-address</i> primary</code></li> <li>• <code>set switch-options vtep-source-Interface lo0.0</code></li> </ul>	–

For OVSDDB-VXLAN scenarios in which these Juniper Networks devices are commonly deployed, see the following topics:

- ["Example: Setting Up Inter-VXLAN Unicast Routing and OVSDDB Connections in a Data Center"](#) on page 33
- ["Example: Setting Up Inter-VXLAN Unicast and Multicast Routing and OVSDDB Connections in a Data Center"](#) on page 48
- ["Example: Configuring VXLAN to VPLS Stitching with OVSDDB"](#) on page 66

## RELATED DOCUMENTATION

Verifying That a Logical Switch and Corresponding Junos OS OVSDb-Managed VXLAN Are Working Properly | 186

## Installing OVSDb on Juniper Networks Devices

**NOTE:** The Open vSwitch Database (OVSDb) software is included in the **jsdn** package. For some Juniper Networks devices, the **jsdn** package is included in the Junos OS software (**jinstall**) package. On these Juniper Networks devices, you do not need to install the separate **jsdn** package, which means that you can skip the task described in this topic. For information about which devices do not require installation of the separate **jsdn** package, see ["OVSDb Support on Juniper Networks Devices" on page 2](#).

If the **jsdn** package for your Juniper Networks device is not included in the **jinstall** package, you must copy a separate **jsdn** package to the Juniper Networks device and then install the package. The package name uses the following format:

*jsdn-packageID-release*

where:

- *packageID* identifies the package that must run on each Juniper Networks device.
- *release* identifies the release; for example, 16.2. The **jsdn** package release and the **jinstall** release running on the device must be the same.

To install the **jsdn** package on a Juniper Networks device:

1. Download the software package to the Juniper Networks device.
2. If an older **jsdn** package already exists on the Juniper Networks device, remove the package by issuing the request `system software delete operational mode` command.

```
user@device> request system software delete existing-ovsdb-package
```

3. Install the new **jsdn** package by using the request `system software add operational mode` command.

```
user@device> request system software add path-to-ovsdb-package
```

## RELATED DOCUMENTATION

[Understanding the OVSDb Protocol Running on Juniper Networks Devices | 6](#)

[OVSDb and VXLAN Configuration Workflows for VMware NSX Environment | 17](#)

## Creating and Installing an SSL Key and Certificate on a Juniper Networks Device for Connection with SDN Controllers

To secure a connection between a Juniper Networks device that supports the Open vSwitch Database (OVSDb) management protocol and one or more software-defined networking (SDN) controllers, the following Secure Sockets Layer (SSL) files must be present in the `/var/db/certs` directory on the device:

- **vtep-privkey.pem**
- **vtep-cert.pem**
- **ca-cert.pem**

You must create the **vtep-privkey.pem** and **vtep-cert.pem** files for the device and then install the two files in the `/var/db/certs` directory on the device.

Upon initial connection between a Juniper Networks device with OVSDb implemented and an SDN controller, the **ca-cert.pem** file is automatically generated and then installed in the `/var/db/certs` directory on the device.

**NOTE:** The situation at your particular site determines the possible methods that you can use to create the **vtep-privkey.pem** and **vtep-cert.pem** files and install them in the Juniper Networks device. Instead of providing procedures for all possible situations, this topic provides a procedure for one common scenario.

The procedure provided in this topic uses the OpenFlow public key infrastructure (PKI) management utility `ovs-pki` on a Linux computer to initialize a PKI and create the **vtep-privkey.pem** and **vtep-cert.pem** files. (If you have an existing PKI on your Linux computer, you can skip the step to initialize a new one.) By default, the utility initializes the PKI and places these files in the `/usr/local/share/openvswitch/pki` directory of the Linux computer.

To create and install an SSL key and certificate on a Juniper Networks device:

1. Initialize a PKI if one does not already exist on your Linux computer.

```
# ovs-pki init
```

2. On the same Linux computer on which the PKI exists, create a new key and certificate for the Juniper Networks device.

```
# ovs-pki req+sign vtep
```

3. Copy only the **vtep-privkey.pem** and **vtep-cert.pem** files from the Linux computer to the **/var/db/certs** directory on the Juniper Networks device.

## RELATED DOCUMENTATION

[Understanding How to Set Up OVSDDB Connections on a Juniper Networks Device](#) | 7

## Setting Up the OVSDDB Protocol on Juniper Networks Devices that Support Manual Configuration of VXLANs

To implement the Open vSwitch Database (OVSDDB) management protocol on a Juniper Networks device, you must explicitly configure a connection to at least one VMware NSX controller, using the Junos OS CLI.

All NSX controller connections are made on the management interface (fxp0 or me0) of the Juniper Networks device. This connection is secured by using the Secure Sockets Layer (SSL) protocol. The default port number over which the connection is made is 6632.

You must also specify that any interface implemented for a Virtual Extensible LAN (VXLAN) is managed by OVSDDB. By performing this configuration, you are essentially disabling the Juniper Networks device from learning about other Juniper Networks devices that function as hardware virtual tunnel endpoints (VTEPs) and the MAC addresses learned by the hardware VTEPs. Instead, you are enabling OVSDDB to learn about the other hardware VTEPs and the MAC addresses learned by the hardware VTEPs.

Before setting up OVSDDB on a Juniper Networks device, you must do the following:

- Ensure that the Juniper Networks device has a Juniper Networks VMware NSX software package installed, and that the software package release is the same as the Junos OS release running on the device.



- Create an SSL private key and certificate, and install them in the `/var/db/certs` directory of the Juniper Networks device. For more information, see ["Creating and Installing an SSL Key and Certificate on a Juniper Networks Device for Connection with SDN Controllers"](#) on page 23.
- Determine the IP address of the NSX controller.

To set up OVSDb on a Juniper Networks device:

1. Specify the IP address of the NSX controller.

```
[edit protocols ovbdb]
user@host# set controller ip-address
```

2. Specify SSL as the protocol that secures the connection.

```
[edit protocols ovbdb controller ip-address]
user@host# set protocol ssl
```

3. Set the number of the port over which the connection to the NSX controller is made.

```
[edit protocols ovbdb controller ip-address protocol ssl]
user@host# set port number
```

4. (Optional) Specify (in milliseconds) how long the connection can be inactive before an inactivity probe is sent.

```
[edit protocols ovbdb controller ip-address]
user@host# set inactivity-probe-duration milliseconds
```

5. (Optional) Specify (in milliseconds) how long the device must wait before it can try to connect to the NSX controller again if the previous attempt failed.

```
[edit protocols ovbdb controller ip-address]
user@host# set maximum-backoff-duration milliseconds
```

6. (Optional) Repeat steps 1 through 5 to explicitly configure a connection to an additional NSX controller in the same cluster.

7. Specify the interfaces that you want OVSDB to manage.

```
[edit protocols ovbdb]  
user@host# set interfaces interface-name unit logical-unit-number
```

## RELATED DOCUMENTATION

[Understanding How to Set Up OVSDB Connections on a Juniper Networks Device](#) | 7

## Configuring OVSDB-Managed VXLANs

**NOTE:** This topic does not apply to QFX5100 and QFX10002 switches, which support the dynamic configuration of OVSDB-managed VXLANs. Although the OVSDB-managed VXLAN configuration is automated on these switches, there are tasks that you must perform before and after the dynamic configuration. For more information about the required tasks, see *Understanding Dynamically Configured VXLANs in an OVSDB Environment*.

To implement the OVSDB management protocol on a Juniper Networks device, you must configure OVSDB-managed VXLANs.

For Layer 2 broadcast, unknown unicast, and multicast (BUM) traffic that originates in an OVSDB-managed VXLAN and is forwarded to interfaces within the same VXLAN, you can optionally enable ingress node replication. With this feature enabled, the Juniper Networks device handles the replication of these packets and the forwarding of the replicas to interfaces within the same OVSDB-managed VXLAN. For more information about using ingress node replication or a service node, which is the default way to handle Layer 2 BUM traffic, see "[Understanding How Layer 2 BUM and Layer 3 Routed Multicast Traffic Are Handled with OVSDB](#)" on page 8.

**NOTE:** When Juniper Networks devices replicate Layer 2 BUM packets to a large number of remote software virtual tunnel endpoints (VTEPs), the performance of the Juniper Networks devices might be impacted.

Before you configure VXLANs on a Juniper Networks device, using the Junos OS CLI:

- You must perform the configuration described in "[Setting Up the OVSDB Protocol on Juniper Networks Devices that Support Manual Configuration of VXLANs](#)" on page 24.

- For each OVSDb-managed VXLAN that you plan to configure on a Juniper Networks device, you must configure a logical switch in VMware NSX Manager or in the NSX API. (For information about configuring a logical switch, see the documentation that accompanies NSX Manager or the NSX API.) Based on the name and VXLAN network identifier (VNI) that you configure for the logical switch, NSX automatically generates a universally unique identifier (UUID) for the logical switch. You must retain the UUID of the logical switch for use when configuring a corresponding VXLAN on the Juniper Networks device as described in the following procedure.

To configure an OVSDb-managed VXLAN on a Juniper Networks device:

1. Configure the VXLANs that you want OVSDb to manage. You can configure the VXLANs in the context of a bridge domain, VLAN, routing instance, or switching instance.

**NOTE:** For the name of the bridge domain or VLAN, you must specify the UUID for the logical switch configured in NSX Manager or the NSX API.

Bridge domains:

```
[edit bridge-domains bridge-domain-name vxlan]
user@host# set ovssdb-managed
```

VLANs:

```
[edit vlans vlan-name vxlan]
user@device# set ovssdb-managed
```

Bridge domains within the specified routing instance:

```
[edit routing-instances routing-instance-name bridge-domains bridge-domain-name vxlan]
user@host# set ovssdb-managed
```

VLANs within the specified routing instance:

```
[edit routing-instances routing-instance-name vlans vlan-name vxlan]
user@device# set ovssdb-managed
```

Default switching instance within the specified routing instance:

```
[edit routing-instances routing-instance-name switch-options]
user@host# set ovssdb-managed
```

All VXLAN entities within the specified routing instance:

```
[edit routing-instances routing-instance-name vxlan]
user@host# set ovssdb-managed
```

2. (Optional) Enable ingress node replication to handle Layer 2 BUM traffic on interfaces in the same VXLAN in which the traffic originated. You can configure ingress node replication in the context of a bridge domain, VLAN, or routing instance.

Bridge domains:

```
[edit bridge-domains bridge-domain-name vxlan]
user@host# set ingress-node-replication
```

VLANs:

```
[edit vlans vlan-name vxlan]
user@device# set ingress-node-replication
```

Bridge domains, VLANs, or all VXLAN entities, respectively, within the specified routing instance:

```
[edit routing-instances routing-instance-name bridge-domains bridge-domain-name vxlan]
[edit routing-instances routing-instance-name vlans vlan-name vxlan]
[edit routing-instances routing-instance-name vxlan]
user@host# set ingress-node-replication
```

## RELATED DOCUMENTATION

[Understanding How to Manually Configure OVSSDB-Managed VXLANs | 10](#)

[OVSSDB and VXLAN Configuration Workflows for VMware NSX Environment | 17](#)

[Example: Setting Up Inter-VXLAN Unicast Routing and OVSSDB Connections in a Data Center | 33](#)

[Example: Setting Up Inter-VXLAN Unicast and Multicast Routing and OVSSDB Connections in a Data Center | 48](#)

## VMware NSX Configuration for Juniper Networks Devices Functioning as Virtual Tunnel Endpoints

### IN THIS SECTION

- [Creating a Gateway | 30](#)
- [Creating a Gateway Service | 30](#)
- [Creating a Logical Switch Port | 31](#)

When implementing the Open vSwitch Database (OVSDB) management protocol and Virtual Extensible LANs (VXLANs) on a Juniper Networks device, you must perform the following tasks in VMware NSX Manager or in the NSX API:

- For each Juniper Networks device on which OVSDB-managed VXLANs and physical interfaces are configured, you must create an NSX-equivalent entity, which is known as a *gateway*.
- For each OVSDB-managed physical interface that you configure on a Juniper Networks device, you must configure a gateway service—for example, a VTEP Layer 2 gateway service.
- For each logical interface that you want to implement for a VXLAN, you must configure a logical switch port.

The configurations described in this topic enable connectivity between physical servers in the physical network and virtual machines (VMs) in the virtual network.

This topic provides a high-level summary of the tasks that you must perform to create a gateway, gateway service, and logical switch ports. Although you can create these virtual entities either in NSX Manager or in the NSX API, this topic only describes how to perform the tasks in NSX Manager. Also, this topic does not include a complete procedure for each task. Rather, it includes key NSX Manager configuration details for ensuring the correct configuration of the virtual entities so that they function properly with the physical entities.

For complete information about performing the tasks described in this topic, see the documentation that accompanies NSX Manager.

This topic describes the following tasks:

## Creating a Gateway

In NSX Manager, you must create a gateway for each Juniper Networks device on which OVSDB-managed VXLANs and physical interfaces are configured. [Table 6 on page 30](#) provides a summary of key configuration fields in NSX Manager and how to configure them when creating a gateway.

**Table 6: Key Configurations to Create a Gateway in NSX Manager**

NSX Manager Configuration Page or Dialog Box	NSX Manager Configuration Field	How to Configure
Type	Transport Node Type	Select <b>Gateway</b> .
Properties	VTEP Enabled	Select <b>VTEP Enabled</b> .
Credential	Type	Select <b>Management Address</b> .
Credential	Management Address	Specify the management IP address of the Juniper Networks device.
Connections/Create Transport Connector	Transport Type	Select <b>VXLAN</b> .
Connections/Create Transport Connector	Transport Zone UUID	Select the UUID of an existing transport zone, or create a new transport zone.
Connections/Create Transport Connector	IP Address	Specify the IP address of the loopback interface (lo0) of the Juniper Networks device.

## Creating a Gateway Service

In NSX Manager, you must create a gateway service for each OVSDB-managed physical interface that you configure on a Juniper Networks device. Creating a gateway service essentially does the following for each OVSDB-managed physical interface:

- Specifies a gateway service—for example, a VTEP Layer 2 gateway service.
- Binds the interface to a gateway that you created in ["Creating a Gateway" on page 30](#).

Before you start this task, you must complete the following configurations:

- A gateway for the Juniper Networks device on which the OVSDB-managed physical interfaces are configured. See ["Creating a Gateway" on page 30](#).
- The OVSDB-managed physical interfaces on the Juniper Networks device. For information about configuring OVSDB-managed interfaces on Juniper Networks devices that support the dynamic configuration of VXLANs, see *Setting Up OVSDB on Juniper Networks Devices That Support the Dynamic Configuration of VXLANs*. For information about configuring OVSDB-managed interfaces on Juniper Networks devices that support the manual configuration of VXLANs, see ["Setting Up the OVSDB Protocol on Juniper Networks Devices that Support Manual Configuration of VXLANs" on page 24](#).

[Table 7 on page 31](#) provides a summary of key configuration fields in NSX Manager and how to configure them when creating a gateway service.

**Table 7: Key Configurations to Create a Gateway Service in NSX Manager**

NSX Manager Configuration Page or Dialog Box	NSX Manager Configuration Field	How to Configure
Type	Gateway Service Type	Select <b>VTEP L2 Gateway Service</b> .
Transport Nodes/Edit Gateway	Transport Node	Select the gateway that you created for the Juniper Networks device.
Transport Nodes/Edit Gateway	Port ID	Select an OVSDB-managed physical interface configured on the Juniper Networks device.

## Creating a Logical Switch Port

In NSX Manager, you must create a logical switch port for each logical interface that you plan to implement for a VXLAN. Creating the logical switch port essentially does the following for each logical interface:

- Binds the logical switch port to a logical switch that you created in NSX Manager or in the NSX API.
- Binds the logical interface to a gateway service that you configured in ["Creating a Gateway Service" on page 30](#).

Before you start this task, you must complete the following configurations:

- A logical switch with which this logical port is associated. For information about configuring a logical switch, see the VMware documentation that accompanies NSX Manager or the NSX API.

- A gateway service that specifies the OVSDB-managed physical interface with which the logical interface is associated. See ["Creating a Gateway Service" on page 30](#).

[Table 8 on page 32](#) provides a summary of key configuration fields in NSX Manager and how to configure them when creating a logical switch port.

**Table 8: Key Configurations to Create a Logical Switch Port in NSX Manager**

NSX Manager Configuration Page or Dialog Box	NSX Manager Configuration Field	How to Configure
Logical Switch	Logical Switch UUID	Select the UUID of a logical switch.
Attachment	Attachment Type	Select <b>VTEP L2 Gateway</b> .
Attachment	VTEP L2 Gateway Service UUID	Select the UUID of a gateway service.
Attachment	VLAN	<p>Select <b>0</b> to specify that the port handles untagged packets.</p> <p>Select <b>1</b> through <b>4000</b> to specify that the port handles tagged packets.</p> <p><b>NOTE:</b> VLAN ID 4094 is reserved for a native VLAN in an OVSDB environment. Specifying this VLAN ID results in an error message. Do not specify this VLAN ID or any VLAN ID not in the accepted range.</p>

## RELATED DOCUMENTATION

[OVSDB and VXLAN Configuration Workflows for VMware NSX Environment](#) | 17



## Example: Setting Up Inter-VXLAN Unicast Routing and OVSDb Connections in a Data Center

### IN THIS SECTION

- Requirements | 33
- Overview and Topology | 34
- Configuration | 38
- Verification | 45

This example shows how to set up a data center in which virtual machines (VMs) in different Virtual Extensible LANs (VXLANs) need to communicate. The Juniper Networks device that is integrated into this environment functions as a hardware virtual tunnel endpoint (VTEP) that can route VM traffic from one VXLAN (Layer 2) environment to another.

The Juniper Networks device implements the Open vSwitch Database (OVSDb) management protocol and has a connection with a VMware NSX controller, both of which enable the device and the NSX controller to exchange MAC routes to and from VMs in the physical and virtual networks.

This example explains how to configure a Juniper Networks device as a hardware VTEP, set up the routing of unicast packets between VXLANs, and set up an OVSDb connection with an NSX controller. For information about setting up the routing of unicast and multicast packets between VXLANs, see ["Example: Setting Up Inter-VXLAN Unicast and Multicast Routing and OVSDb Connections in a Data Center" on page 48](#).

### Requirements

The topology for this example includes the following hardware and software components:

- A cluster of five NSX controllers.
- NSX Manager.
- A service node that handles broadcast, unknown unicast, and multicast (BUM) traffic within each of the two VXLANs.
- Two hosts, each of which includes VMs managed by a hypervisor. Each hypervisor includes a software VTEP. The VMs on each of the hosts belong to different VXLANs.
- A Juniper Networks device that routes VM traffic between the two VXLANs. For example, an MX Series router running Junos OS Release 14.1R2 or later, or an EX9200 switch running Junos OS

release 14.2 or later. The Juniper Networks device must also run an OVSDB software package, and the release of this package must be the same as the Junos OS release running on the device. This device is configured to function as a hardware VTEP.

Before you begin the configuration of the Juniper Networks device, you need to perform the following tasks:

- In NSX Manager or the NSX API, specify the IP address of the service node.
- In NSX Manager or the NSX API, configure a logical switch for each VXLAN that OVSDB will manage. This example implements two OVSDB-managed VXLANs; therefore, you must configure two logical switches. After the configuration of each logical switch, NSX automatically generates a universally unique identifier (UUID) for the logical switch. If you have not already, retrieve the UUID for each logical switch. A sample UUID is 28805c1d-0122-495d-85df-19abd647d772. When configuring the equivalent VXLANs on the Juniper Networks device, you must use the UUID of the logical switch as the bridge domain or VLAN name.

For more information about logical switches and VXLANs, see ["Understanding How to Manually Configure OVSDB-Managed VXLANs" on page 10](#).

- Create an SSL private key and certificate, and install them in the `/var/db/certs` directory of the Juniper Networks device. For more information, see ["Creating and Installing an SSL Key and Certificate on a Juniper Networks Device for Connection with SDN Controllers" on page 23](#).

For information about using NSX Manager or the NSX API to perform these configuration tasks, see the documentation that accompanies the respective products.

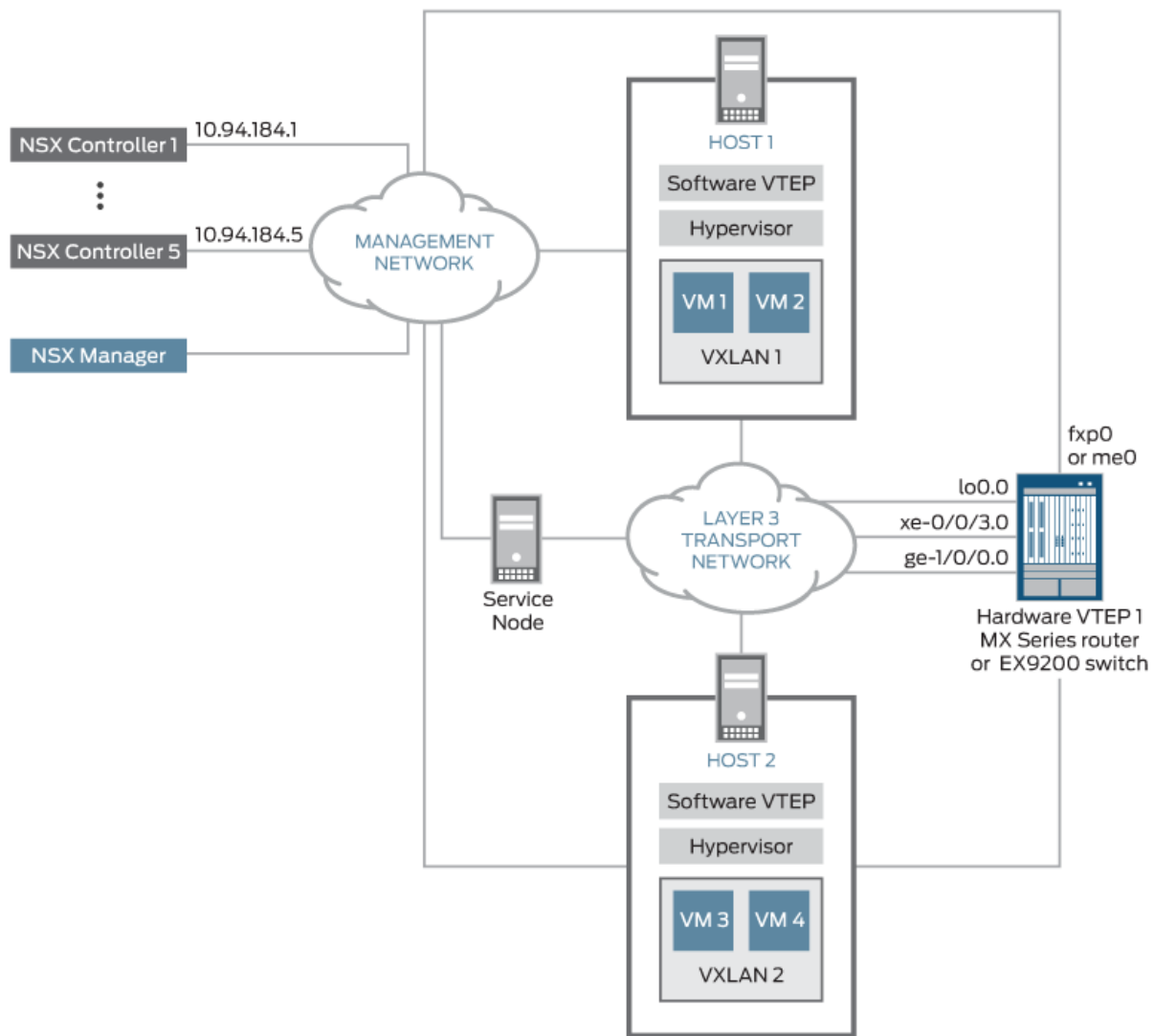
## Overview and Topology

### IN THIS SECTION

- [Topology | 36](#)

In the topology shown in [Figure 3 on page 35](#), VM 1 in VXLAN 1 needs to communicate with VM 3 in VXLAN 2. To enable this communication, hardware VTEP 1, which can be an MX Series router or an EX9200 switch, is configured to route VM unicast traffic between the two VXLANs.

**Figure 3: Inter-VXLAN Unicast Routing and OVSDB Topology**



On hardware VTEP 1, a routing instance (virtual switch) is set up. Within the routing instance, two VXLANs are configured: VXLAN 1 and VXLAN 2. Each VXLAN has an integrated routing and bridging (IRB) interface associated with it. The IRB interfaces handle the routing of VM unicast traffic between the VXLANs,

Within each of the two VXLANs, a service node replicates Layer 2 BUM packets then forwards the replicas to all interfaces in the VXLANs. Having the service node handle the Layer 2 BUM traffic is the default behavior, and no configuration is required for this Juniper Networks device.

On hardware VTEP 1, a connection with an NSX controller is configured on the management interface (fxp0 for an MX Series router and me0 for an EX9200 switch). This configuration enables the NSX controller to push MAC routes for VM 1 and VM 3 to the hardware VTEP by way of the table for remote unicast MAC addresses in the OVSDb schema for physical devices.

Each VXLAN-encapsulated packet must include a source IP address, which identifies the source hardware or software VTEP, in the outer IP header. In this example, for hardware VTEP 1, the IP address of the loopback interface (lo0.0) is used.

In this example, the tracing of all OVSDb events is configured. The output of the OVSDb events are placed in a file named **ovsdb**, which is stored in the **/var/log** directory. By default, a maximum of 10 trace files can exist, and the configured maximum size of each file is 50 MB.

## Topology

[Table 9 on page 36](#) describes the components for setting up inter-VXLAN routing and an OVSDb connection.

**Table 9: Components for Setting Up Inter-VXLAN Routing and OVSDb Connections in a Data Center**

Property	Settings
Routing instance	Name: vx1  Type: virtual switch  OVSDb-managed VXLANs included: VXLAN 1 and VXLAN 2
VXLAN 1	Bridge domain or VLAN associated with: 28805c1d-0122-495d-85df-19abd647d772  Interface: xe-0/0/2.0  VLAN ID: 100  VNI: 100

**Table 9: Components for Setting Up Inter-VXLAN Routing and OVSDb Connections in a Data Center**  
*(Continued)*

Property	Settings
VXLAN 2	<p>Bridge domain or VLAN associated with: 96a382cd-a570-4ac8-a77a-8bb8b16bde70</p> <p>Interface: xe-1/2/0.0</p> <p>VLAN ID: 200</p> <p>VNI: 200</p>
Inter-VXLAN unicast routing and forwarding with IRB interfaces	<p>VXLAN 1: irb.0; 10.20.20.1/24; associated with routing interface vx1, and bridge domain or VLAN 28805c1d-0122-495d-85df-19abd647d772</p> <p>VXLAN 2: irb.1; 10.10.10.3/24; associated with routing interface vx1, and bridge domain or VLAN 96a382cd-a570-4ac8-a77a-8bb8b16bde70</p>
Handling of BUM traffic in each VXLAN	<p>Service node</p> <p><b>NOTE:</b> By default, one or more service nodes handle Layer 2 BUM traffic in a VXLAN; therefore, no configuration is required.</p>
NSX controller	IP address: 10.94.184.1
Hardware VTEP source identifier	<p>Source interface: loopback (lo0.0)</p> <p>Source IP address: 10.19.19.19/32</p>
OVSDb tracing operations	<p>Filename: /var/log/ovsdb</p> <p>File size: 50 MB</p> <p>Flag: All</p>

## Configuration

### IN THIS SECTION

- [CLI Quick Configuration | 38](#)
- [Configuring an MX Series Router as a Hardware VTEP with an OVSDDB Connection | 40](#)
- [Configuring an EX9200 Switch as a Hardware VTEP with an OVSDDB Connection | 43](#)

An MX Series router or an EX9200 switch can function as hardware VTEP 1 in this example. Because the configuration for each device is slightly different, a separate configuration is provided for each device.

To configure inter-VXLAN unicast routing and OVSDDB connections in a data center topology, you need to perform one of these tasks:

### 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 (for example, IP addresses, interface names, and UUIDs), copy and paste the commands into the CLI at the [edit] hierarchy level, and then enter `commit` from configuration mode.

**NOTE:** After completing this configuration, you must configure a gateway, which is the NSX-equivalent of a hardware VTEP. This example implements one hardware VTEP; therefore, you must configure one gateway, a gateway service, and a logical switch port using NSX Manager or the NSX API. For more information about the tasks you must perform and key NSX Manager configuration details, see ["VMware NSX Configuration for Juniper Networks Devices Functioning as Virtual Tunnel Endpoints" on page 29](#).

MX Series router configuration:

```
set chassis network-services enhanced-ip
set interfaces xe-0/0/3 unit 0 family inet address 10.50.50.2/24
set interfaces ge-1/0/0 unit 0 family inet address 10.100.100.99/24
set routing-options router-id 10.19.19.19
set protocols ospf area 0.0.0.0 interface xe-0/0/3.0
set protocols ospf area 0.0.0.0 interface ge-1/0/0.0
```

```

set protocols ospf area 0.0.0.0 interface lo0.0
set interfaces xe-0/0/2 unit 0 family bridge interface-mode access
set interfaces xe-0/0/2 unit 0 family bridge vlan-id 100
set interfaces xe-1/2/0 unit 0 family bridge interface-mode access
set interfaces xe-1/2/0 unit 0 family bridge vlan-id 200
set interfaces irb unit 0 family inet address 10.20.20.1/24
set interfaces irb unit 1 family inet address 10.10.10.3/24
set routing-instances vx1 vtep-source-interface lo0.0
set routing-instances vx1 instance-type virtual-switch
set routing-instances vx1 interface xe-0/0/2.0
set routing-instances vx1 interface xe-1/2/0.0
set routing-instances vx1 bridge-domains 28805c1d-0122-495d-85df-19abd647d772 vlan-id 100
set routing-instances vx1 bridge-domains 28805c1d-0122-495d-85df-19abd647d772 routing-interface
irb.0
set routing-instances vx1 bridge-domains 28805c1d-0122-495d-85df-19abd647d772 vxlan ovssdb-managed
set routing-instances vx1 bridge-domains 28805c1d-0122-495d-85df-19abd647d772 vxlan vni 100
set routing-instances vx1 bridge-domains 96a382cd-a570-4ac8-a77a-8bb8b16bde70 vlan-id 200
set routing-instances vx1 bridge-domains 96a382cd-a570-4ac8-a77a-8bb8b16bde70 routing-interface
irb.1
set routing-instances vx1 bridge-domains 96a382cd-a570-4ac8-a77a-8bb8b16bde70 vxlan ovssdb-managed
set routing-instances vx1 bridge-domains 96a382cd-a570-4ac8-a77a-8bb8b16bde70 vxlan vni 200
set interfaces lo0 unit 0 family inet address 10.19.19.19/32 primary
set interfaces lo0 unit 0 family inet address 10.19.19.19/32 preferred
set protocols ovssdb traceoptions file ovssdb
set protocols ovssdb traceoptions file size 50m
set protocols ovssdb traceoptions flag all
set protocols ovssdb controller 10.94.184.1
set protocols ovssdb interfaces xe-0/0/2.0
set protocols ovssdb interfaces xe-1/2/0.0

```

EX9200 switch configuration:

```

set interfaces xe-0/0/3 unit 0 family inet address 10.50.50.2/24
set interfaces ge-1/0/0 unit 0 family inet address 10.100.100.99/24
set routing-options router-id 10.19.19.19
set protocols ospf area 0.0.0.0 interface xe-0/0/3.0
set protocols ospf area 0.0.0.0 interface ge-1/0/0.0
set protocols ospf area 0.0.0.0 interface lo0.0
set interfaces xe-0/0/2 unit 0 family ethernet-switching interface-mode access
set interfaces xe-0/0/2 unit 0 family ethernet-switching vlan-id 100
set interfaces xe-1/2/0 unit 0 family ethernet-switching interface-mode access
set interfaces xe-1/2/0 unit 0 family ethernet-switching vlan-id 200

```

```

set interfaces irb unit 0 family inet address 10.20.20.1/24
set interfaces irb unit 1 family inet address 10.10.10.3/24
set routing-instances vx1 vtep-source-interface lo0.0
set routing-instances vx1 instance-type virtual-switch
set routing-instances vx1 interface xe-0/0/2.0
set routing-instances vx1 interface xe-1/2/0.0
set routing-instances vx1 vlans 28805c1d-0122-495d-85df-19abd647d772 vlan-id 100
set routing-instances vx1 vlans 28805c1d-0122-495d-85df-19abd647d772 routing-interface irb.0
set routing-instances vx1 vlans 28805c1d-0122-495d-85df-19abd647d772 vxlan ovssdb-managed
set routing-instances vx1 vlans 28805c1d-0122-495d-85df-19abd647d772 vxlan vni 100
set routing-instances vx1 vlans 96a382cd-a570-4ac8-a77a-8bb8b16bde70 vlan-id 200
set routing-instances vx1 vlans 96a382cd-a570-4ac8-a77a-8bb8b16bde70 routing-interface irb.1
set routing-instances vx1 vlans 96a382cd-a570-4ac8-a77a-8bb8b16bde70 vxlan ovssdb-managed
set routing-instances vx1 vlans 96a382cd-a570-4ac8-a77a-8bb8b16bde70 vxlan vni 200
set interfaces lo0 unit 0 family inet address 10.19.19.19/32 preferred
set interfaces lo0 unit 0 family inet address 10.19.19.19/32 primary
set protocols ovssdb traceoptions file ovssdb
set protocols ovssdb traceoptions file size 50m
set protocols ovssdb traceoptions flag all
set protocols ovssdb controller 10.94.184.1
set protocols ovssdb interfaces xe-0/0/2.0
set protocols ovssdb interfaces xe-1/2/0.0

```

## Configuring an MX Series Router as a Hardware VTEP with an OVSSDB Connection

### Step-by-Step Procedure

To configure an MX Series router as hardware VTEP 1 with an OVSSDB connection to an NSX controller, follow these steps:

1. Create the Layer 3 network.

```

[edit chassis]
user@router# set network-services enhanced-ip
[edit interfaces]
user@router# set xe-0/0/3 unit 0 family inet address 10.50.50.2/24
user@router# set ge-1/0/0 unit 0 family inet address 10.100.100.99/24
[edit routing-options]
user@router# set router-id 10.19.19.19
[edit protocols]
user@router# set ospf area 0.0.0.0 interface xe-0/0/3.0

```



```
user@router# set ospf area 0.0.0.0 interface ge-1/0/0.0
user@router# set ospf area 0.0.0.0 interface lo0.0
```

2. Create an access interface for VXLAN 1, and associate the interface with the VXLAN.

```
[edit interfaces]
user@router# set xe-0/0/2 unit 0 family bridge interface-mode access
user@router# set xe-0/0/2 unit 0 family bridge vlan-id 100
```

3. Create an access interface for VXLAN 2, and associate the interface with the VXLAN.

```
[edit interfaces]
user@router# set xe-1/2/0 unit 0 family bridge interface-mode access
user@router# set xe-1/2/0 unit 0 family bridge vlan-id 200
```

4. Create an IRB interface to handle inter-VXLAN unicast traffic for VXLAN 1.

```
[edit interfaces]
user@router# set irb unit 0 family inet address 10.20.20.1/24
```

5. Create an IRB interface to handle inter-VXLAN unicast traffic for VXLAN 2.

```
[edit interfaces]
user@router# set irb unit 1 family inet address 10.10.10.3/24
```

6. Set up the virtual switch routing instance.

```
[edit routing-instances]
user@router# set vx1 vtep-source-interface lo0.0
user@router# set vx1 instance-type virtual-switch
user@router# set vx1 interface xe-0/0/2.0
user@router# set vx1 interface xe-1/2/0.0
user@router# set vx1 bridge-domains 28805c1d-0122-495d-85df-19abd647d772 vlan-id 100
user@router# set vx1 bridge-domains 28805c1d-0122-495d-85df-19abd647d772 routing-interface
irb.0
user@router# set vx1 bridge-domains 28805c1d-0122-495d-85df-19abd647d772 vxlan ovsdb-
managed
user@router# set vx1 bridge-domains 28805c1d-0122-495d-85df-19abd647d772 vxlan vni 100
```

```

user@router# set vx1 bridge-domains 96a382cd-a570-4ac8-a77a-8bb8b16bde70 vlan-id 200
user@router# set vx1 bridge-domains 96a382cd-a570-4ac8-a77a-8bb8b16bde70 routing-interface
irb.1
user@router# set vx1 bridge-domains 96a382cd-a570-4ac8-a77a-8bb8b16bde70 vxlan ovsdb-
managed
user@router# set vx1 bridge-domains 96a382cd-a570-4ac8-a77a-8bb8b16bde70 vxlan vni 200

```

7. 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@router# set lo0 unit 0 family inet address 10.19.19.19/32 primary
user@router# set lo0 unit 0 family inet address 10.19.19.19/32 preferred

```

8. Set up OVSDB tracing operations.

```

[edit protocols]
user@router# set ovsdb traceoptions file ovsdb
user@router# set ovsdb traceoptions file size 50m
user@router# set ovsdb traceoptions flag all

```

9. Configure a connection with an NSX controller.

```

[edit protocols]
user@router# set ovsdb controller 10.94.184.1

```

10. Configure interfaces xe-0/0/2.0 and xe-1/2/0.0 to be managed by OVSDB.

```

[edit protocols]
user@router# set ovsdb interfaces xe-0/0/2.0
user@router# set ovsdb interfaces xe-1/2/0.0

```

**NOTE:** After completing this configuration, you must configure a gateway, which is the NSX-equivalent of a hardware VTEP. This example implements one hardware VTEP; therefore, you must configure one gateway, a gateway service, and a logical switch port by using NSX Manager or the NSX API. For more information about the tasks you must perform and key

NSX Manager configuration details, see ["VMware NSX Configuration for Juniper Networks Devices Functioning as Virtual Tunnel Endpoints"](#) on page 29.

## Configuring an EX9200 Switch as a Hardware VTEP with an OVSDDB Connection

### Step-by-Step Procedure

To configure an EX9200 switch as hardware VTEP 1 with an OVSDDB connection to an NSX controller, follow these steps:

1. Create the Layer 3 network.

```
[edit chassis]
[edit interfaces]
user@switch# set xe-0/0/3 unit 0 family inet address 10.50.50.2/24
user@switch# set ge-1/0/0 unit 0 family inet address 10.100.100.99/24
[edit routing-options]
user@switch# set router-id 10.19.19.19
[edit protocols]
user@switch# set ospf area 0.0.0.0 interface xe-0/0/3.0
user@switch# set ospf area 0.0.0.0 interface ge-1/0/0.0
user@switch# set ospf area 0.0.0.0 interface lo0.0
```

2. Create an access interface for VXLAN 1, and associate the interface with the VXLAN.

```
[edit interfaces]
user@switch# set xe-0/0/2 unit 0 family ethernet-switching interface-mode access
user@switch# set xe-0/0/2 unit 0 family ethernet-switching vlan-id 100
```

3. Create an access interface for VXLAN 2, and associate the interface with the VXLAN.

```
[edit interfaces]
user@switch# set xe-1/2/0 unit 0 family ethernet-switching interface-mode access
user@switch# set xe-1/2/0 unit 0 family ethernet-switching vlan-id 200
```

4. Create an IRB interface to handle inter-VXLAN unicast traffic for VXLAN 1.

```
[edit interfaces]
user@switch# set irb unit 0 family inet address 10.20.20.1/24
```

5. Create an IRB interface to handle inter-VXLAN unicast traffic for VXLAN 2.

```
[edit interfaces]
user@switch# set irb unit 1 family inet address 10.10.10.3/24
```

6. Set up the virtual switch routing instance.

```
[edit routing-instances]
user@switch# set vx1 vtep-source-interface lo0.0
user@switch# set vx1 instance-type virtual-switch
user@switch# set vx1 interface xe-0/0/2.0
user@switch# set vx1 interface xe-1/2/0.0
user@switch# set vx1 vlans 28805c1d-0122-495d-85df-19abd647d772 vlan-id 100
user@switch# set vx1 vlans 28805c1d-0122-495d-85df-19abd647d772 routing-interface irb.0
user@switch# set vx1 vlans 28805c1d-0122-495d-85df-19abd647d772 vxlan ovsdb-managed
user@switch# set vx1 vlans 28805c1d-0122-495d-85df-19abd647d772 vxlan vni 100
user@switch# set vx1 vlans 96a382cd-a570-4ac8-a77a-8bb8b16bde70 vlan-id 200
user@switch# set vx1 vlans 96a382cd-a570-4ac8-a77a-8bb8b16bde70 routing-interface irb.1
user@switch# set vx1 vlans 96a382cd-a570-4ac8-a77a-8bb8b16bde70 vxlan ovsdb-managed
user@switch# set vx1 vlans 96a382cd-a570-4ac8-a77a-8bb8b16bde70 vxlan vni 200
```

7. 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 10.19.19.19/32 primary
user@switch# set lo0 unit 0 family inet address 10.19.19.19/32 preferred
```

8. Set up tracing operations to be performed for the OVSDb management protocol.

```
[edit protocols]
user@switch# set ovsdb traceoptions file ovsdb
```

```
user@switch# set ovssdb traceoptions file size 50m
user@switch# set ovssdb traceoptions flag all
```

9. Configure a connection with an NSX controller.

```
[edit protocols]
user@switch# set ovssdb controller 10.94.184.1
```

10. Configure interfaces xe-0/0/2.0 and xe-1/2/0.0 to be managed by OVSSDB.

```
[edit protocols]
user@router# set ovssdb interfaces xe-0/0/2.0
user@router# set ovssdb interfaces xe-1/2/0.0
```

**NOTE:** After completing this configuration, you must configure a gateway, which is the NSX-equivalent of a hardware VTEP. This example implements one hardware VTEP; therefore, you must configure one gateway, a gateway service, and a logical switch port by using NSX Manager or the NSX API. For more information about the tasks you must perform and key NSX Manager configuration details, see ["VMware NSX Configuration for Juniper Networks Devices Functioning as Virtual Tunnel Endpoints" on page 29](#).

## Verification

### IN THIS SECTION

- [Verifying the Logical Switches | 46](#)
- [Verifying the MAC Addresses of VM 1 and VM 3 | 46](#)
- [Verifying the NSX Controller Connection | 47](#)

## Verifying the Logical Switches

### Purpose

Verify that logical switches with the UUIDs of 28805c1d-0122-495d-85df-19abd647d772 and 96a382cd-a570-4ac8-a77a-8bb8b16bde70 are configured in NSX Manager or in the NSX API, and that information about the logical switches is published in the OVSDb schema.

### Action

Issue the `show ovssdb logical-switch operational mode` command.

```
user@host> show ovssdb logical-switch
Logical switch information:
Logical Switch Name: 28805c1d-0122-495d-85df-19abd647d772
Flags: Created by both
VNI: 100
Num of Remote MAC: 1
Num of Local MAC: 0
Logical Switch Name: 96a382cd-a570-4ac8-a77a-8bb8b16bde70
Flags: Created by both
VNI: 200
Num of Remote MAC: 1
Num of Local MAC: 1
```

### Meaning

The output verifies that information about the logical switches is published in the OVSDb schema. The Created by both state indicates that the logical switches are configured in NSX Manager or the NSX API, and the corresponding VXLANs are configured on the Juniper Networks device. In this state, the logical switches and VXLANs are operational.

If the state of the logical switches is something other than Created by both, see ["Troubleshooting a Nonoperational Logical Switch and Corresponding Junos OS OVSDb-Managed VXLAN"](#) on page 188.

## Verifying the MAC Addresses of VM 1 and VM 3

### Purpose

Verify that the MAC addresses of VM 1 and VM 3 are present in the OVSDb schema.

## Action

Issue the `show ovssdb mac remote` operational mode command to verify that the MAC addresses for VM 1 and VM 3 are present.

```
user@host> show ovssdb mac remote
Logical Switch Name: 28805c1d-0122-495d-85df-19abd647d772
  Mac          IP          Encapsulation  Vtep
  Address      Address
  08:33:9d:5f:a7:f1  0.0.0.0      Vxlan over Ipv4  10.19.19.19
Logical Switch Name: 96a382cd-a570-4ac8-a77a-8bb8b16bde70
  Mac          IP          Encapsulation  Vtep
  Address      Address
  a8:59:5e:f6:38:90  0.0.0.0      Vxlan over Ipv4  10.19.19.10
```

## Meaning

The output shows that the MAC addresses for VM 1 and VM 3 are present and are associated with logical switches with the UUIDs of 28805c1d-0122-495d-85df-19abd647d772 and 96a382cd-a570-4ac8-a77a-8bb8b16bde70, respectively. Given that the MAC addresses are present, VM 1 and VM 3 are reachable through hardware VTEP 1.

## Verifying the NSX Controller Connection

### Purpose

Verify that the connection with the NSX controller is up.

## Action

Issue the `show ovssdb controller` operational mode command, and verify that the controller connection state is up.

```
user@host> show ovssdb controller
VTEP controller information:
Controller IP address: 10.94.184.1
Controller protocol: ssl
Controller port: 6632
Controller connection: up
Controller seconds-since-connect: 542325
```

```
Controller seconds-since-disconnect: 542346
Controller connection status: active
```

## Meaning

The output shows that the connection state of the NSX controller is up, in addition to other information about the controller. When this connection is up, OVSDB is enabled on the Juniper Networks device.

## RELATED DOCUMENTATION

| *OVSDB Schema for Physical Devices*

## Example: Setting Up Inter-VXLAN Unicast and Multicast Routing and OVSDB Connections in a Data Center

### IN THIS SECTION

- [Requirements | 49](#)
- [Overview and Topology | 50](#)
- [Configuration | 54](#)
- [Verification | 63](#)

This example shows how to set up a data center in which virtual machines (VMs) in different Virtual Extensible LANs (VXLANs) need to communicate. The Juniper Networks device that is integrated into this environment functions as a hardware virtual tunnel endpoint (VTEP) that can route VM traffic from one VXLAN (Layer 2) environment to another.

The Juniper Networks device implements the Open vSwitch Database (OVSDB) management protocol and has a connection with a VMware NSX controller, both of which enable the device and the NSX controller to exchange MAC routes to and from VMs in the physical and virtual networks.

This example explains how to configure a Juniper Networks device as a hardware VTEP, set up the routing of unicast and multicast packets between VXLANs, and set up an OVSDB connection with an NSX controller. For information about setting up the routing of unicast packets only between VXLANs,



see ["Example: Setting Up Inter-VXLAN Unicast Routing and OVSDB Connections in a Data Center"](#) on page 33.

## Requirements

The topology for this example includes the following hardware and software components:

- A cluster of five NSX controllers.
- NSX Manager.
- A service node that handles broadcast, unknown unicast, and multicast (BUM) traffic within each of the two VXLANs.
- Two hosts, each of which includes VMs managed by a hypervisor. Each hypervisor includes a software VTEP. The VMs on each of the hosts belong to different VXLANs.
- A Juniper Networks device that routes VM traffic between the two VXLANs. For example, an MX Series router running Junos OS Release 14.1R2 or later, or an EX9200 switch running Junos OS release 14.2 or later. The Juniper Networks device must also run an OVSDB software package, and the release of this package must be the same as the Junos OS release running on the device. This device is configured to function as a hardware VTEP.

Before you begin the configuration of the Juniper Networks device, you need to perform the following tasks:

- In NSX Manager or the NSX API, specify the IP address of the service node.
- In NSX Manager or the NSX API, configure a logical switch for each VXLAN that OVSDB will manage. This example implements two OVSDB-managed VXLANs; therefore, you must configure two logical switches. After the configuration of each logical switch, NSX automatically generates a universally unique identifier (UUID) for the logical switch. If you have not already, retrieve the UUID for each logical switch. A sample UUID is 28805c1d-0122-495d-85df-19abd647d772. When configuring the equivalent VXLANs on the Juniper Networks device, you must use the UUID of the logical switch as the bridge domain or VLAN name.

For more information about logical switches and VXLANs, see ["Understanding How to Manually Configure OVSDB-Managed VXLANs"](#) on page 10.

- Create an SSL private key and certificate, and install them in the `/var/db/certs` directory of the Juniper Networks device. For more information, see ["Creating and Installing an SSL Key and Certificate on a Juniper Networks Device for Connection with SDN Controllers"](#) on page 23.

For information about using NSX Manager or the NSX API to perform these configuration tasks, see the documentation that accompanies the respective products.

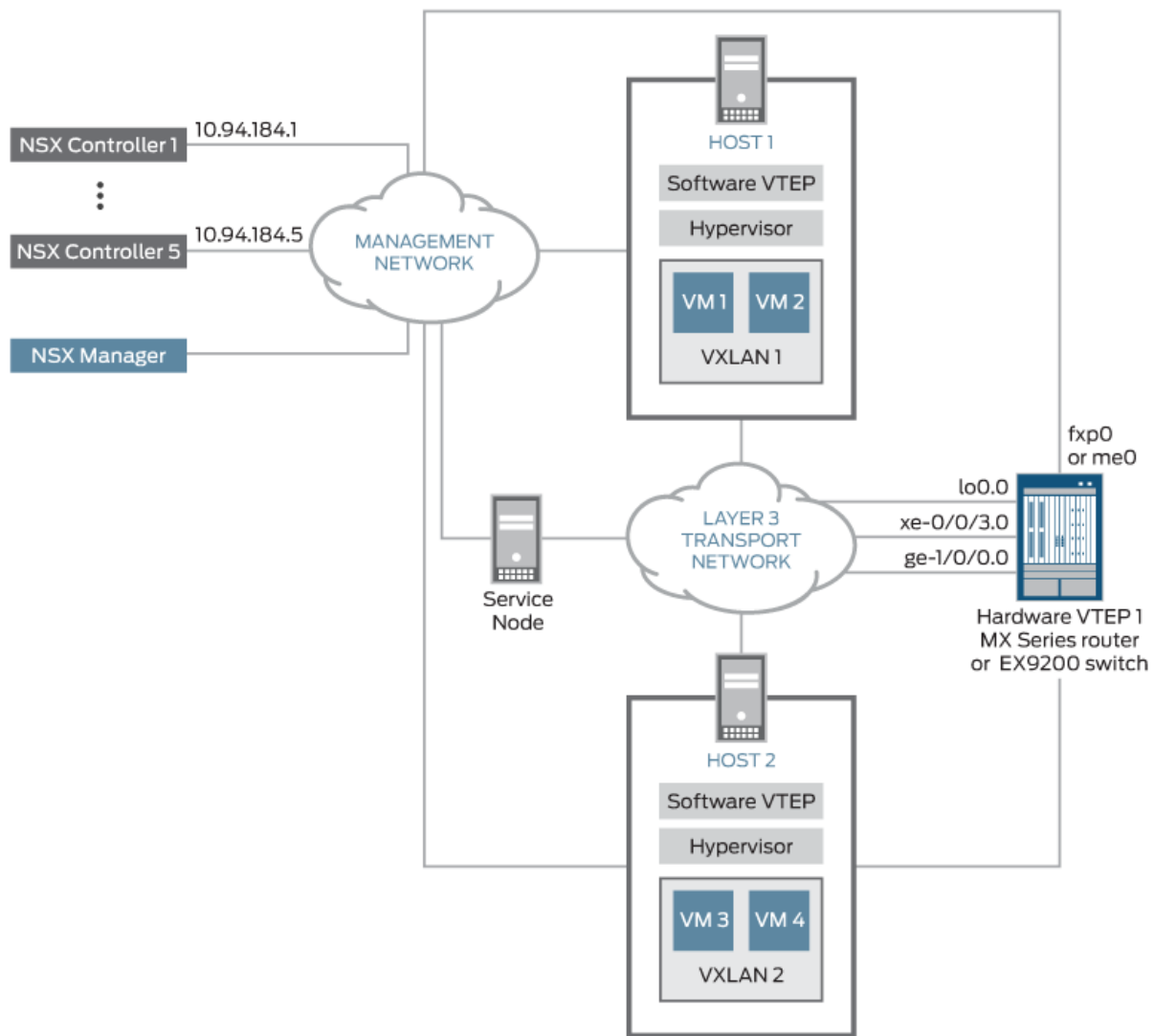
## Overview and Topology

### IN THIS SECTION

- [Topology | 52](#)

In the topology shown in [Figure 4 on page 51](#), VM 1 in VXLAN 1 needs to communicate with VM 3 in VXLAN 2. To enable this communication, hardware VTEP 1, which can be an MX Series router or an EX9200 switch, is configured to route VM traffic between the two VXLANs.

**Figure 4: Inter-VXLAN Unicast and Multicast Routing and OVSDb Topology**



On hardware VTEP 1, a routing instance (virtual switch) is set up. Within the routing instance, two VXLANs are configured: VXLAN 1 and VXLAN 2. Each VXLAN has an integrated routing and bridging (IRB) interface associated with it. The IRB interfaces handle the routing of VM unicast traffic between the VXLANs,

To handle multicast traffic between the VXLANs, each IRB interface is configured as a member of an Internet Group Management Protocol (IGMP) static group, and the MX Series router or EX9200 switch

is configured to function as a PIM rendezvous point (RP) that forwards multicast traffic to each VXLAN through its associated IRB interface.

In this topology, when a multicast packet is received by a VXLAN, for example, VXLAN 1, the following packet handling occurs:

- Within VXLAN 1, the packet is handled as a Layer 2 multicast packet, which means that it is sent to the service node. The service node replicates Layer 2 multicast, as well as Layer 2 broadcast and unknown unicast, packets then forwards the replicas to all interfaces in VXLAN 1. Having the service node handle the Layer 2 BUM traffic is the default behavior, and no configuration is required for the MX Series router or the EX9200 switch.
- The IRB interface associated with VXLAN 1 sends the packet to the PIM RP, which forwards the packet to the IRB associated with VXLAN 2. The IRB interface associated with VXLAN 2 then replicates the packet and forwards the replicas to all hardware and software VTEPs that host VMs, but not to service nodes, in VXLAN 2. The ability of an IRB interface to replicate the Layer 3 multicast packets and forward the replicas to hardware and software VTEPs in a VXLAN is known as *ingress node replication*. This feature is automatically implemented and does not need to be configured.

On hardware VTEP 1, a connection with an NSX controller is configured on the management interface (fxp0 for an MX Series router and me0 for an EX9200 switch). This configuration enables the NSX controller to push MAC routes for VM 1 and VM 3 to the hardware VTEP by way of the table for remote unicast MAC addresses in the OVSDB schema for physical devices.

Each VXLAN-encapsulated packet must include a source IP address, which identifies the source hardware or software VTEP, in the outer IP header. In this example, for hardware VTEP 1, the IP address of the loopback interface (lo0.0) is used.

In this example, the tracing of all OVSDB events is configured. The output of the OVSDB events are placed in a file named **ovsdb**, which is stored in the **/var/log** directory. By default, a maximum of 10 trace files can exist, and the configured maximum size of each file is 50 MB.

## Topology

[Table 10 on page 53](#) describes the components for setting up inter-VXLAN routing and an OVSDB connection.

**Table 10: Components for Setting Up Inter-VXLAN Unicast and Multicast Routing and OVSDDB Connections in a Data Center**

Property	Settings
Routing instance	<p>Name: vx1</p> <p>Type: virtual switch</p> <p>OVSDDB-managed VXLANs included: VXLAN 1 and VXLAN 2</p>
VXLAN 1	<p>Bridge domain or VLAN associated with: 28805c1d-0122-495d-85df-19abd647d772</p> <p>Interface: xe-0/0/2.0</p> <p>VLAN ID: 100</p> <p>VNI: 100</p>
VXLAN 2	<p>Bridge domain or VLAN associated with: 96a382cd-a570-4ac8-a77a-8bb8b16bde70</p> <p>Interface: xe-1/2/0.0</p> <p>VLAN ID: 200</p> <p>VNI: 200</p>
Inter-VXLAN unicast routing and forwarding with IRB interfaces	<p>VXLAN 1: irb.0; 10.20.20.1/24; associated with routing interface vx1, and bridge domain or VLAN 28805c1d-0122-495d-85df-19abd647d772</p> <p>VXLAN 2: irb.1; 10.10.10.3/24; associated with routing interface vx1, and bridge domain or VLAN 96a382cd-a570-4ac8-a77a-8bb8b16bde70</p>

**Table 10: Components for Setting Up Inter-VXLAN Unicast and Multicast Routing and OVSDB Connections in a Data Center *(Continued)***

Property	Settings
Inter-VXLAN multicast routing and forwarding with IRB interfaces	<p>PIM RP: 10.19.19.19</p> <p>VXLAN 1: PIM interface irb.0; IGMP static group 233.252.0.100</p> <p>VXLAN 2: PIM interface irb.1; IGMP static groups 233.252.0.100</p> <p><b>NOTE:</b> On IRB interfaces that forward Layer 3 multicast traffic from one OVSDB-managed VXLAN to another, ingress node replication is automatically implemented; therefore, no configuration is required.</p>
Handling of Layer 2 BUM traffic in each VXLAN	<p>Service node</p> <p><b>NOTE:</b> By default, one or more service nodes handle Layer 2 BUM traffic in a VXLAN; therefore, no configuration is required.</p>
NSX controller	IP address: 10.94.184.1
Hardware VTEP source identifier	<p>Source interface: loopback (lo0.0)</p> <p>Source IP address: 10.19.19.19/32</p>
OVSDB tracing operations	<p>Filename: /var/log/ovsdb</p> <p>File size: 50 MB</p> <p>Flag: All</p>

## Configuration

### IN THIS SECTION

 CLI Quick Configuration | 55

- [Configuring an MX Series Router as a Hardware VTEP with an OVSDDB Connection | 57](#)
- [Configuring an EX9200 Switch as a Hardware VTEP with an OVSDDB Connection | 60](#)

An MX Series router or an EX9200 switch can function as hardware VTEP 1 in this example. Because the configuration for each device is slightly different, a separate configuration is provided for each device.

To configure inter-VXLAN unicast and multicast routing and OVSDDB connections in a data center topology, you need to perform one of these tasks:

### 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 (for example, IP addresses, interface names, and UUIDs), copy and paste the commands into the CLI at the [edit] hierarchy level, and then enter `commit` from configuration mode.

**NOTE:** After completing this configuration, you must configure a gateway, which is the NSX-equivalent of a hardware VTEP. This example implements one hardware VTEP; therefore, you must configure one gateway, a gateway service, and a logical switch port using NSX Manager or the NSX API. For more information about the tasks you must perform and key NSX Manager configuration details, see ["VMware NSX Configuration for Juniper Networks Devices Functioning as Virtual Tunnel Endpoints" on page 29](#).

MX Series router configuration:

```
set chassis network-services enhanced-ip
set interfaces xe-0/0/3 unit 0 family inet address 10.50.50.2/24
set interfaces ge-1/0/0 unit 0 family inet address 10.100.100.99/24
set routing-options router-id 10.19.19.19
set protocols ospf area 0.0.0.0 interface xe-0/0/3.0
set protocols ospf area 0.0.0.0 interface ge-1/0/0.0
set protocols ospf area 0.0.0.0 interface lo0.0
set interfaces xe-0/0/2 unit 0 family bridge interface-mode access
set interfaces xe-0/0/2 unit 0 family bridge vlan-id 100
set interfaces xe-1/2/0 unit 0 family bridge interface-mode access
set interfaces xe-1/2/0 unit 0 family bridge vlan-id 200
```

```

set interfaces irb unit 0 family inet address 10.20.20.1/24
set interfaces irb unit 1 family inet address 10.10.10.3/24
set protocols igmp interface irb.0 static group 233.252.0.100
set protocols igmp interface irb.1 static group 233.252.0.100
set protocols pim rp local address 10.19.19.19
set protocols pim interface irb.0
set protocols pim interface irb.1
set routing-instances vx1 vtep-source-interface lo0.0
set routing-instances vx1 instance-type virtual-switch
set routing-instances vx1 interface xe-0/0/2.0
set routing-instances vx1 interface xe-1/2/0.0
set routing-instances vx1 bridge-domains 28805c1d-0122-495d-85df-19abd647d772 vlan-id 100
set routing-instances vx1 bridge-domains 28805c1d-0122-495d-85df-19abd647d772 routing-interface
irb.0
set routing-instances vx1 bridge-domains 28805c1d-0122-495d-85df-19abd647d772 vxlan ovssdb-managed
set routing-instances vx1 bridge-domains 28805c1d-0122-495d-85df-19abd647d772 vxlan vni 100
set routing-instances vx1 bridge-domains 96a382cd-a570-4ac8-a77a-8bb8b16bde70 vlan-id 200
set routing-instances vx1 bridge-domains 96a382cd-a570-4ac8-a77a-8bb8b16bde70 routing-interface
irb.1
set routing-instances vx1 bridge-domains 96a382cd-a570-4ac8-a77a-8bb8b16bde70 vxlan ovssdb-managed
set routing-instances vx1 bridge-domains 96a382cd-a570-4ac8-a77a-8bb8b16bde70 vxlan vni 200
set interfaces lo0 unit 0 family inet address 10.19.19.19/32 primary
set interfaces lo0 unit 0 family inet address 10.19.19.19/32 preferred
set protocols ovssdb traceoptions file ovssdb
set protocols ovssdb traceoptions file size 50m
set protocols ovssdb traceoptions flag all
set protocols ovssdb controller 10.94.184.1
set protocols ovssdb interfaces xe-0/0/2.0
set protocols ovssdb interfaces xe-1/2/0.0

```

EX9200 switch configuration:

```

set interfaces xe-0/0/3 unit 0 family inet address 10.50.50.2/24
set interfaces ge-1/0/0 unit 0 family inet address 10.100.100.99/24
set routing-options router-id 10.19.19.19
set protocols ospf area 0.0.0.0 interface xe-0/0/3.0
set protocols ospf area 0.0.0.0 interface ge-1/0/0.0
set protocols ospf area 0.0.0.0 interface lo0.0
set interfaces xe-0/0/2 unit 0 family ethernet-switching interface-mode access
set interfaces xe-0/0/2 unit 0 family ethernet-switching vlan-id 100
set interfaces xe-1/2/0 unit 0 family ethernet-switching interface-mode access
set interfaces xe-1/2/0 unit 0 family ethernet-switching vlan-id 200

```



```

set interfaces irb unit 0 family inet address 10.20.20.1/24
set interfaces irb unit 1 family inet address 10.10.10.3/24
set protocols igmp interface irb.0 static group 225.1.1.100
set protocols igmp interface irb.1 static group 225.1.1.100
set protocols pim rp local address 10.19.19.19
set protocols pim interface irb.0
set protocols pim interface irb.1
set routing-instances vx1 vtep-source-interface lo0.0
set routing-instances vx1 instance-type virtual-switch
set routing-instances vx1 interface xe-0/0/2.0
set routing-instances vx1 interface xe-1/2/0.0
set routing-instances vx1 vlans 28805c1d-0122-495d-85df-19abd647d772 vlan-id 100
set routing-instances vx1 vlans 28805c1d-0122-495d-85df-19abd647d772 routing-interface irb.0
set routing-instances vx1 vlans 28805c1d-0122-495d-85df-19abd647d772 vxlan ovsdb-managed
set routing-instances vx1 vlans 28805c1d-0122-495d-85df-19abd647d772 vxlan vni 100
set routing-instances vx1 vlans 96a382cd-a570-4ac8-a77a-8bb8b16bde70 vlan-id 200
set routing-instances vx1 vlans 96a382cd-a570-4ac8-a77a-8bb8b16bde70 routing-interface irb.1
set routing-instances vx1 vlans 96a382cd-a570-4ac8-a77a-8bb8b16bde70 vxlan ovsdb-managed
set routing-instances vx1 vlans 96a382cd-a570-4ac8-a77a-8bb8b16bde70 vxlan vni 200
set interfaces lo0 unit 0 family inet address 10.19.19.19/32 preferred
set interfaces lo0 unit 0 family inet address 10.19.19.19/32 primary
set protocols ovsdb traceoptions file ovsdb
set protocols ovsdb traceoptions file size 50m
set protocols ovsdb traceoptions flag all
set protocols ovsdb controller 10.94.184.1
set protocols ovsdb interfaces xe-0/0/2.0
set protocols ovsdb interfaces xe-1/2/0.0

```

## Configuring an MX Series Router as a Hardware VTEP with an OVSDb Connection

### Step-by-Step Procedure

To configure an MX Series router as hardware VTEP 1 with an OVSDb connection to an NSX controller, follow these steps:

1. Create the Layer 3 network.

```

[edit chassis]
user@router# set network-services enhanced-ip
[edit interfaces]
user@router# set xe-0/0/3 unit 0 family inet address 10.50.50.2/24

```

```

user@router# set ge-1/0/0 unit 0 family inet address 10.100.100.99/24
[edit routing-options]
user@router# set router-id 10.19.19.19
[edit protocols]
user@router# set ospf area 0.0.0.0 interface xe-0/0/3.0
user@router# set ospf area 0.0.0.0 interface ge-1/0/0.0
user@router# set ospf area 0.0.0.0 interface lo0.0

```

2. Create an access interface for VXLAN 1, and associate the interface with the VXLAN.

```

[edit interfaces]
user@router# set xe-0/0/2 unit 0 family bridge interface-mode access
user@router# set xe-0/0/2 unit 0 family bridge vlan-id 100

```

3. Create an access interface for VXLAN 2, and associate the interface with the VXLAN.

```

[edit interfaces]
user@router# set xe-1/2/0 unit 0 family bridge interface-mode access
user@router# set xe-1/2/0 unit 0 family bridge vlan-id 200

```

4. Create an IRB interface to handle inter-VXLAN unicast traffic for VXLAN 1.

```

[edit interfaces]
user@router# set irb unit 0 family inet address 10.20.20.1/24

```

5. Create an IRB interface to handle inter-VXLAN unicast traffic for VXLAN 2.

```

[edit interfaces]
user@router# set irb unit 1 family inet address 10.10.10.3/24

```

6. Configure PIM and IGMP to handle inter-VXLAN multicast traffic.

```

[edit protocols]
user@router# set pim rp local address 10.19.19.19
user@router# set pim interface irb.0
user@router# set pim interface irb.1

```

```

user@router# set igmp interface irb.0 static group 225.1.1.100
user@router# set igmp interface irb.1 static group 225.1.1.100

```

7. Set up the virtual switch routing instance.

```

[edit routing-instances]
user@router# set vx1 vtep-source-interface lo0.0
user@router# set vx1 instance-type virtual-switch
user@router# set vx1 interface xe-0/0/2.0
user@router# set vx1 interface xe-1/2/0.0
user@router# set vx1 bridge-domains 28805c1d-0122-495d-85df-19abd647d772 vlan-id 100
user@router# set vx1 bridge-domains 28805c1d-0122-495d-85df-19abd647d772 routing-interface
irb.0
user@router# set vx1 bridge-domains 28805c1d-0122-495d-85df-19abd647d772 vxlan ovsdb-
managed
user@router# set vx1 bridge-domains 28805c1d-0122-495d-85df-19abd647d772 vxlan vni 100
user@router# set vx1 bridge-domains 96a382cd-a570-4ac8-a77a-8bb8b16bde70 vlan-id 200
user@router# set vx1 bridge-domains 96a382cd-a570-4ac8-a77a-8bb8b16bde70 routing-interface
irb.1
user@router# set vx1 bridge-domains 96a382cd-a570-4ac8-a77a-8bb8b16bde70 vxlan ovsdb-
managed
user@router# set vx1 bridge-domains 96a382cd-a570-4ac8-a77a-8bb8b16bde70 vxlan vni 200

```

8. 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@router# set lo0 unit 0 family inet address 10.19.19.19/32 primary
user@router# set lo0 unit 0 family inet address 10.19.19.19/32 preferred

```

9. Set up OVSDb tracing operations.

```

[edit protocols]
user@router# set ovsdb traceoptions file ovsdb
user@router# set ovsdb traceoptions file size 50m
user@router# set ovsdb traceoptions flag all

```

10. Configure a connection with an NSX controller.

```
[edit protocols]
user@router# set ovssdb controller 10.94.184.1
```

11. Configure interfaces xe-0/0/2.0 and xe-1/2/0.0 to be managed by OVSSDB.

```
[edit protocols]
user@router# set ovssdb interfaces xe-0/0/2.0
user@router# set ovssdb interfaces xe-1/2/0.0
```

**NOTE:** After completing this configuration, you must configure a gateway, which is the NSX-equivalent of a hardware VTEP. This example implements one hardware VTEP; therefore, you must configure one gateway, a gateway service, and a logical switch port by using NSX Manager or the NSX API. For more information about the tasks you must perform and key NSX Manager configuration details, see ["VMware NSX Configuration for Juniper Networks Devices Functioning as Virtual Tunnel Endpoints" on page 29](#).

## Configuring an EX9200 Switch as a Hardware VTEP with an OVSSDB Connection

### Step-by-Step Procedure

To configure an EX9200 switch as hardware VTEP 1 with an OVSSDB connection to an NSX controller, follow these steps:

1. Create the Layer 3 network.

```
[edit chassis]
[edit interfaces]
user@switch# set xe-0/0/3 unit 0 family inet address 10.50.50.2/24
user@switch# set ge-1/0/0 unit 0 family inet address 10.100.100.99/24
[edit routing-options]
user@switch# set router-id 10.19.19.19
[edit protocols]
user@switch# set ospf area 0.0.0.0 interface xe-0/0/3.0
user@switch# set ospf area 0.0.0.0 interface ge-1/0/0.0
user@switch# set ospf area 0.0.0.0 interface lo0.0
```

2. Create an access interface for VXLAN 1, and associate the interface with the VXLAN.

```
[edit interfaces]
user@switch# set xe-0/0/2 unit 0 family ethernet-switching interface-mode access
user@switch# set xe-0/0/2 unit 0 family ethernet-switching vlan-id 100
```

3. Create an access interface for VXLAN 2, and associate the interface with the VXLAN.

```
[edit interfaces]
user@switch# set xe-1/2/0 unit 0 family ethernet-switching interface-mode access
user@switch# set xe-1/2/0 unit 0 family ethernet-switching vlan-id 200
```

4. Create an IRB interface to handle inter-VXLAN unicast traffic for VXLAN 1.

```
[edit interfaces]
user@switch# set irb unit 0 family inet address 10.20.20.1/24
```

5. Create an IRB interface to handle inter-VXLAN unicast traffic for VXLAN 2.

```
[edit interfaces]
user@switch# set irb unit 1 family inet address 10.10.10.3/24
```

6. Configure PIM and IGMP to handle inter-VXLAN multicast traffic.

```
[edit protocols]
user@switch# set pim rp local address 10.19.19.19
user@switch# set pim interface irb.0
user@switch# set pim interface irb.1
user@switch# set igmp interface irb.0 static group 225.1.1.100
user@switch# set igmp interface irb.1 static group 225.1.1.100
```

7. Set up the virtual switch routing instance.

```
[edit routing-instances]
user@switch# set vx1 vtep-source-interface lo0.0
user@switch# set vx1 instance-type virtual-switch
user@switch# set vx1 interface xe-0/0/2.0
```

```

user@switch# set vx1 interface xe-1/2/0.0
user@switch# set vx1 vlans 28805c1d-0122-495d-85df-19abd647d772 vlan-id 100
user@switch# set vx1 vlans 28805c1d-0122-495d-85df-19abd647d772 routing-interface irb.0
user@switch# set vx1 vlans 28805c1d-0122-495d-85df-19abd647d772 vxlan ovssdb-managed
user@switch# set vx1 vlans 28805c1d-0122-495d-85df-19abd647d772 vxlan vni 100
user@switch# set vx1 vlans 96a382cd-a570-4ac8-a77a-8bb8b16bde70 vlan-id 200
user@switch# set vx1 vlans 96a382cd-a570-4ac8-a77a-8bb8b16bde70 routing-interface irb.1
user@switch# set vx1 vlans 96a382cd-a570-4ac8-a77a-8bb8b16bde70 vxlan ovssdb-managed
user@switch# set vx1 vlans 96a382cd-a570-4ac8-a77a-8bb8b16bde70 vxlan vni 200

```

8. 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 10.19.19.19/32 primary
user@switch# set lo0 unit 0 family inet address 10.19.19.19/32 preferred

```

9. Set up tracing operations to be performed for the OVSSDB management protocol.

```

[edit protocols]
user@switch# set ovssdb traceoptions file ovssdb
user@switch# set ovssdb traceoptions file size 50m
user@switch# set ovssdb traceoptions flag all

```

10. Configure a connection with an NSX controller.

```

[edit protocols]
user@switch# set ovssdb controller 10.94.184.1

```

11. Configure interfaces xe-0/0/2.0 and xe-1/2/0.0 to be managed by OVSSDB.

```

[edit protocols]
user@router# set ovssdb interfaces xe-0/0/2.0
user@router# set ovssdb interfaces xe-1/2/0.0

```

**NOTE:** After completing this configuration, you must configure a gateway, which is the NSX-equivalent of a hardware VTEP. This example implements one hardware VTEP; therefore, you must configure one gateway, a gateway service, and a logical switch port by using NSX Manager or the NSX API. For more information about the tasks you must perform and key NSX Manager configuration details, see ["VMware NSX Configuration for Juniper Networks Devices Functioning as Virtual Tunnel Endpoints" on page 29](#).

## Verification

### IN THIS SECTION

- [Verifying the Logical Switches | 63](#)
- [Verifying the MAC Addresses of VM 1 and VM 3 | 64](#)
- [Verifying the NSX Controller Connection | 65](#)

## Verifying the Logical Switches

### Purpose

Verify that logical switches with the UUIDs of 28805c1d-0122-495d-85df-19abd647d772 and 96a382cd-a570-4ac8-a77a-8bb8b16bde70 are configured in NSX Manager or in the NSX API, and that information about the logical switches is published in the OVSDb schema.

### Action

Issue the `show ovssdb logical-switch operational mode` command.

```
user@host> show ovssdb logical-switch
Logical switch information:
Logical Switch Name: 28805c1d-0122-495d-85df-19abd647d772
Flags: Created by both
VNI: 100
Num of Remote MAC: 1
Num of Local MAC: 0
```

```

Logical Switch Name: 96a382cd-a570-4ac8-a77a-8bb8b16bde70
Flags: Created by both
VNI: 200
Num of Remote MAC: 1
Num of Local MAC: 1

```

## Meaning

The output verifies that information about the logical switches is published in the OVSDB schema. The Created by both state indicates that the logical switches are configured in NSX Manager or the NSX API, and the corresponding VXLANs are configured on the Juniper Networks device. In this state, the logical switches and VXLANs are operational.

If the state of the logical switches is something other than Created by both, see ["Troubleshooting a Nonoperational Logical Switch and Corresponding Junos OS OVSDB-Managed VXLAN"](#) on page 188.

## Verifying the MAC Addresses of VM 1 and VM 3

### Purpose

Verify that the MAC addresses of VM 1 and VM 3 are present in the OVSDB schema.

### Action

Issue the `show ovssdb mac remote operational mode` command to verify that the MAC addresses for VM 1 and VM 3 are present.

```

user@host> show ovssdb mac remote
Logical Switch Name: 28805c1d-0122-495d-85df-19abd647d772
  Mac          IP          Encapsulation  Vtep
  Address      Address
  08:33:9d:5f:a7:f1  0.0.0.0      Vxlan over Ipv4  10.19.19.19
Logical Switch Name: 96a382cd-a570-4ac8-a77a-8bb8b16bde70
  Mac          IP          Encapsulation  Vtep
  Address      Address
  a8:59:5e:f6:38:90  0.0.0.0      Vxlan over Ipv4  10.19.19.10

```



## Meaning

The output shows that the MAC addresses for VM 1 and VM 3 are present and are associated with logical switches with the UUIDs of 28805c1d-0122-495d-85df-19abd647d772 and 96a382cd-a570-4ac8-a77a-8bb8b16bde70, respectively. Given that the MAC addresses are present, VM 1 and VM 3 are reachable through hardware VTEP 1.

## Verifying the NSX Controller Connection

### Purpose

Verify that the connection with the NSX controller is up.

### Action

Issue the `show ovssdb controller operational mode` command, and verify that the controller connection state is up.

```
user@host> show ovssdb controller
VTEP controller information:
Controller IP address: 10.94.184.1
Controller protocol: ssl
Controller port: 6632
Controller connection: up
Controller seconds-since-connect: 542325
Controller seconds-since-disconnect: 542346
Controller connection status: active
```

## Meaning

The output shows that the connection state of the NSX controller is up, in addition to other information about the controller. When this connection is up, OVSSDB is enabled on the Juniper Networks device.

## RELATED DOCUMENTATION

[Understanding How Layer 2 BUM and Layer 3 Routed Multicast Traffic Are Handled with OVSSDB | 8](#)

*OVSSDB Schema for Physical Devices*

## Example: Configuring VXLAN to VPLS Stitching with OVSDDB

### IN THIS SECTION

- Requirements | 66
- Overview | 67
- Configuration | 68
- Verification | 78

Virtual Extensible LAN (VXLAN) can be utilized with the Open vSwitch Database (OVSDDB) management protocol in a VPLS-enabled network to stitch a virtualized data center into a Layer 2 VPN network. This configuration allows for seamless interconnection between different data centers using Layer 2 VPN regardless of whether it is virtualized, physical, or both.

### Requirements

This example uses the following hardware and software components:

- Two MX Series routers running Junos OS 14.1R2 or later
- Two MX Series routers running Junos OS 14.1R2 or later with an OVSDDB software package. The release of this package must be the same as the Junos OS release running on the device.
- One EX9200 switch
- One VMware NSX controller
- NSX Manager

Before you start the configuration, you must perform the following tasks:

- In NSX Manager or the NSX API, configure a logical switch for each VXLAN that OVSDDB will manage. This example implements two OVSDDB-managed VXLANs, so you must configure two logical switches. After the configuration of each logical switch, NSX automatically generates a universally unique identifier (UUID) for the logical switch. If you have not done so already, retrieve the UUID for each logical switch. A sample UUID is 28805c1d-0122-495d-85df-19abd647d772. When configuring the equivalent VXLANs on the Juniper Networks device, you must use the UUID of the logical switch as the bridge domain name.

For more information about logical switches and VXLANs, see ["Understanding How to Manually Configure OVSDDB-Managed VXLANs" on page 10](#).

- Create an SSL private key and certificate, and install them in the /var/db/certs directory of the Juniper Networks device. For more information, see ["Creating and Installing an SSL Key and Certificate on a Juniper Networks Device for Connection with SDN Controllers"](#) on page 23.

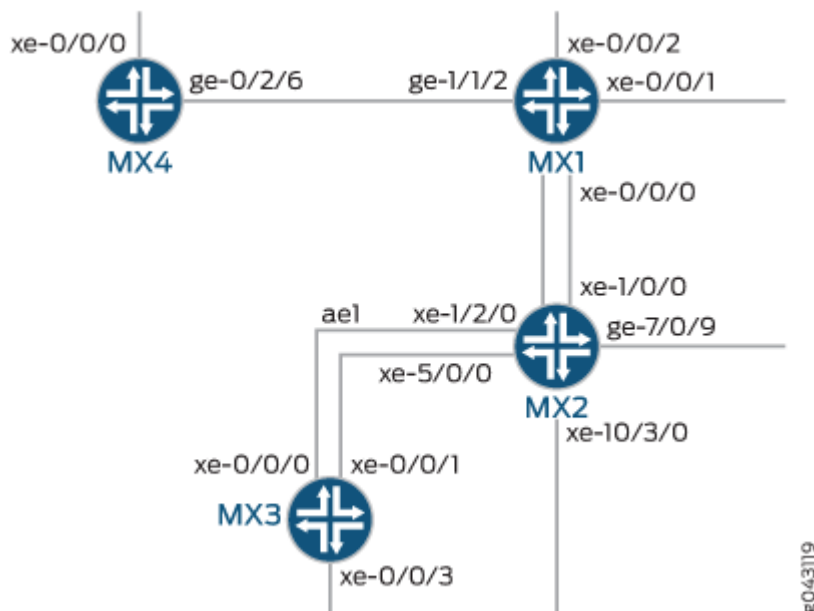
## Overview

### IN THIS SECTION

- [Topology](#) | 67

In this example, four MX Series routers are configured to function together for VXLAN to virtual private LAN service (VPLS) stitching. Each router performs a different role in the configuration. The following diagram shows the topology of these MX Series routers. MX1 is the core router that handles Layer 3 traffic and protocols. MX2 is the VXLAN gateway router that functions as a virtual tunnel endpoint (VTEP) and handles switching for Layer 2, VPLS, and VXLAN. The MX3 router is configured to handle VPLS traffic. The MX4 router is configured as a VTEP to accept and decapsulate VXLAN packets.

### Topology



## Configuration

### IN THIS SECTION

- [CLI Quick Configuration | 68](#)
- [Configuring MX1 | 71](#)
- [Configuring MX2 | 72](#)
- [Configuring MX3 | 74](#)
- [Configuring MX4 | 76](#)
- [Results | 78](#)

To configure VXLAN to VPLS stitching with OVSDDB:

### 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.

#### MX1

```
set interfaces lo0 unit 0 family inet address 10.255.181.13/32 primary
set interfaces ge-1/1/2 unit 0 family inet address 10.30.30.2/30
set interfaces xe-0/0/0 unit 0 family inet address 10.20.20.2/30
set protocols ospf area 0.0.0.0 interface all
```

#### MX2

```
set interfaces lo0 unit 0 family inet address 10.255.181.72/32 primary
set lag-options interfaces <ae*> mtu 9192
set lag-options interfaces <ae*> aggregated ether-options minimum-links 1
set chassis aggregated-devices ethernet device-count 40
set chassis fpc 1 pic 0 tunnel-services bandwidth 10g
set chassis network-services enhanced-ip
set interfaces xe-1/2/0 gigether-options 802.3ad ae1
set interfaces xe-0/0/0 unit 0 family inet address 10.20.20.1/30
set interfaces ge-7/0/9 vlan tagging
```

```

set interfaces ge-7/0/9 unit 1 vlan-id 3
set interfaces ge-7/0/9 unit 1 family vpls
set interfaces xe-10/3/0 vlan tagging
set interfaces xe-10/3/0 unit 1 vlan-id 3
set interfaces xe-10/3/0 unit 0 family vpls
set interfaces ae1 unit 0 family inet address 10.1.1.1/30
set interfaces ae1 unit 0 family mpls
set routing-options autonomous-system 100
set protocols rsvp interface all
set protocols mpls no cspf
set protocols mpls label-switched-path-to-mx3 to 10.255.181.98
set protocols mpls interface all
set protocols bgp family l2vpn signaling
set protocols bgp group ibgp type internal
set protocols bgp group ibgp neighbor 10.255.181.98 local-address 10.255.181.72
set protocols ospf area 0.0.0.0 interface xe-0/0/0.0
set protocols ospf area 0.0.0.0 interface fxp0.0 disable
set protocols ospf area 0.0.0.0 interface lo0.0 passive
set protocols ospf area 0.0.0.0 interface ae1.0
set protocols ovssdb traceoptions file ovssdb.log size 100m files 10
set protocols ovssdb traceoptions file ovssdb.level all
set protocols ovssdb traceoptions file ovssdb.flag all
set protocols ovssdb interfaces xe-10/3/0.1
set protocols ovssdb interfaces ge-7/0/9.1
set protocols ovssdb controller 192.168.182.45 protocol ssl port 6632
set routing-instances 24a76aff-7e61-4520-a78d-3eca26ad7510 vtep-source-interface lo0.0
set routing-instances 24a76aff-7e61-4520-a78d-3eca26ad7510 instance-type vpls
set routing-instances 24a76aff-7e61-4520-a78d-3eca26ad7510 vlan-id 3
set routing-instances 24a76aff-7e61-4520-a78d-3eca26ad7510 interface ge-7/0/9.1
set routing-instances 24a76aff-7e61-4520-a78d-3eca26ad7510 interface xe-10/3/0.1
set routing-instances 24a76aff-7e61-4520-a78d-3eca26ad7510 routing-interface irb.3
set routing-instances 24a76aff-7e61-4520-a78d-3eca26ad7510 vxlan ovssdb-managed
set routing-instances 24a76aff-7e61-4520-a78d-3eca26ad7510 vxlan vni 3
set routing-instances 24a76aff-7e61-4520-a78d-3eca26ad7510 route-distinguisher 10.255.181.72:3
set routing-instances 24a76aff-7e61-4520-a78d-3eca26ad7510 vrf-target target:3:3
set routing-instances 24a76aff-7e61-4520-a78d-3eca26ad7510 protocols vpls site mx2 site-
identifier 1

```

### MX3

```

set interfaces lo0 unit 0 family inet address 127.0.0.1/32
set interfaces lo0 unit 0 family inet address 10.255.181.98/32 primary

```

```

set lag-options interfaces <ae*> mtu 9192
set lag-options interfaces <ae*> aggregated-ether-options minimum-links 1
set lag-options interfaces <ae*> aggregated-ether-options lacp active
set interfaces xe-0/0/0 gigether-options 802.3ad ae1
set interfaces xe-0/0/1 gigether-options 802.3ad ae1
set interfaces xe-0/0/3 vlan tagging
set interfaces xe-0/0/3 unit 1 vlan-id 3
set interfaces xe-0/0/3 unit 1 family vpls
set interfaces ae1 unit 0 family inet address 10.1.1.2/30
set interfaces ae1 unit 0 family mpls
set routing-options autonomous-system 100
set protocols rsvp interface all
set protocols mpls no-cspf
set protocols mpls label-switched-path-to-mx2 to 10.255.181.72
set protocols mpls interface all
set protocols bgp family l2vpn signaling
set protocols bgp group ibgp type internal
set protocols bgp group ibgp neighbor 10.255.181.72 local-address 10.255.181.98
set protocols ospf area 0.0.0.0 interface lo0.0 passive
set protocols ospf area 0.0.0.0 interface ae1.0
set routing-instances vpls3 instance-type vpls
set routing-instances vpls3 vlan-id 3
set routing-instances vpls3 interface xe-0/0/3.1
set routing-instances vpls3 route-distinguisher 10.255.181.98:3
set routing-instances vpls3 vrf-target target:3:3
set routing-instances vpls3 protocols vpls no-tunnel-services
set routing-instances vpls3 protocols vpls site mx3 site-identifier 2

```

## MX4

```

set interfaces lo0 unit 0 family inet address 10.255.181.43/32 primary
set interfaces xe-0/0/0 vlan-tagging
set interfaces xe-0/0/0 encapsulation flexible-ethernet-services
set interfaces xe-0/0/0 unit 0 family bridge interface-mode trunk
set interfaces xe-0/0/0 unit 0 family bridge vlan-id-list 1-10
set interfaces ge-0/2/6 unit 0 family inet address 10.30.30.1/30
set protocols ospf area 0.0.0.0 interface ge-0/2/6.0
set protocols ospf area 0.0.0.0 interface fxp0.0 disable
set protocols ospf area 0.0.0.0 interface lo0.0 passive
set protocols ovssdb traceoptions file ovssdb.log size 100m files 10
set protocols ovssdb traceoptions level all
set protocols ovssdb traceoptions flag all

```

```

set protocols ovssdb interfaces xe-0/0/0.0
set protocols ovssdb controller 192.168.182.45 protocol ssl port 6632
set routing-instances default-vs1 vtep-source-interface lo0.0
set routing-instances default-vs1 instance-type virtual-switch
set routing-instances default-vs1 interface xe-0/0/0.1
set bridge-domains 24a76aff-7e61-4520-a78d-3eca26ad7510 vlan-id 3
set bridge-domains 24a76aff-7e61-4520-a78d-3eca26ad7510 vxlan ovssdb-managed
set bridge-domains 24a76aff-7e61-4520-a78d-3eca26ad7510 vxlan vni 3
set bridge-domains 24a76aff-7e61-4520-a78d-3eca26ad7510 vxlan ingress-node-replication
set switch-options vtep-source-interface lo0.0

```

## Configuring MX1

### Step-by-Step Procedure

The first router to be configured is the core router. This MX Series router handles Layer 3 traffic and protocols for the rest of the network.

To configure the MX1 router:

1. Specify the IPv4 address for the loopback interface.

```

[edit interfaces]
user@MX1# set lo0 unit 0 family inet address 10.255.181.13/32 primary

```

2. Configure the Layer 3 network.

```

[edit interfaces]
user@MX1# set ge-1/1/2 unit 0 family inet address 10.30.30.2/30
user@MX1# set xe-0/0/0 unit 0 family inet address 10.20.20.2/30

```

3. Enable OSPF on all interfaces.

```

[edit protocols]
user@MX1# set ospf area 0.0.0.0 interface all

```

## Configuring MX2

### Step-by-Step Procedure

The second router to be configured is the VXLAN gateway router. This MX Series router is configured as a VTEP, and it handles switching for Layer 2, VPLS, and VXLAN.

To configure the MX2 router:

1. Configure interfaces for the VXLAN gateway.

```
[edit interfaces]
user@MX2# set lo0 unit 0 family inet address 10.255.181.72/32 primary
user@MX2# set xe-1/2/0 gigether-options 802.3ad ae1
user@MX2# set xe-0/0/0 unit 0 family inet address 10.20.20.1/30
user@MX2# set ge-7/0/9 vlan tagging
user@MX2# set ge-7/0/9 unit 1 vlan-id 3
user@MX2# set ge-7/0/9 unit 1 family vpls
user@MX2# set xe-10/3/0 vlan tagging
user@MX2# set xe-10/3/0 unit 1 vlan-id 3
user@MX2# set xe-10/3/0 unit 0 family vpls
user@MX2# set ae1 unit 0 family inet address 10.1.1.1/30
user@MX2# set ae1 unit 0 family mpls
```

2. Set up LAG options

```
[edit lag-options]
user@MX2# set interfaces <ae*> mtu 9192
user@MX2# set interfaces <ae*> aggregated ether-options minimum-links 1
```

3. Configure chassis settings.

```
[edit chassis]
user@MX2# set aggregated-devices ethernet device-count 40
user@MX2# set fpc 1 pic 0 tunnel-services bandwidth 10g
user@MX2# set network-services enhanced-ip
```



4. Configure routing options.

```
[edit routing-options]
user@MX2# set autonomous-system 100
```

5. Set up RSVP, MPLS, and BGP protocols.

```
[edit protocols]
user@MX2# set rsvp interface all
user@MX2# set mpls no cspf
user@MX2# set mpls label-switched-path-to-mx3 to 10.255.181.98
user@MX2# set mpls interface all
user@MX2# set bgp family l2vpn signaling
user@MX2# set bgp group ibgp type internal
user@MX2# set bgp group ibgp neighbor 10.255.181.98 local-address 10.255.181.72
```

6. Configure OSPF interface settings.

```
[edit protocols]
user@MX2# set ospf area 0.0.0.0 interface xe-0/0/0.0
user@MX2# set ospf area 0.0.0.0 interface fxp0.0 disable
user@MX2# set ospf area 0.0.0.0 interface lo0.0 passive
user@MX2# set ospf area 0.0.0.0 interface ae1.0
```

7. Set up OVSDb tracing operations.

```
[edit protocols]
user@MX2# set ovssdb traceoptions file ovssdb.log size 100m files 10
user@MX2# set ovssdb traceoptions file ovssdb.level all
user@MX2# set ovssdb traceoptions file ovssdb.flag all
```

8. Specify that interfaces xe-10/3/0.1 and ge-7/0/9.1 are managed by OVSDb.

```
[edit protocols]
user@MX2# set ovssdb interfaces xe-10/3/0.1
user@MX2# set ovssdb interfaces ge-7/0/9.1
```

## 9. Configure a connection with an NSX controller.

```
[edit protocols]
user@MX2# set ovssdb controller 192.168.182.45 protocol ssl port 6632
```

## 10. Create a VPLS routing instance with VXLAN functionality.

```
[edit routing-instances]
user@MX2# set 24a76aff-7e61-4520-a78d-3eca26ad7510 vtep-source-interface lo0.0
user@MX2# set 24a76aff-7e61-4520-a78d-3eca26ad7510 instance-type vpls
user@MX2# set 24a76aff-7e61-4520-a78d-3eca26ad7510 vlan-id 3
user@MX2# set 24a76aff-7e61-4520-a78d-3eca26ad7510 interface ge-7/0/9.1
user@MX2# set 24a76aff-7e61-4520-a78d-3eca26ad7510 interface xe-10/3/0.1
user@MX2# set 24a76aff-7e61-4520-a78d-3eca26ad7510 routing-interface irb.3
user@MX2# set 24a76aff-7e61-4520-a78d-3eca26ad7510 vxlan ovssdb-managed
user@MX2# set 24a76aff-7e61-4520-a78d-3eca26ad7510 vxlan vni 3
user@MX2# set 24a76aff-7e61-4520-a78d-3eca26ad7510 route-distinguisher 10.255.181.72:3
user@MX2# set 24a76aff-7e61-4520-a78d-3eca26ad7510 vrf-target target:3:3
user@MX2# set 24a76aff-7e61-4520-a78d-3eca26ad7510 protocols vpls site mx2 site-identifier 1
```

**NOTE:** After completing this configuration, you must configure a gateway, which is the NSX equivalent of a hardware VTEP. This configuration implements one hardware VTEP, so you must configure one gateway, a gateway service, and a logical switch port using NSX Manager or the NSX API. For more information about the tasks you must perform as well as key NSX Manager configuration details, see ["VMware NSX Configuration for Juniper Networks Devices Functioning as Virtual Tunnel Endpoints" on page 29](#).

## Configuring MX3

### Step-by-Step Procedure

The third MX Series router must be configured to handle VPLS traffic.

To configure the MX3 router:

1. Specify the IPv4, IPv6, and ISO addresses for the loopback interface.

```
[edit interfaces]
user@MX3# set lo0 unit 0 family inet address 127.0.0.1/32
user@MX3# set lo0 unit 0 family inet address 10.255.181.98/32 primary
```

2. Configure the network interfaces.

```
[edit interfaces]
user@MX3# set xe-0/0/0 gigether-options 802.3ad ae1
user@MX3# set xe-0/0/1 gigether-options 802.3ad ae1
user@MX3# set xe-0/0/3 vlan tagging
user@MX3# set xe-0/0/3 unit 1 vlan-id 3
user@MX3# set xe-0/0/3 unit 1 family vpls
user@MX3# set ae1 unit 0 family inet address 10.1.1.2/30
user@MX3# set ae1 unit 0 family mpls
```

3. Set up LAG options

```
[edit lag-options]
user@MX3# set interfaces <ae*> mtu 9192
user@MX3# set interfaces <ae*> aggregated-ether-options minimum-links 1
user@MX3# set interfaces <ae*> aggregated-ether-options lacp active
```

4. Configure routing options.

```
[edit routing-options]
user@MX3# set autonomous-system 100
```

5. Set up RSVP, MPLS, and BGP protocols.

```
[edit protocols]
user@MX3# set rsvp interface all
user@MX3# set mpls no cspf
user@MX3# set mpls label-switched-path-to-mx2 to 10.255.181.72
user@MX3# set mpls interface all
user@MX3# set bgp family l2vpn signaling
```

```

user@MX3# set bgp group ibgp type internal
user@MX3# set bgp group ibgp neighbor 10.255.181.72 local-address 10.255.181.98

```

## 6. Configure OSPF interface settings.

```

[edit protocols]
user@MX3# set ospf area 0.0.0.0 interface lo0.0 passive
user@MX3# set ospf area 0.0.0.0 interface ae1.0

```

## 7. Create a VPLS routing instance.

```

[edit routing-instances]
set vpls3 instance-type vpls
set vpls3 vlan-id 3
set vpls3 interface xe-0/0/3.1
set vpls3 route-distinguisher 10.255.181.98:3
set vpls3 vrf-target target:3:3
set vpls3 protocols vpls no-tunnel-services
set vpls3 protocols vpls site mx3 site-identifier 2

```

## Configuring MX4

### Step-by-Step Procedure

The fourth MX Series router is configured as a VTEP to accept and decapsulate VXLAN packets.

To configure the MX4 router:

1. Specify the IPv4, IPv6, and ISO addresses for the loopback interface.

```

[edit interfaces]
user@MX4# set lo0 unit 0 family inet address 10.255.181.43/32 primary

```

2. Configure the interfaces.

```

[edit interfaces]
user@MX4# set xe-0/0/0 vlan-tagging
user@MX4# set xe-0/0/0 encapsulation flexible-ethernet-services
user@MX4# set xe-0/0/0 unit 0 family bridge interface-mode trunk

```

```
user@MX4# set xe-0/0/0 unit 0 family bridge vlan-id-list 1-10
user@MX4# set ge-0/2/6 unit 0 family inet address 10.30.30.1/30
```

### 3. Configure OSPF interface settings.

```
[edit protocols]
user@MX4# set ospf area 0.0.0.0 interface ge-0/2/6.0
user@MX4# set ospf area 0.0.0.0 interface fxp0.0 disable
user@MX4# set ospf area 0.0.0.0 interface lo0.0 passive
```

### 4. Set up OVSDb tracing operations.

```
[edit protocols]
user@MX4# set ovssdb traceoptions file ovssdb.log size 100m files 10
user@MX4# set ovssdb traceoptions level all
user@MX4# set ovssdb traceoptions flag all
```

### 5. Specify that the xe-0/0/0.0 interface is managed by OVSDb.

```
[edit protocols]
user@MX4# set ovssdb interfaces xe-0/0/0.0
```

### 6. Configure a connection with an NSX controller.

```
[edit protocols]
user@MX4# set ovssdb controller 192.168.182.45 protocol ssl port 6632
```

### 7. Configure the VPLS interface.

```
[edit routing-instances]
user@MX4# set default-vs1 vtep-source-interface lo0.0
user@MX4# set default-vs1 instance-type virtual-switch
user@MX4# set default-vs1 interface xe-0/0/0.1
```

## 8. Configure a set of VXLAN-enabled bridge domains.

```
[edit bridge-domains]
user@MX4# set 24a76aff-7e61-4520-a78d-3eca26ad7510 vlan-id 3
user@MX4# set 24a76aff-7e61-4520-a78d-3eca26ad7510 vxlan ovssdb-managed
user@MX4# set 24a76aff-7e61-4520-a78d-3eca26ad7510 vxlan vni 3
user@MX4# set 24a76aff-7e61-4520-a78d-3eca26ad7510 vxlan ingress-node-replication
```

## 9. Configure the loopback interface to be used as the tunnel source address.

```
[edit switch-options]
user@MX4# set vtep-source-interface lo0.0
```

**NOTE:** After completing this configuration, you must configure a gateway, which is the NSX equivalent of a hardware VTEP. This configuration implements one hardware VTEP, so you must configure one gateway, a gateway service, and a logical switch port using NSX Manager or the NSX API. For more information about the tasks you must perform as well as key NSX Manager configuration details, see ["VMware NSX Configuration for Juniper Networks Devices Functioning as Virtual Tunnel Endpoints" on page 29](#).

## Results

From configuration mode, confirm your configuration by entering the following commands on each router. If the output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

## Verification

### IN THIS SECTION

- [Verifying MX1 | 79](#)
- [Verifying MX2 | 80](#)
- [Verifying MX3 | 89](#)
- [Verifying MX4 | 92](#)

Confirm that the configuration is working properly.

## Verifying MX1

### Purpose

Verify your configuration on MX1.

### Action

Verify that the interfaces are configured properly.

```
user@MX1# show interface
```

```
lo0 {
  unit 0 {
    family inet {
      address 10.255.181.13/32 {
        primary;
      }
    }
  }
}
ge-1/1/2 {
  unit 0 {
    family inet {
      address 10.30.30.2/30;
    }
  }
}
xe-0/0/0 {
  unit 0 {
    family inet {
      address 10.20.20.2/30;
    }
  }
}
```

Verify that OSPF is configured correctly.

```
user@MX1# show protocols
```

```
ospf {  
  area 0.0.0.0 {  
    interface all;  
  }  
}
```

## Verifying MX2

### Purpose

Verify your configuration on MX2.

### Action

Verify that the interfaces are configured properly.

```
user@MX2# show interfaces
```

```
lo0 {  
  unit 0 {  
    family inet {  
      address 10.255.181.72/32 {  
        primary;  
      }  
    }  
  }  
}  
xe-1/2/0 {  
  gigether-options {  
    802.3ad ae1;  
  }  
}  
xe-0/0/0 {  
  unit 0 {
```



```

        family inet {
            address 10.20.20.1/30;
        }
    }
}
ge-7/0/9 {
    vlan-tagging;
    unit 0 {
        family vpls;
    }
    unit 1 {
        vlan-id 3;
    }
}
xe-10/3/0 {
    vlan-tagging;
    unit 0 {
        family vpls;
    }
    unit 1 {
        vlan-id 3;
    }
}
ae1 {
    unit 0 {
        family inet {
            address 10.1.1.1/30;
        }
        family mpls;
    }
}

```

Verify that OSPF is configured properly.

```
user@MX2# show protocols ospf
```

```

ospf {
  area 0.0.0.0 {
    interface xe-0/0/0.0;
    interface fxp0.0 {
      disable;
    }
  }
}

```

```

    }
    interface lo0.0 {
        passive;
    }
    interface ae1.0;
}
}

```

Verify that OVSDb is configured properly.

```
user@MX2# show protocols ovbdb
```

```

ovbdb {
  traceoptions {
    file ovbdb.log size 100m files 10;
    level all;
    flag all;
  }
  interfaces {
    xe-10/3/0.1;
    ge-7/0/9.0;
    ge-7/0/9.1;
  }
  controller 192.168.182.45 {
    protocol {
      ssl port 6632;
    }
  }
}

```

Verify the default-VS1 routing instance configuration.

```
user@MX2# show routing-instances
```

```

routing-instances {
  24a76aff-7e61-4520-a78d-3eca26ad7510 {
    vtep-source-interface lo0.0;
    instance-type vpls;
    vlan-id 3;
  }
}

```

```

interface ge-7/0/9.1;
interface xe-10/3/0.1;
routing-interface irb.3;
vxlan {
    ovssdb-managed;
    vni 3;
    encapsulate-inner-vlan;
    decapsulate-accept-inner-vlan;
    ingress-node-replication;
}
route-distinguisher 10.255.181.72:3;
vrf-target target:3:3;
protocols {
    vpls {
        traceoptions {
            file vpls.log;
            flag all;
        }
        site MX2 {
            site-identifier 1;
        }
    }
}
}
cadbc185-f60f-48a6-93fd-dc14a6420c60 {
    vtep-source-interface lo0.0;
    instance-type vpls;
    vlan-id 2;
    interface ge-7/0/9.0;
    interface xe-10/3/0.0;
    routing-interface irb.2;
    vxlan {
        ovssdb-managed;
        vni 2;
        encapsulate-inner-vlan;
        decapsulate-accept-inner-vlan;
        ingress-node-replication;
    }
    route-distinguisher 10.255.181.72:10;
    vrf-target target:10:10;
    protocols {
        vpls {
            traceoptions {

```

```

        file vpls.log;
        flag all;
    }
    site MX2 {
        site-identifier 1;
    }
}
}
}
vpls11 {
    vtep-source-interface lo0.1;
    instance-type vpls;
    vlan-id 11;
    interface ge-7/0/9.11;
    interface xe-10/3/0.11;
    routing-interface irb.11;
    vxlan {
        ovsdb-managed;
        vni 11;
        ingress-node-replication;
    }
    route-distinguisher 10.255.181.72:11;
    vrf-target target:11:11;
    protocols {
        vpls {
            traceoptions {
                file vpls.log;
                flag all;
            }
            site MX2 {
                site-identifier 1;
            }
        }
    }
}
vpls12 {
    vtep-source-interface lo0.1;
    instance-type vpls;
    vlan-id 12;
    interface ge-7/0/9.12;
    interface xe-10/3/0.12;
    routing-interface irb.12;
    vxlan {

```

```

        ovssdb-managed;
        vni 12;
        ingress-node-replication;
    }
    route-distinguisher 10.255.181.72:12;
    vrf-target target:12:12;
    protocols {
        vpls {
            traceoptions {
                file vpls.log;
                flag all;
            }
            site mx2 {
                site-identifier 1;
            }
        }
    }
}
vpls13 {
    vtep-source-interface lo0.1;
    instance-type vpls;
    vlan-id 13;
    interface ge-7/0/9.13;
    interface xe-10/3/0.13;
    routing-interface irb.13;
    vxlan {
        vni 13;
        multicast-group 233.252.0.13;
    }
    route-distinguisher 10.255.181.72:13;
    vrf-target target:13:13;
    protocols {
        vpls {
            traceoptions {
                file vpls.log;
                flag all;
            }
            site mx2 {
                site-identifier 1;
            }
        }
    }
}
}

```

```

vpls14 {
    vtep-source-interface lo0.1;
    instance-type vpls;
    vlan-id 14;
    interface ge-7/0/9.14;
    interface xe-10/3/0.14;
    routing-interface irb.14;
    vxlan {
        vni 14;
        multicast-group 233.252.0.14;
    }
    route-distinguisher 10.255.181.72:14;
    vrf-target target:14:14;
    protocols {
        vpls {
            traceoptions {
                file vpls.log;
                flag all;
            }
            site mx2 {
                site-identifier 1;
            }
        }
    }
}

vpls15 {
    vtep-source-interface lo0.1;
    instance-type vpls;
    vlan-id 15;
    interface ge-7/0/9.15;
    interface xe-10/3/0.15;
    routing-interface irb.15;
    vxlan {
        vni 15;
        multicast-group 233.252.0.15;
    }
    route-distinguisher 10.255.181.72:15;
    vrf-target target:15:15;
    protocols {
        vpls {
            traceoptions {
                file vpls.log;
                flag all;
            }
        }
    }
}

```

```

        }
        site mx2 {
            site-identifier 1;
        }
    }
}

vpls4 {
    vtep-source-interface lo0.0;
    instance-type vpls;
    vlan-id 4;
    interface ge-7/0/9.4;
    interface xe-10/3/0.4;
    routing-interface irb.4;
    vxlan {
        vni 4;
        multicast-group 233.252.0.4;
    }
    route-distinguisher 10.255.181.72:4;
    vrf-target target:4:4;
    protocols {
        vpls {
            traceoptions {
                file vpls.log;
                flag all;
            }
            site mx2 {
                site-identifier 1;
            }
        }
    }
}

vpls5 {
    vtep-source-interface lo0.0;
    instance-type vpls;
    vlan-id 5;
    interface ge-7/0/9.5;
    interface xe-10/3/0.5;
    routing-interface irb.5;
    vxlan {
        vni 5;
        multicast-group 233.252.0.5;
    }
}

```

```

route-distinguisher 10.255.181.72:5;
vrf-target target:5:5;
protocols {
    vpls {
        traceoptions {
            file vpls.log;
            flag all;
        }
        site mx2 {
            site-identifier 1;
        }
    }
}
vrf1 {
    instance-type vrf;
    interface ae2.0;
    interface lo0.1;
    route-distinguisher 100:100;
    vrf-target target:100:100;
    protocols {
        ospf {
            area 0.0.0.0 {
                interface ae2.0;
                interface lo0.1 {
                    passive;
                }
            }
        }
        pim {
            rp {
                static {
                    address 10.255.181.13;
                }
            }
            interface all;
        }
    }
}
}

```



Verify the vrf1 routing instance configuration.

```
user@MX2# show routing-instances
```

```
24a76aff-7e61-4520-a78d-3eca26ad7510 {
  vtep-source-interface lo0.0;
  instance-type vpls;
  vlan-id 3;
  interface ge-7/0/9.1;
  interface xe-10/3/0.1;
  routing-interface irb.3;
  vxlan {
    ovssdb-managed;
    vni 3;
  }
  route-distinguisher 10.255.181.72:3;
  vrf-target target:3:3;
  protocols {
    vpls {
      site mx2 {
        site-identifier 1;
      }
    }
  }
}
```

## Verifying MX3

### Purpose

Verify your configuration on MX3.

## Action

Verify that the interfaces are configured properly.

```
user@MX3# show interfaces
```

```
xe-0/0/0 {  
  gigether-options {  
    802.3ad ae1;  
  }  
}  
xe-0/0/1 {  
  gigether-options {  
    802.3ad ae1;  
  }  
}  
xe-0/0/3 {  
  vlan-tagging;  
  unit 1 {  
    vlan-id 3;  
    family vpls;  
  }  
}  
ae1 {  
  unit 0 {  
    family inet {  
      address 10.1.1.2/30;  
    }  
    family mpls;  
  }  
}
```

Verify the RSVP, MPLS, BGP and OSPF protocols are configured properly.

```
user@MX3# show protocols
```

```
protocols {  
  rsvp {  
    interface all;
```

```

}
mpls {
    no-cspf;
    label-switched-path to-mx2 {
        to 10.255.181.72;
    }
    interface all;
}
bgp {
    family l2vpn {
        signaling;
    }
    group ibgp {
        type internal;
        neighbor 10.255.181.72 {
            local-address 10.255.181.98;
        }
    }
}
ospf {
    area 0.0.0.0 {
        interface lo0.0 {
            passive;
        }
        interface ae1.0;
    }
}
}

```

Verify the VPLS routing instance configuration.

```
user@MX3# show routing-instances
```

```

routing-instances {
    vpls3 {
        instance-type vpls;
        vlan-id 3;
        interface xe-0/0/3.1;
        route-distinguisher 10.255.181.98:3;
        vrf-target target:3:3;
        protocols {

```

```

        vpls {
            no-tunnel-services;
            site mx3 {
                site-identifier 2;
            }
        }
    }
}

```

## Verifying MX4

### Purpose

Verify your configuration on MX4.

### Action

Verify that the global group interfaces are configured properly.

```
user@MX4# show groups global interfaces
```

Verify that the interfaces are configured properly.

```
user@MX4# show interfaces
```

```

lo0 {
    unit 0 {
        family inet {
            address 10.255.181.43/32 {
                primary;
            }
        }
    }
}
xe-0/0/0 {
    vlan-tagging;
}

```

```

encapsulation flexible-ethernet-services;
unit 0 {
    family bridge {
        interface-mode trunk;
        vlan-id-list 1-10;
    }
}
ge-0/2/6 {
    unit 0 {
        family inet {
            address 30.30.30.1/30;
        }
    }
}

```

Verify that the OSPF interface settings are configured properly.

```

user@MX4# show protocols ospf
    area 0.0.0.0 {
        interface ge-0/2/6.0;
        interface fxp0.0 {
            disable;
        }
        interface lo0.0 {
            passive;
        }
    }

```

Verify that OVSDB is configured properly.

```

user@MX4# show protocols ovsdb
    traceoptions {
        file ovsdb.log size 100m files 10;
        level all;
        flag all;
    }
    interfaces {
        xe-0/0/0.0;
    }
    controller 192.168.182.45 {

```

```

        protocol {
            ssl port 6632;
        }
    }
}

```

Verify the default-VS1 routing instance configuration and bridge domains.

```

user@MX4# show routing-instances default-VS1
    vtep-source-interface lo0.0;
    instance-type virtual-switch;
    interface xe-0/0/0.1;

```

Verify that the bridge domains are configured properly.

```

user@MX4# show bridge-domains
    24a76aff-7e61-4520-a78d-3eca26ad7510 {
        vlan-id 3;
        vxlan {
            ovsdb-managed;
            vni 3;
            ingress-node-replication;
        }
    }
}

```

Verify that the loopback interface is used as the tunnel source address.

```

user@MX4# show switch-options
    vtep-source-interface lo0.0;

```

## RELATED DOCUMENTATION

[Understanding How to Manually Configure OVSDB-Managed VXLANs | 10](#)

[Creating and Installing an SSL Key and Certificate on a Juniper Networks Device for Connection with SDN Controllers | 23](#)

## Example: Configuring Inter-VXLAN Traffic Routing from One Bridge Domain to Another Using an MX Series Router as a Layer 3 Gateway

### IN THIS SECTION

- [Requirements | 95](#)
- [Overview and Topology | 95](#)
- [Configuration | 98](#)
- [Verification | 102](#)

You can configure an MX Series router to act as a Layer 3 gateway to route traffic in a Virtual Extensible LAN (VXLAN) domain managed by an Open vSwitch Database (OVSDb) controller such as a VMware NSX controller. Using this configuration, you can route traffic from one bridge domain to another.

### Requirements

This example uses the following hardware and software components:

- An MX Series router
- Junos OS Release 17.2R1 or later
- A VMware vSphere Distributed Switch (VDS)
- Five virtual machines (VMs)
- An NSX controller
- An NSX manager
- Three servers

### Overview and Topology

#### IN THIS SECTION

- [Topology | 97](#)

Figure 5 on page 97 shows a data center topology is shown in . A VDS provides the centralized interface for configuring, monitoring and administering virtual machine access switching. A VDS is a software switch present in the NSX compute node. Logical switches represent VXLAN in the NSX-V solution. Two logical switches are connected to this VDS. Each logical switch has its own bridge domain in the MX Series router. The logical switches support five VMs. Logical Switch 5000 supports VM4 and VM 5. Logical Switch 5001 supports VM1, VM2, and VM3. Each logical switch has its own bridge domain. Two integrated routing and bridging (IRB) interfaces are associated with these bridge domains and assist in the transfer of packets from one bridge domain to the other through the MX Series router, acting as the Layer 3 gateway. Logical Switch 5000 uses `irb.1`, and Logical Switch 5001 uses `irb.2`.

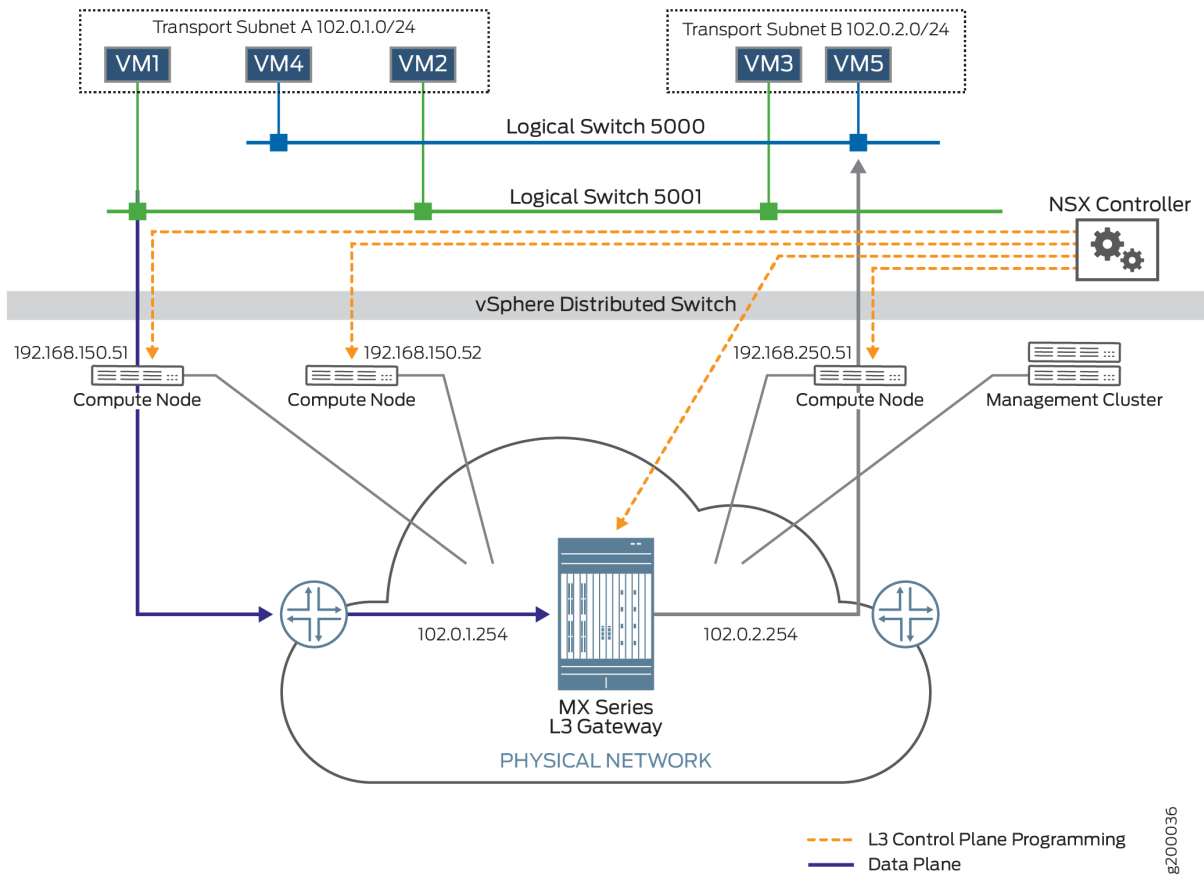
There are three servers in this data center:

- Server 192.168.150.51 supports VM1 and VM4
- Server 192.168.150.52 supports VM2
- Server 192.168.250.51 supports VM3 and VM5

The management cluster contains an NSX controller and an NSX manager. The NSX controller maintains the runtime space and distributes information to the compute nodes. When a VM is brought up on a compute node, the compute node sends information about the VM, such as MAC and IP addresses, to the NSX controller. The NSX controller then pushes this information to all the servers. The NSX manager handles the management plane, supporting the API and the configuration. The NSX manager provisions and manages the network, network services, and VXLAN preparation.

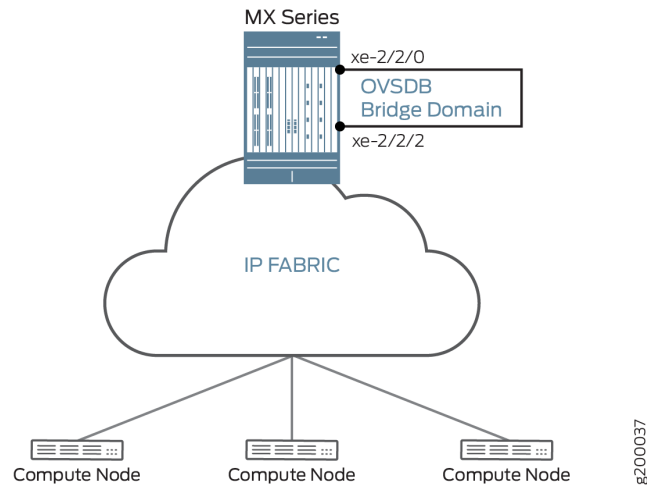


## Topology



**Figure 5: MX Series Router Acting as a Layer 3 Gateway**

In [Figure 6 on page 98](#), a Layer 2 port is created and looped on the MX Series router. This is done in order to bring up the IRB interfaces that map the MX Series router to the NSX controller.



**Figure 6: L2 Port is Created and Looped on the MX router**

In this example, VM1 will send packets to VM5 using the MX Series router to do the inter-VXLAN translation. The router removes the VXLAN5001 header, encapsulates the packet with VXLAN 5000 header information, then sends the packet to VM5. VM1 uses the router to send a packet to VM5 (in another bridge domain) because it learns this information comes from the NSX controller. All hosts learn from the NSX Controller that VM5 is in the bridge domain of Logical Switch 5000 and therefore, must go through the gateway to reach that bridge domain.

## Configuration

### IN THIS SECTION

- [Procedure | 98](#)

### Procedure

#### CLI Quick Configuration

To quickly configure an MX Series router to act as a Layer 3 gateway, copy the following commands and paste them into the switch terminal window:

```
set bridge-domains a35fe7f7-fe82-37b4-b69a-0af4244d1fca vlan-id 1
set bridge-domains a35fe7f7-fe82-37b4-b69a-0af4244d1fca routing-interface irb.1
```

```

set bridge-domains a35fe7f7-fe82-37b4-b69a-0af4244d1fca vxlan ovssdb-managed
set bridge-domains a35fe7f7-fe82-37b4-b69a-0af4244d1fca vxlan vni 5000
set bridge-domains 03b264c5-9540-3666-a34a-c75d828439bc vlan-id 2
set bridge-domains 03b264c5-9540-3666-a34a-c75d828439bc routing-interface irb.2
set bridge-domains 03b264c5-9540-3666-a34a-c75d828439bc ovssdb-managed
set bridge-domains 03b264c5-9540-3666-a34a-c75d828439bc vni 5001
set interfaces xe-2/0/2 flexible-vlan-tagging
set interfaces xe-2/0/2 encapsulation flexible-ethernet-services
set interfaces xe-2/0/2 unit 1 family bridge interface-mode trunk
set interfaces xe-2/0/2 unit 1 family bridge vlan-id-list 1
set interfaces xe-2/0/2 unit 2 family bridge interface-mode trunk
set interfaces xe-2/0/2 unit 2 family bridge vlan-id-list 2
set interfaces irb unit 1 family inet address 102.0.1.254/24
set interfaces irb unit 2 family inet address 102.0.2.254/24
set protocols ovssdb interfaces xe-2/0/2.1
set protocols ovssdb interfaces xe-2/0/2.2

```

## Step-by-Step Procedure

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

To configure an MX Series router to act as a Layer 3 gateway:

1. Specify the bridge domain configuration. The bridge domain name must be the universally unique identifier (UUID) of the logical switch created in the NSX manager (in this topology, logical switches 5000 and 5001). Use the **routing-interface** statement to specify a routing interface to include in the bridge domain. Include the Virtual Extensible LAN (VXLAN) identifier number using the **vni** statement. Add the **ovssdb-managed** statement to specify that MX router will use the *Open vSwitch Database (OVSSDB)* management protocol to learn about the hardware VTEPs in the VXLAN and the MAC addresses learned by the hardware VTEPs.

To locate the UUID number, issue the **show ovssdb logical-switch** command. The UUID number is found in the Logical Switch Name field:

```

user@host> show ovssdb logical-switch
Logical switch information:
Logical Switch Name: a35fe7f7-fe82-37b4-b69a-0af4244d1fca
Flags: Created by both
VNI: 5000

```

```
Logical switch information:
Logical Switch Name: 03b264c5-9540-3666-a34a-c75d828439bc
Flags: Created by both
VNI: 5001
```

#### [edit bridge-domains]

```
user@switch# set a35fe7f7-fe82-37b4-b69a-0af4244d1fca vlan-id 1
user@switch# set a35fe7f7-fe82-37b4-b69a-0af4244d1fca routing-interface irb.1
user@switch# set a35fe7f7-fe82-37b4-b69a-0af4244d1fca ovsdb-managed
user@switch# set a35fe7f7-fe82-37b4-b69a-0af4244d1fca vni 5000
```

#### [edit bridge-domains]

```
user@switch# set 03b264c5-9540-3666-a34a-c75d828439bc vlan-id 2
user@switch# set 03b264c5-9540-3666-a34a-c75d828439bc routing-interface irb.2
user@switch# set 03b264c5-9540-3666-a34a-c75d828439bc vxlan ovsdb-managed
user@switch# set 03b264c5-9540-3666-a34a-c75d828439bc vxlan vni 5001
```

## 2. Configure the Layer 2 port.

#### [edit interfaces]

```
user@switch# set xe-2/0/2 flexible-vlan-tagging
user@switch# set xe-2/0/2 encapsulation flexible-ethernet-services
user@switch# set xe-2/0/2 unit 1 family bridge interface-mode trunk
user@switch# set xe-2/0/2 unit 1 family bridge vlan-id-list1
user@switch# set xe-2/0/2 unit 2family bridge interface-mode trunk
user@switch# set xe-2/0/2 unit 2family bridge vlan-id-list2
```

## 3. Configure the IRB interfaces to route traffic between VXLAN domains

#### [edit interfaces]

```
user@switch# set irb unit 1 family inet address 102.0.1.254/24
user@switch# set irb unit 2family inet address 102.0.2.254/24
```

## 4. Configure the interfaces for the OVSDb protocol:

**NOTE:** The interfaces must also be configured on the NSX controller.

[edit protocols]

```
user@switch# set ovssdb interfaces xe-2/0/2.1
```

```
user@switch# set ovssdb interfaces xe-2/0/2.2
```

## Results

From configuration mode, confirm your configuration by entering the `show bridge domain` command for bridge domains `a35fe7f7-fe82-37b4-b69a-0af4244d1fca` and `03b264c5-9540-3666-a34a-c75d828439bci`:

[edit]

```
user@switch# show bridge domain a35fe7f7-fe82-37b4-b69a-0af4244d1fca
```

```
May 04 16:28:04
```

```
domain-type bridge;
```

```
vlan-id 1;
```

```
routing-interface irb.1;
```

```
vxlan {
```

```
    ovssdb-managed;
```

```
    decapsulate-accept-inner-vlan;
```

```
    vni 5000;
```

```
}
```

```
user@switch# show bridge domain 03b264c5-9540-3666-a34a-c75d828439bc
```

```
May 04 16:28:04
```

```
domain-type bridge;
```

```
vlan-id 2;
```

```
routing-interface irb.2;
```

```
vxlan {
```

```
    ovssdb-managed;
```

```
    decapsulate-accept-inner-vlan;
```

```
    vni 5001;
```

```
}
```

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

## Verification

### IN THIS SECTION

- [Checking the Server IP Address and the VM MAC Address | 102](#)
- [Checking the NSX Controller Connection | 103](#)
- [Checking the OVSDb-Managed Interfaces | 103](#)

Confirm that the configuration is working properly.

### Checking the Server IP Address and the VM MAC Address

#### Purpose

Verify that the server IP address and the VM MAC address are correct.

#### Action

Issue the `show ovssdb mac logical-switch` command, and verify the server IP address and the VM MAC address being used by the bridge domain.

```
user@switch> show ovssdb mac logical-switch a35fe7f7-fe82-37b4-b69a-0af4244d1fca
May 04 16:30:01
Logical Switch Name: a35fe7f7-fe82-37b4-b69a-0af4244d1fca
  Mac          IP          Encapsulation  Vtep
  Address      Address
ff:ff:ff:ff:ff:ff  0.0.0.0      Vxlan over Ipv4  10.255.178.171
00:21:59:ad:27:f0  0.0.0.0      Vxlan over Ipv4  10.255.178.171
00:50:56:83:cb:b3  0.0.0.0      Vxlan over Ipv4  192.168.150.51
ff:ff:ff:ff:ff:ff  0.0.0.0      Vxlan over Ipv4  10.10.0.2
-                  -              -                11.11.0.2
```

#### Meaning

The results displayed by the `show ovssdb mac logical-switch a35fe7f7-fe82-37b4-b69a-0af4244d1fca` command output show the server IP address is 192.168.150.51 and the VM MAC address is 00:50:56:83:cb:b3.

## Checking the NSX Controller Connection

### Purpose

Verify that the connection with the NSX controller is up.

### Action

Issue the `show ovssdb controller` command, and verify that the controller connection state is up.

```
user@switch> show ovssdb controller
May 04 16:32:21
VTEP controller information:
Controller IP address: 25.25.25.25
Controller protocol: ssl
Controller port: 6640
Controller connection: up
Controller seconds-since-connect: 253770
Controller seconds-since-disconnect: 167262
Controller connection status: backoff

Controller IP address: 25.25.25.26
Controller protocol: ssl
Controller port: 6640
Controller connection: up
Controller seconds-since-connect: 253767
Controller seconds-since-disconnect: 167293
Controller connection status: backoff
```

### Meaning

The output shows that the connection state of the NSX controller is up, in addition to other information about the controller. When this connection is up, OVSSDB is enabled on the Juniper Networks device.

## Checking the OVSSDB-Managed Interfaces

### Purpose

Verify the interfaces mapped to OVSSDB.

Action

Issue the `show ovssdb interface` command, and verify the interfaces managed by OVSSDB.

```
user@switch> show ovssdb interface
May 04 16:33:23
Interface          VLAN ID      Bridge-domain
-----
evpn
  irb.1             0            a35fe7f7-fe82-37b4-b69a-0af4244d1fca
  irb.2             0            03b264c5-9540-3666-a34a-c75d828439bc
l3
vpls
  xe-2/0/2.1        1            a35fe7f7-fe82-37b4-b69a-0af4244d1fca
  xe-2/0/2.2        2            03b264c5-9540-3666-a34a-c75d828439bc
```

Meaning

The `show ovssdb interface` command shows that `irb.1` and `xe-2/0/2.1` are being managed in the `a35fe7f7-fe82-37b4-b69a-0af4244d1fca` bridge domain, and `irb.2` and `xe-2/0/2.2` are being managed in the `03b264c5-9540-3666-a34a-c75d828439bc` bridge domain.

RELATED DOCUMENTATION

<a href="#">OVSSDB Support on Juniper Networks Devices   2</a>
<a href="#">Configuring OVSSDB-Managed VXLANs   26</a>
<a href="#">VMware NSX Configuration for Juniper Networks Devices Functioning as Virtual Tunnel Endpoints   29</a>
<a href="#">Example: Setting Up Inter-VXLAN Unicast Routing and OVSSDB Connections in a Data Center   33</a>
<a href="#">Example: Setting Up Inter-VXLAN Unicast and Multicast Routing and OVSSDB Connections in a Data Center   48</a>
<a href="#">Example: Passing Traffic Between Data Centers with DCI in an OVSSDB-Managed Network with MX Series Routers   105</a>



## Example: Passing Traffic Between Data Centers with DCI in an OVSDB-Managed Network with MX Series Routers

### IN THIS SECTION

- [Requirements | 105](#)
- [Overview and Topology | 106](#)
- [Configuration | 108](#)
- [Verification | 113](#)

You can configure an MX Series 5G Universal Routing Platform to route Virtual Extensible LAN (VXLAN) traffic from a local data center in an OVSDB-managed network to a remote data center using Data Center Interconnect (DCI). DCI connects data centers in an enterprise IT environment to share resources or pass traffic between one another.

In this example, an MX Series router is used as the DCI for traffic to pass from a bridge domain in a local data center to a remote data center.

### Requirements

This example uses the following hardware and software components:

- An MX Series router
- Junos OS Release 17.2R1 or later
- A VMware vSphere Distributed Switch (VDS)
- Five virtual machines (VMs)
- An NSX controller
- An NSX manager
- Three servers

## Overview and Topology

### IN THIS SECTION

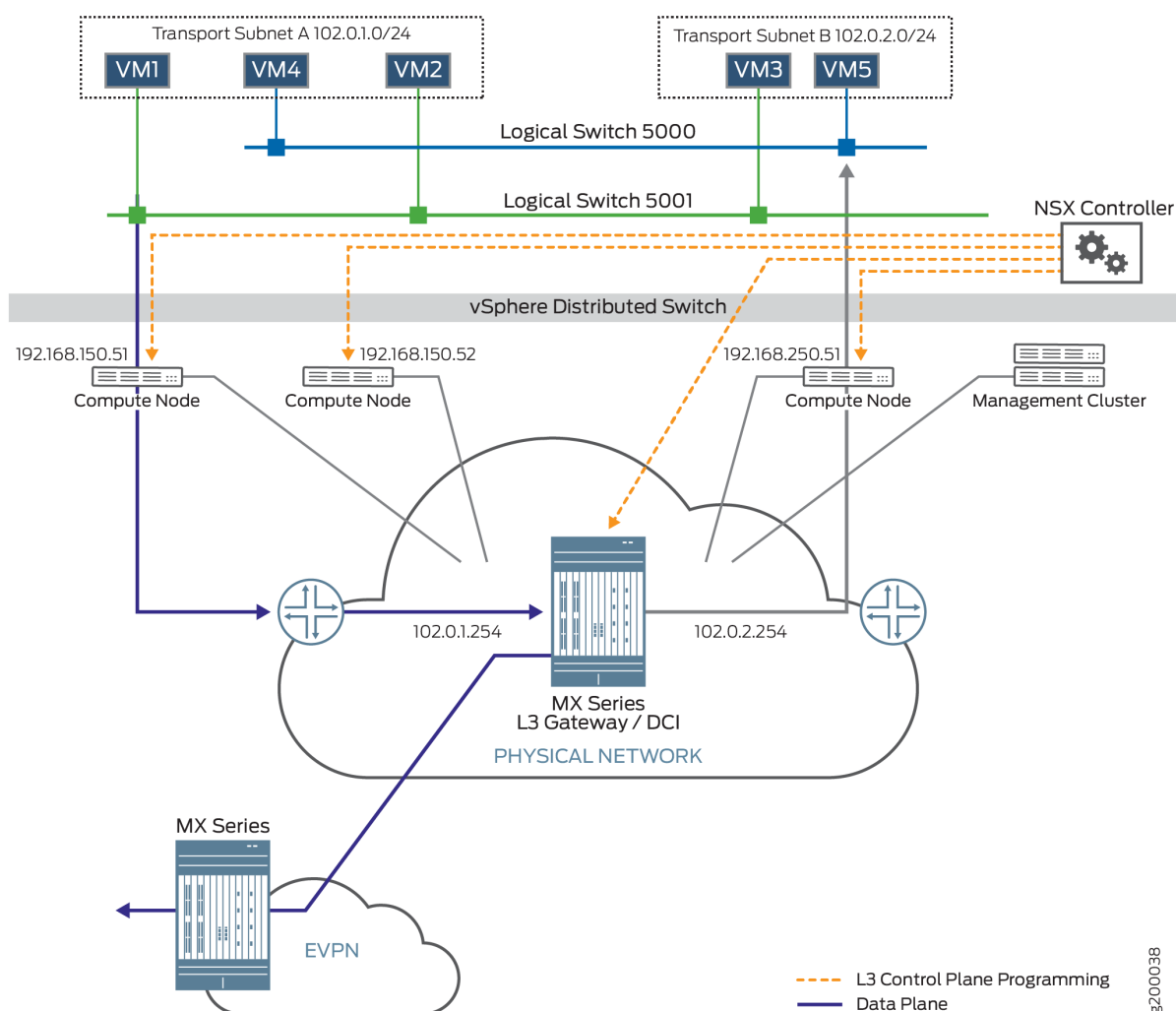
- [Topology | 107](#)

[Figure 7 on page 107](#) shows a data center topology. A VDS provides the centralized interface for configuring, monitoring, and administering virtual machine access switching. Two logical switches are connected to this VDS. The logical switches support five VMs. Logical Switch 5000 supports VM4 and VM5. Logical Switch 5001 supports VM1, VM2, and VM3. Each logical switch has its own bridge domain. Two integrated routing and bridging (IRB) interfaces are associated with these bridge domains and assist in the transfer of packets from one bridge domain to the other through the MX Series router, acting as the Layer 3 gateway. Logical Switch 5000 uses `irb.1`, and Logical Switch 5001 uses `irb.2`.

There are three servers in this data center:

- Server 192.168.150.51 supports VM1 and VM4
- Server 192.168.150.52 supports VM2
- Server 192.168.250.51 supports VM3 and VM5

## Topology

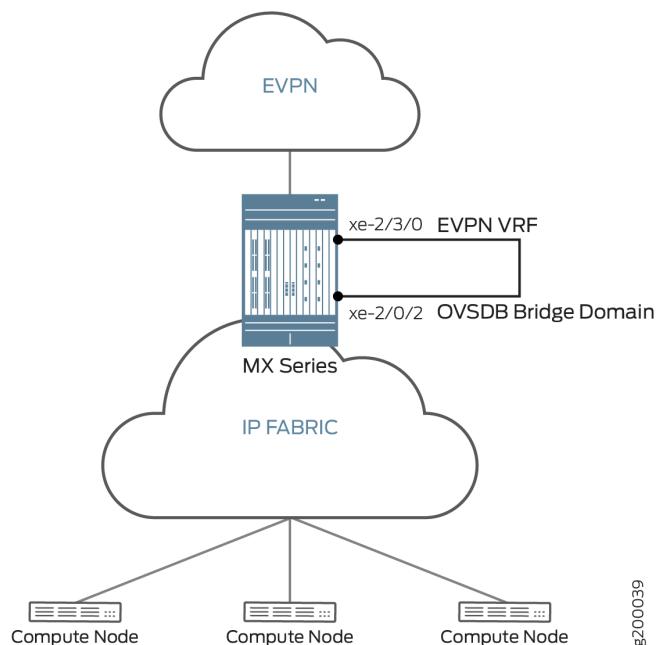


**Figure 7: MX Series Router Acting as DCI**

The management cluster contains an NSX controller and an NSX manager. The NSX controller maintains the runtime space and distributes information to the compute nodes. When a VM is brought up on a compute node, the compute node sends information about the VM, such as MAC and IP addresses, to the NSX controller. The NSX controller then pushes this information to all the servers. The NSX manager handles the management plane, supporting the API and the configuration. It provisions and manages the network, network services, and VXLAN preparation.

In this topology, a Layer 2 port is created and looped on the MX Series router. One end of the Layer 2 port is in the VXLAN bridge domain and the other end is in the EVPN routing instance. [Figure 8 on page 108](#) shows interface xe-2/0/2 looped to interface xe-2/3/0. Interface xe-2/0/2 is part of the VXLAN

bridge domain and interface xe-2/3/0 is part of the EVPN routing instance. Interface xe-2/0/2 ifls are also added in the NSX manager. Using this topology and configuration, the IRB interfaces that map the MX Series router to the NSX controller are brought up. Traffic travels from the local data center to the remote data center by means of a virtual routing instance.



**Figure 8: Layer 2 Port Straddling VXLAN Bridge Domain and EVPN Routing Instance**

In this example, VM1 sends packets from the local data center for delivery to the remote data center. The packets go through the Layer 3 gateway on the MX Series router. There the router de-encapsulates the VXLAN5001 header from the packet and sends the packet out through an EVPN to the remote data center.

## Configuration

### IN THIS SECTION

- Procedure | 109

## Procedure

### CLI Quick Configuration

To quickly configure an MX Series router to act as a DCI and enable VXLAN traffic to travel from a local data center to a remote data center, copy the following commands and paste them into the switch terminal window:

```
[edit]
set bridge-domains a35fe7f7-fe82-37b4-b69a-0af4244d1fca vlan-id 1
set bridge-domains a35fe7f7-fe82-37b4-b69a-0af4244d1fca vxlan ovsdb-managed
set bridge-domains a35fe7f7-fe82-37b4-b69a-0af4244d1fca vxlan vni 5000
set bridge-domains 03b264c5-9540-3666-a34a-c75d828439bc vlan-id 2
set bridge-domains 03b264c5-9540-3666-a34a-c75d828439bc ovsdb-managed
set bridge-domains 03b264c5-9540-3666-a34a-c75d828439bc vni 5001
set interfaces xe-2/3/0 flexible-vlan-tagging
set interfaces xe-2/3/0 encapsulation flexible-ethernet-services
set interfaces xe-2/3/0 unit 1 family bridge interface-mode trunk
set interfaces xe-2/3/0 unit 1 family bridge vlan-id-list 1
set interfaces xe-2/3/0 unit 2 family bridge interface-mode trunk
set interfaces xe-2/3/0 unit 2 family bridge vlan-id-list 2
set interfaces xe-2/0/2 flexible-vlan-tagging
set interfaces xe-2/0/2 encapsulation flexible-ethernet-services
set interfaces xe-2/0/2 unit 1 family bridge interface-mode trunk
set interfaces xe-2/0/2 unit 1 family bridge vlan-id-list 1
set interfaces xe-2/0/2 unit 2 family bridge interface-mode trunk
set interfaces xe-2/0/2 unit 2 family bridge vlan-id-list 2
set interfaces irb unit 1 family inet address 102.0.1.254/24
set interfaces irb unit 2 family inet address 102.0.2.254/24
set routing-instances evpn1 instance-type virtual-switch
set routing-instances evpn1 interface xe-2/3/0.1
set routing-instances evpn1 interface xe-2/3/0.2
set routing-instances evpn1 route-distinguisher 64512:1
set routing-instances evpn1 vrf-target target:64512:1
set routing-instances evpn1 protocols evpnextended-vlan-list 1-2
set routing-instances evpn1 bridge-domains vlan1 domain-type bridge
set routing-instances evpn1 bridge-domains vlan1 vlan-id 1
set routing-instances evpn1 bridge-domains vlan1 routing-interface irb.1
set routing-instances evpn1 bridge-domains vlan2 domain-type bridge
set routing-instances evpn1 bridge-domains vlan2 vlan-id 2
set routing-instances evpn1 bridge-domains vlan2 routing-interface irb.2
```

```
set protocols ovssdb interfaces xe-2/0/2.1
set protocols ovssdb interfaces xe-2/0/2.2
```

## Step-by-Step Procedure

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

To configure an MX Series router as a DCI to transport traffic from a local data center to a remote data center:

1. Specify the bridge domain configuration. The bridge domain name must be the universally unique identifier (UUID) of the logical switch created in NSX manager (in this topology, logical switches 5000 and 5001). Include the VXLAN identifier number using the **vni** statement. Add the **ovssdb-managed** statement to specify that the MX Series router will use the *Open vSwitch Database (OVSSDB)* management protocol to learn about the hardware VTEPs in the VXLAN and the MAC addresses learned by the hardware VTEPs.

To locate the UUID number, issue the **show ovssdb logical-switch** command. The UUID number is found in the Logical Switch Name field.

```
user@host> show ovssdb logical-switch
Logical switch information:
Logical Switch Name: a35fe7f7-fe82-37b4-b69a-0af4244d1fca
Flags: Created by both
VNI: 5000

Logical switch information:
Logical Switch Name: 03b264c5-9540-3666-a34a-c75d828439bc
Flags: Created by both
VNI: 5001
```

```
[edit bridge-domains]
user@switch# set a35fe7f7-fe82-37b4-b69a-0af4244d1fca vlan-id 1
user@switch# set a35fe7f7-fe82-37b4-b69a-0af4244d1fca vxlan ovssdb-managed
user@switch# set a35fe7f7-fe82-37b4-b69a-0af4244d1fca vxlan vni 5000
user@switch# set 03b264c5-9540-3666-a34a-c75d828439bc vlan-id 2
user@switch# set 03b264c5-9540-3666-a34a-c75d828439bc vxlan ovssdb-managed
user@switch# set 03b264c5-9540-3666-a34a-c75d828439bc vxlan vni 5001
```

2. Configure the looped Layer 2 port.

```
[edit interfaces]
user@switch# set xe-2/3/0 flexible-vlan-tagging
user@switch# set xe-2/3/0 encapsulation flexible-ethernet-services
user@switch# set xe-2/3/0 unit 1 family bridge interface-mode trunk
user@switch# set xe-2/3/0 unit 1 family bridge vlan-id-list1
user@switch# set xe-2/3/0 unit 2family bridge interface-mode trunk
user@switch# set xe-2/3/0 unit 2family bridge vlan-id-list2
```

3. Configure the MX Series router port that is connected to the VXLAN bridge.

```
[edit interfaces]
user@switch# set xe-2/0/2 flexible-vlan-tagging
user@switch# set xe-2/0/2 encapsulation flexible-ethernet-services
user@switch# set xe-2/0/2 unit 1 family bridge interface-mode trunk
user@switch# set xe-2/0/2 unit 1 family bridge vlan-id-list1
user@switch# set xe-2/0/2 unit 2family bridge interface-mode trunk
user@switch# set xe-2/0/2 unit 2family bridge vlan-id-list2
```

4. Configure the IRB interfaces to route traffic between the VXLAN domains.

```
[edit interfaces]
user@switch# set irb unit 1 family inet address 102.0.1.254/24
user@switch# set irb unit 2family inet address 102.0.2.254/24
```

5. Configure the virtual switch routing instance to the remote data center.

```
[edit routing-instances]
user@switch# set evpn1 instance-type virtual-switch
user@switch# set evpn1 interface xe-2/3/0.1
user@switch# set evpn1 interface xe-2/3/0.2
user@switch# set evpn1 route-distinguisher 64512:1
user@switch# set evpn1 vrf-target target:64512:1
user@switch# set evpn1 protocols evpNextended-vlan-list 1-2
user@switch# set evpn1 bridge-domains vlan1 domain-type bridge
user@switch# set evpn1 bridge-domains vlan1 vlan-id 1
user@switch# set evpn1 bridge-domains vlan1 routing-interface irb.1
user@switch# set evpn1 bridge-domains vlan2 domain-type bridge
```

```

user@switch# set evpn1 bridge-domains vlan2 vlan-id 2
user@switch# set evpn1 bridge-domains vlan2 routing-interface irb.2

```

6. Configure the OVSDB protocol on the ports. You must also add these same ports as part of the logical switch in NSX manager. When they are added, NSX manager identifies which port on the MX Series router is mapped to the respective VXLAN VNI. In this case VXLAN VNI 5000 is mapped to xe-2/0/2.1 and VXLAN VNI 5001 is mapped to xe-2/0/2.2.

```

[edit protocols]
user@switch# set ovsdb interfaces xe-2/0/2.1
user@switch# set ovsdb interfaces xe-2/0/2.2

```

## Results

From configuration mode, confirm your configuration by entering the `show bridge domain` command for bridge domains `a35fe7f7-fe82-37b4-b69a-0af4244d1fca` and `03b264c5-9540-3666-a34a-c75d828439bc`

```

user@switch# show bridge domain a35fe7f7-fe82-37b4-b69a-0af4244d1fca
May 04 16:27:35
domain-type bridge;
vlan-id 1;
routing-interface irb.1;
vxlan {
    ovsdb-managed;
    vni 5000;
    decapsulate-accept-inner-vlan;
    vni 5000;
}

```

```

user@switch# show bridge domain 03b264c5-9540-3666-a34a-c75d828439bc
May 04 16:28:04
domain-type bridge;
vlan-id 2;
routing-interface irb.2;
vxlan {
    ovsdb-managed;
    vni 5001;
    decapsulate-accept-inner-vlan;
}

```



```
vni 5000;  
}
```

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

Verification

IN THIS SECTION

- [Checking the Server IP Address and the VM MAC Address | 113](#)
- [Checking the NSX Controller Connection | 114](#)
- [Checking the OVSDb-Managed Interfaces | 115](#)
- [Checking the Routing Instance for the EVPN | 115](#)
- [Checking the IRBs for the EVPN Routing Instance | 116](#)

To confirm that the configuration is working properly, perform these tasks:

Checking the Server IP Address and the VM MAC Address

Purpose

Verify that the server IP address and the VM MAC address are correct.

Action

Issue the `show ovssdb mac logical-switch` command, and verify the server IP address and the VM MAC address being used by the bridge domain.

```
user@switch> show ovssdb mac logical-switch a35fe7f7-fe82-37b4-b69a-0af4244d1fca  
May 04 16:30:01  
Logical Switch Name: a35fe7f7-fe82-37b4-b69a-0af4244d1fca  
Mac          IP          Encapsulation  Vtep  
Address      Address  
ff:ff:ff:ff:ff:ff  0.0.0.0    Vxlan over Ipv4  10.255.178.171  
00:21:59:ad:27:f0  0.0.0.0    Vxlan over Ipv4  10.255.178.171  
00:50:56:83:cb:b3  0.0.0.0    Vxlan over Ipv4  192.168.150.51
```

ff:ff:ff:ff:ff:ff	0.0.0.0	Vxlan over Ipv4	10.10.0.2
-			11.11.0.2

## Meaning

The results displayed by the `show ovssdb mac logical-switch a35fe7f7-fe82-37b4-b69a-0af4244d1fca`, command output show the server IP address is 192.168.150.51 and the VM MAC address is 00:50:56:83:cb:b3.

## Checking the NSX Controller Connection

### Purpose

Verify that the connection with the NSX controller is up.

### Action

Issue the `show ovssdb controller` command, and verify that the controller connection state is up.

```
user@switch> show ovssdb controller
May 04 16:32:21
VTEP controller information:
Controller IP address: 25.25.25.25
Controller protocol: ssl
Controller port: 6640
Controller connection: up
Controller seconds-since-connect: 253770
Controller seconds-since-disconnect: 167262
Controller connection status: backoff

Controller IP address: 25.25.25.26
Controller protocol: ssl
Controller port: 6640
Controller connection: up
Controller seconds-since-connect: 253767
Controller seconds-since-disconnect: 167293
Controller connection status: backoff
```

## Meaning

The output shows that the connection state of the NSX controller is up, in addition to other information about the controller. When this connection is up, OVSDb is enabled on the Juniper Networks device.

## Checking the OVSDb-Managed Interfaces

### Purpose

Verify the interfaces mapped to OVSDb.

### Action

Issue the `show ovssdb interface` command, and verify the interfaces managed by OVSDb.

```
user@switch> show ovssdb interface
May 04 16:33:23
Interface          VLAN ID      Bridge-domain
-----
evpn
  irb.1             0            a35fe7f7-fe82-37b4-b69a-0af4244d1fca
  irb.2             0            03b264c5-9540-3666-a34a-c75d828439bc
l3
vpls
  xe-2/0/2.1        1            a35fe7f7-fe82-37b4-b69a-0af4244d1fca
  xe-2/0/2.2        2            03b264c5-9540-3666-a34a-c75d828439bc
```

## Meaning

The `show ovssdb interface` command shows that `irb.1` and `xe-2/0/2.1` are being managed in the `a35fe7f7-fe82-37b4-b69a-0af4244d1fca` bridge domain, and `irb.2` and `xe-2/0/2.2` are being managed in the `03b264c5-9540-3666-a34a-c75d828439bc` bridge domain.

## Checking the Routing Instance for the EVPN

### Purpose

Verify that the IRB interfaces are configured and active for the EVPN.

## Action

Issue the `show evpn instance` command, and verify the EVPN routing instance information.

```
user@switch> show evpn instance evpn1
May 11 10:56:45
```

	Intfs		IRB intfs			MH	MAC addresses	
Instance	Total	Up	Total	Up	Nbrs	ESIs	Local	Remote
evpn1	2	2	2	2	0	0	2	0

Issue the `show evpn database instance evpn1` command, and verify the Active Source information.

```
user@switch> show evpn database instance evpn1
May 11 10:58:24
Instance: evpn1
```

VLAN	DomainId	MAC address	Active source	Timestamp	IP address
3		00:21:59:ad:27:f0	irb.1	May 09 16:43:54	102.0.1.254
4		00:21:59:ad:27:f0	irb.2	May 09 16:43:54	102.0.2.254

## Meaning

The results displayed by the `show evpn instance evpn1` command verify the routing instance and the field IRB intfs shows that two IRB interfaces are up. The results displayed by the `show evpn database instance evpn1` command show that under the Active Source field, irb.1 and irb.2 are traffic sources.

## Checking the IRBs for the EVPN Routing Instance

### Purpose

Verify the IRB interfaces for the EVPN.

## Action

Issue the `show evpn database instance` command, and verify the EVPN routing instance information.

```
user@switch> show evpn database instance evpn1
May 11 10:58:24
Instance: evpn1
```

VLAN	DomainId	MAC address	Active source	Timestamp	IP address
3		00:21:59:ad:27:f0	irb.1	May 09 16:43:54	102.0.1.254
4		00:21:59:ad:27:f0	irb.2	May 09 16:43:54	102.0.2.254

## Meaning

The results displayed by the `show evpn database instance evpn1` command show the EVPN1 instance and verifies under IRB intfs that two IRB interfaces are up and running.

## RELATED DOCUMENTATION

[OVSDb Support on Juniper Networks Devices | 2](#)

[Configuring OVSDb-Managed VXLANs | 26](#)

[VMware NSX Configuration for Juniper Networks Devices Functioning as Virtual Tunnel Endpoints | 29](#)

[Example: Configuring Inter-VXLAN Traffic Routing from One Bridge Domain to Another Using an MX Series Router as a Layer 3 Gateway | 95](#)

[Example: Setting Up Inter-VXLAN Unicast Routing and OVSDb Connections in a Data Center | 33](#)

[Example: Setting Up Inter-VXLAN Unicast and Multicast Routing and OVSDb Connections in a Data Center | 48](#)

## CHAPTER 3

# OVSDB Configuration Statements

**IN THIS CHAPTER**

- [bridge-domains | 118](#)
- [controller \(OVSDB\) | 121](#)
- [inactivity-probe-duration | 123](#)
- [ingress-node-replication | 124](#)
- [interfaces \(OVSDB\) | 126](#)
- [maximum-backoff-duration | 127](#)
- [ovsdb | 129](#)
- [ovsdb-managed | 130](#)
- [port \(OVSDB\) | 132](#)
- [protocol \(OVSDB\) | 133](#)
- [traceoptions \(OVSDB\) | 135](#)
- [routing-instances \(Multiple Routing Entities\) | 137](#)
- [interface-mode | 140](#)
- [unit | 142](#)
- [vlan-id-list \(Interface in Bridge Domain\) | 153](#)

## bridge-domains

**IN THIS SECTION**

- [Syntax | 119](#)
- [Hierarchy Level | 119](#)
- [Description | 120](#)
- [Options | 120](#)

- Required Privilege Level | 120
- Release Information | 120

## Syntax

```

bridge-domains {
  bridge-domain-name {
    bridge-options {
      ...bridge-options-configuration...
    }
    domain-type bridge;
    interface interface-name;
    no-irb-layer-2-copy;
    no-local-switching;
    routing-interface routing-interface-name;
    vlan-id (all | none | number);
    vlan-id-list [ vlan-id-numbers ];
    vlan-tags outer number inner number;
    bridge-options {
      interface interface-name {
        mac-pinning
        static-mac mac-address;
      }
      interface-mac-limit limit;
      mac-statistics;
      mac-table-size limit;
      no-mac-learning;
    }
  }
}

```

## Hierarchy Level

```

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

```

## Description

(MX Series routers only) Configure a domain that includes a set of logical ports that share the same flooding or broadcast characteristics in order to perform Layer 2 bridging.

## Options

***bridge-domain-name***—Name of the bridge domain.

**NOTE:** You cannot use the slash (/) character as part of the bridge domain name. If you do, the configuration will not commit.

The remaining statements are explained separately. See [CLI Explorer](#).

## Required Privilege Level

routing—To view this statement in the configuration.

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

## Release Information

Statement introduced in Junos OS Release 8.4.

Support for logical systems added in Junos OS Release 9.6.

Support for the `no-irb-layer-2-copy` statement added in Junos OS Release 10.2.

## RELATED DOCUMENTATION

[Configuring a Bridge Domain](#)

[Configuring a Layer 2 Virtual Switch](#)



## controller (OVSDB)

### IN THIS SECTION

- Syntax | 121
- Hierarchy Level | 121
- Description | 121
- Options | 122
- Required Privilege Level | 122
- Release Information | 122

### Syntax

```
controller ip-address {
    inactivity-probe-duration milliseconds;
    maximum-backoff-duration milliseconds;
    protocol protocol {
        port number;
    }
}
```

### Hierarchy Level

```
[edit protocols ovsdb]
```

### Description

Configure a connection between a Juniper Networks device running the *Open vSwitch Database (OVSDB)* management protocol and a software-defined networking (SDN) controller. You can connect a Juniper Networks device to more than one SDN controller for redundancy.

In a VMware NSX environment, one cluster of NSX controllers typically includes three or five controllers. To implement the OVSDB management protocol on a Juniper Networks device, you must explicitly configure a connection to one NSX controller, using the Junos OS CLI. If the NSX controller to

which you explicitly configure a connection is in a cluster, the controller pushes information about other controllers in the same cluster to the device, and the device establishes connections with the other controllers. However, you can also explicitly configure connections with the other controllers in the cluster, using the Junos OS CLI.

To implement the OVSDB management protocol on a Juniper Networks device in a Contrail environment, you must configure a connection to a Contrail controller, using the Junos OS CLI.

Connections to all SDN controllers are made on the management interface of the Juniper Networks device.

## Options

*ip-address*—IPv4 address of the SDN controller.

The remaining statements are explained separately. See [CLI Explorer](#).

## Required Privilege Level

admin—To view this statement in the configuration.

admin-control—To add this statement to the configuration.

## Release Information

Statement introduced in Junos OS Release 14.1R2.

## RELATED DOCUMENTATION

*Setting Up OVSDB on Juniper Networks Devices That Support the Dynamic Configuration of VXLANs*

[Setting Up the OVSDB Protocol on Juniper Networks Devices that Support Manual Configuration of VXLANs | 24](#)

[Understanding How to Set Up OVSDB Connections on a Juniper Networks Device | 7](#)

## inactivity-probe-duration

### IN THIS SECTION

- [Syntax | 123](#)
- [Hierarchy Level | 123](#)
- [Description | 123](#)
- [Options | 123](#)
- [Required Privilege Level | 124](#)
- [Release Information | 124](#)

### Syntax

```
inactivity-probe-duration milliseconds;
```

### Hierarchy Level

```
[edit protocols ovsdb controller ]
```

### Description

Configure the maximum amount of time, in milliseconds, that the connection between a Juniper Networks device that supports the *Open vSwitch Database (OVSDB)* management protocol and a software-defined networking (SDN) controller can be inactive before an inactivity probe is sent.

### Options

*milliseconds*—Number of milliseconds that the connection can be inactive before an inactivity probe is sent.

- **Range:** 0 through 4,294,967,295
- **Default:** 0. This value indicates that an inactivity probe is never sent.

## Required Privilege Level

admin—To view this statement in the configuration.

admin-control—To add this statement to the configuration.

## Release Information

Statement introduced in Junos OS Release 14.1R2.

## RELATED DOCUMENTATION

[Understanding How to Set Up OVSDDB Connections on a Juniper Networks Device](#) | 7

# ingress-node-replication

## IN THIS SECTION

- [Syntax](#) | 124
- [Hierarchy Level](#) | 125
- [Description](#) | 125
- [Default](#) | 125
- [Required Privilege Level](#) | 125
- [Release Information](#) | 125

## Syntax

```
ingress-node-replication;
```

## Hierarchy Level

```
[edit bridge-domains bridge-domain-name vxlan],
[edit routing-instances routing-instance-name bridge-domains bridge-domain-name vxlan],
[edit routing-instances routing-instance-name vlans vlan-name vxlan],
[edit routing-instances routing-instance-name vxlan]
[edit vlans vlan-name vxlan]
```

## Description

Enable ingress node replication for a specified Virtual Extensible LAN (VXLAN) that is managed by the Open vSwitch Database (OVSDB) management protocol.

With this feature enabled, instead of service nodes, Juniper Networks devices with OVSDB implemented handle incoming broadcast, unknown unicast, or multicast (BUM) traffic. For more information about the scenarios in which you can use ingress node replication and how it works, see ["Understanding How Layer 2 BUM and Layer 3 Routed Multicast Traffic Are Handled with OVSDB" on page 8](#).

**NOTE:** When Juniper Networks devices replicate Layer 2 BUM packets to a large number of remote software VTEPs, the performance of the Juniper Networks devices might be impacted.

## Default

If you do not include the `ingress-node-replication` statement, one or more service nodes handle BUM traffic.

## Required Privilege Level

admin—To view this statement in the configuration.

admin-control—To add this statement to the configuration.

## Release Information

Statement introduced in Junos OS Release 14.1R2.

## RELATED DOCUMENTATION

| [Configuring OVSDb-Managed VXLANs](#) | 26

## interfaces (OVSDb)

### IN THIS SECTION

- [Syntax](#) | 126
- [Hierarchy Level](#) | 126
- [Description](#) | 126
- [Options](#) | 126
- [Required Privilege Level](#) | 127
- [Release Information](#) | 127

### Syntax

```
interfaces interface-name;
```

### Hierarchy Level

```
[edit protocols ovsdb]
```

### Description

Specify the physical interfaces on a Juniper Networks device that you want the *Open vSwitch Database (OVSDb)* protocol to manage. Typically, the only interfaces that need to be managed by OVSDb are interfaces that are connected to physical servers.

### Options

*interface-name*

Name of the interface.

## Required Privilege Level

admin—To view this statement in the configuration.

admin-control—To add this statement to the configuration.

## Release Information

Statement introduced in Junos OS Release 14.1R2.

## RELATED DOCUMENTATION

[Setting Up the OVSDDB Protocol on Juniper Networks Devices that Support Manual Configuration of VXLANs | 24](#)

*Setting Up OVSDDB on Juniper Networks Devices That Support the Dynamic Configuration of VXLANs*

## maximum-backoff-duration

### IN THIS SECTION

- [Syntax | 127](#)
- [Hierarchy Level | 128](#)
- [Description | 128](#)
- [Options | 128](#)
- [Required Privilege Level | 128](#)
- [Release Information | 128](#)

## Syntax

```
maximum-backoff-duration milliseconds;
```

## Hierarchy Level

```
[edit protocols ovsdb controller ]
```

## Description

Specify (in milliseconds) how long a Juniper Networks device that supports the *Open vSwitch Database (OVSDB)* management protocol waits before it tries again to connect with a software-defined networking (SDN) controller after a previous attempt has failed.

## Options

*milliseconds*—Number of milliseconds a Juniper Networks device waits before it tries again to connect with an SDN controller.

- **Range:** 1000 through 4,294,967,295
- **Default:** 1000

## Required Privilege Level

admin—To view this statement in the configuration.

admin-control—To add this statement to the configuration.

## Release Information

Statement introduced in Junos OS Release 14.1R2.

## RELATED DOCUMENTATION

| [Understanding How to Set Up OVSDB Connections on a Juniper Networks Device](#) | 7



## ovsdb

### IN THIS SECTION

- [Syntax | 129](#)
- [Hierarchy Level | 129](#)
- [Description | 130](#)
- [Default | 130](#)
- [Required Privilege Level | 130](#)
- [Release Information | 130](#)

### Syntax

```
ovsdb {
    controller ip-address {
        inactivity-probe-duration milliseconds;
        maximum-backoff-duration milliseconds;
        protocol protocol {
            port number;
        }
    }
    interfaces interface-name;
    traceoptions {
        file <filename> <files number> <match regular-expression> <no-world-readable | world-
readable> <size size>;
        flag flag;
        no-remote-trace;
    }
}
```

### Hierarchy Level

[edit protocols]

## Description

Configure support for the *Open vSwitch Database (OVSDB)* management protocol on a Juniper Networks device.

The remaining statements are explained separately. See [CLI Explorer](#).

## Default

The OVSDB management protocol is disabled on Juniper Networks devices.

## Required Privilege Level

admin—To view this statement in the configuration.

admin-control—To add this statement to the configuration.

## Release Information

Statement introduced in Junos OS Release 14.1R2.

## RELATED DOCUMENTATION

| [Understanding the OVSDB Protocol Running on Juniper Networks Devices](#) | 6

## ovsdb-managed

### IN THIS SECTION

- [Syntax](#) | 131
- [Hierarchy Level](#) | 131
- [Description](#) | 131
- [Required Privilege Level](#) | 131
- [Release Information](#) | 131

## Syntax

```
ovsdb-managed;
```

## Hierarchy Level

```
[edit bridge-domains bridge-domain-name vxlan],
[edit routing-instances routing-instance-name bridge-domains bridge-domain-name vxlan],
[edit routing-instances routing-instance-name switch-options],
[edit routing-instances routing-instance-name vlans vlan-name vxlan],
[edit routing-instances routing-instance-name vxlan],
[edit switch-options],
[edit vlans vlan-name vxlan]
```

## Description

Disable a Juniper Networks device from learning about other Juniper Networks devices that function as hardware *virtual tunnel endpoints (VTEPs)* in a specified Virtual Extensible LAN (VXLAN) and the media access control (MAC) addresses learned by the hardware VTEPs. Instead, the Juniper Networks device uses the *Open vSwitch Database (OVSDB)* management protocol to learn about the hardware VTEPs in the VXLAN and the MAC addresses learned by the hardware VTEPs.

The specified VXLAN must have a VXLAN network identifier (VNI) configured, using the `vni` statement in the `[edit bridge-domains bridge-domain-name vxlan]`, `[edit routing-instance routing-instance-name vxlan]`, or `[edit vlans vlan-name vxlan]` hierarchy.

Also, for OVSDB-managed VXLANs, the multicast scheme described in "[Understanding How Layer 2 BUM and Layer 3 Routed Multicast Traffic Are Handled with OVSDB](#)" on page 8 is used. Therefore, specifying the `multicast-group` statement in the `[edit bridge-domains bridge-domain-name vxlan]`, `[edit routing-instances routing-instance-name vxlan]`, or `[edit vlans vlan-name vxlan]` hierarchy has no effect.

## Required Privilege Level

admin—To view this statement in the configuration.

admin-control—To add this statement to the configuration.

## Release Information

Statement introduced in Junos OS Release 14.1R2.

## RELATED DOCUMENTATION

*Understanding Dynamically Configured VXLANs in an OVSDb Environment*

[Configuring OVSDb-Managed VXLANs | 26](#)

## port (OVSDb)

### IN THIS SECTION

- [Syntax | 132](#)
- [Hierarchy Level | 132](#)
- [Description | 132](#)
- [Options | 132](#)
- [Required Privilege Level | 133](#)
- [Release Information | 133](#)

### Syntax

```
port number;
```

### Hierarchy Level

```
[edit protocols ovsdb controller protocol ]
```

### Description

Specify the software-defined networking (SDN) controller port to which a Juniper Networks device that supports the *Open vSwitch Database (OVSDb)* management protocol connects.

### Options

*number*                      Number of SDN controller port.

- **Range:** 1024 through 65,535
- **Default:** 6632

## Required Privilege Level

admin—To view this statement in the configuration.

admin-control—To add this statement to the configuration.

## Release Information

Statement introduced in Junos OS Release 14.1R2.

## RELATED DOCUMENTATION

[Understanding How to Set Up OVSDDB Connections on a Juniper Networks Device](#) | 7

## protocol (OVSDDB)

### IN THIS SECTION

- [Syntax](#) | 133
- [Hierarchy Level](#) | 134
- [Description](#) | 134
- [Options](#) | 134
- [Required Privilege Level](#) | 134
- [Release Information](#) | 134

## Syntax

```
protocol protocol {
  port number;
}
```

## Hierarchy Level

```
[edit protocols ovsdb controller ]
```

## Description

Configure the security protocol that protects the connection between a Juniper Networks device that supports the *Open vSwitch Database (OVSDB)* management protocol and a software-defined networking (SDN) controller.

The *Secure Sockets Layer (SSL)* connection requires a private key and certificates, which must be stored in the `/var/db/certs` directory of the Juniper Networks device. See ["Creating and Installing an SSL Key and Certificate on a Juniper Networks Device for Connection with SDN Controllers"](#) on page 23.

## Options

***protocol*** Establish a secure connection to the SDN controller, using SSL or TCP.

**NOTE:** SSL is the only supported connection protocol.

- **Default:** ssl

The remaining statement is explained separately. See [CLI Explorer](#).

## Required Privilege Level

admin—To view this statement in the configuration.

admin-control—To add this statement to the configuration.

## Release Information

Statement introduced in Junos OS Release 14.1R2.

## RELATED DOCUMENTATION

[Understanding How to Set Up OVSDB Connections on a Juniper Networks Device](#) | 7

## traceoptions (OVSDB)

### IN THIS SECTION

- [Syntax | 135](#)
- [Hierarchy Level | 135](#)
- [Description | 135](#)
- [Default | 135](#)
- [Options | 136](#)
- [Required Privilege Level | 137](#)
- [Release Information | 137](#)

### Syntax

```
traceoptions {  
    file <filename> <files number> <match regular-expression> <no-world-readable | world-  
readable> <size size>;  
    flag flag;  
    no-remote-trace;  
}
```

### Hierarchy Level

[edit protocols [ovsdb](#)]

### Description

Define tracing operations for the *Open vSwitch Database (OVSDB)* management protocol, which is supported on Juniper Networks devices.

### Default

If you do not include this statement, OVSDB-specific tracing operations are not performed.

## Options

**file *filename*** Name of file in which the system places the output of the tracing operations. By default, the system places all files in the **/var/log** directory.

- **Default:** **/var/log/vgd**

**files *number*** (Optional) Maximum number of trace files. When a trace file reaches the size specified by the **size** option, the filename is appended with 0 and compressed. For example, a trace file named **trace-file.gz** would be renamed **trace-file.0.gz**. When **trace-file.0.gz** reaches the specified size, it is renamed **trace-file.1.gz** and its contents are compressed to **trace-file.0.gz**. This renaming scheme continues until the maximum number of trace files is reached. Then the oldest trace file is overwritten.

If you specify a maximum number of files, you also must specify a maximum file size with the **size** option and a filename.

- **Range:** 2 through 1000 files
- **Default:** 10 files

**flag *flag*** Tracing operation to perform. You can include one or more of the following flags:

**all** All OVSDB events.

**configuration** OVSDB configuration events.

**core** OVSDB core events.

**function** OVSDB function events.

**interface** OVSDB interface events.

**l2-client** OVSDB Layer 2 client events.

**netconf-client** (QFX Series switches only) Events for the dynamic configuration of Virtual Extensible LANs (VXLANs).

**ovs-client** OVSDB client events.

**match *regular-expression*** (Optional) Only log lines that match the regular expression.



<b>no-remote-trace</b>	(Optional) Disable tracing and logging operations that track normal operations, error conditions, and packets that are generated by or passed through the Juniper Networks device.
<b>no-world-readable</b>	<p>Restrict access to the trace files to the owner.</p> <ul style="list-style-type: none"> <li>• <b>Default:</b> no-world-readable</li> </ul>
<b>size <i>size</i></b>	<p>(Optional) Maximum size of each trace file in bytes, kilobytes (KB), megabytes (MB), or gigabytes (GB). If you do not specify a unit, the default is bytes. If you specify a maximum file size, you also must specify a maximum number of trace files by using the <code>files</code> option and a filename by using the <code>file</code> option.</p> <ul style="list-style-type: none"> <li>• <b>Syntax:</b> <i>size</i> to specify bytes, <i>sizek</i> to specify KB, <i>sizem</i> to specify MB, or <i>sizeg</i> to specify GB.</li> <li>• <b>Range:</b> 10,240 through 1,073,741,824 bytes</li> <li>• <b>Default:</b> 128 KB</li> </ul>
<b>world-readable</b>	Enable any user to access the trace files.

## Required Privilege Level

admin—To view this statement in the configuration.

admin-control—To add this statement to the configuration.

## Release Information

Statement introduced in Junos OS Release 14.1R2.

## routing-instances (Multiple Routing Entities)

### IN THIS SECTION

- [Syntax | 138](#)
- [Hierarchy Level | 138](#)
- [Description | 138](#)

- [Default | 139](#)
- [Options | 139](#)
- [Required Privilege Level | 139](#)
- [Release Information | 139](#)

## Syntax

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

## Hierarchy Level

```
[edit],  
[edit logical-systems logical-system-name]
```

## Description

Configure an additional routing entity for a router. You can create multiple instances of BGP, IS-IS, OSPF, OSPFv3, and RIP for a router. You can also create multiple routing instances for separating routing tables, routing policies, and interfaces for individual wholesale subscribers (retailers) in a Layer 3 wholesale network.

Each routing instance consist of the following:

- A set of routing tables
- A set of interfaces that belong to these routing tables
- A set of routing option configurations

Each routing instance has a unique name and a corresponding IP unicast table. For example, if you configure a routing instance with the name `my-instance`, its corresponding IP unicast table is `my-instance.inet.0`. All routes for `my-instance` are installed into `my-instance.inet.0`.

Routes are installed into the default routing instance `inet.0` by default, unless a routing instance is specified.

In Junos OS Release 9.0 and later, you can no longer specify a routing-instance name of *primary*, *default*, or *bgp* or include special characters within the name of a routing instance.

In Junos OS Release 9.6 and later, you can include a slash (/) in a routing-instance name only if a logical system is not configured. That is, you cannot include the slash character in a routing-instance name if a logical system other than the default is explicitly configured. Routing-instance names, further, are restricted from having the form `__.*__` (beginning and ending with underscores). The colon : character cannot be used when multitopology routing (MTR) is enabled.

## Default

Routing instances are disabled for the router.

## Options

*routing-instance-name* —Name of the routing instance. This must be a non-reserved string of not more than 128 characters.

*remote-vtep-list*      Configure static remote VXLAN tunnel endpoints.

*remote-vtep-v6-list*    Configure static IPv6 remote VXLAN tunnel endpoints.

## Required Privilege Level

routing—To view this statement in the configuration.

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

## Release Information

Statement introduced before Junos OS Release 7.4.

*remote-vtep-v6-list* statement introduced in Junos OS Release 17.3 for MX Series routers with MPC and MIC interfaces.

## RELATED DOCUMENTATION

---

*Example: Configuring Interprovider Layer 3 VPN Option A*

---

*Example: Configuring Interprovider Layer 3 VPN Option B*

---

*Example: Configuring Interprovider Layer 3 VPN Option C*

## interface-mode

### IN THIS SECTION

- Syntax | 140
- Hierarchy Level | 140
- Description | 140
- Options | 141
- Required Privilege Level | 141
- Release Information | 141

### Syntax

```
interface-mode (access | trunk <inter-switch-link>);
```

### Hierarchy Level

```
[edit interfaces interface-name unit logical-unit-number family bridge],  
[edit interfaces interface-name unit logical-unit-number family ethernet-switching],  
[edit logical-systems logical-system-name interfaces interface-name unit logical-unit-number  
family bridge]
```

### Description

**NOTE:** This statement supports the Enhanced Layer 2 Software (ELS) configuration style. If your switch runs software that does not support ELS, see [port-mode](#). For ELS details, see [Using the Enhanced Layer 2 Software CLI](#).

QFX3500 and QFX3600 standalone switches—Determine whether the logical interface accepts or discards packets based on VLAN tags. Specify the trunk option to accept packets with a VLAN ID that matches the list of VLAN IDs specified in the `vlan-id` or `vlan-id-list` statement, then forward the packet within the bridge domain or VLAN configured with the matching VLAN ID. Specify the access option to

accept packets with no VLAN ID, then forward the packet within the bridge domain or VLAN configured with the VLAN ID that matches the VLAN ID specified in the `vlan-id` statement.

**NOTE:** On MX Series routers, if you want IGMP snooping to be functional for a bridge domain, then you should not configure `interface-mode` and `irb` for that bridge. Such a configuration commit succeeds, but IGMP snooping is not functional, and a message informing the same is displayed. For more information, see [Configuring a Trunk Interface on a Bridge Network](#).

## Options

`access`—Configure a logical interface to accept untagged packets. Specify the VLAN to which this interface belongs using the `vlan-id` statement.

`trunk`—Configure a single logical interface to accept packets tagged with any VLAN ID specified with the `vlan-id` or `vlan-id-list` statement.

`trunk inter-switch-link`—For a private VLAN, configure the InterSwitch Link protocol (ISL) on a trunk port of the primary VLAN in order to connect the switches composing the PVLAN to each other. You do not need to configure an ISL when a PVLAN is configured on a single switch. This configuration specifies whether the particular interface assumes the role of interswitch link for the PVLAN domains of which it is a member. This option is supported only on MX240, MX480, and MX960 routers in enhanced LAN mode.

## Required Privilege Level

`interface`—To view this statement in the configuration.

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

## Release Information

Statement introduced in Junos OS Release 9.2.

`inter-switch-link` option introduced in Junos OS Release 14.2 for MX240, MX480, and MX960 routers in enhanced LAN mode.

## RELATED DOCUMENTATION

[Configuring Access Mode on a Logical Interface](#)

[Configuring a Logical Interface for Trunk Mode](#)

[Example: Connecting Access Switches with ELS Support to a Distribution Switch with ELS Support](#)

[Tunnel Services Overview](#)

[Tunnel Interface Configuration on MX Series Routers Overview](#)

## unit

### IN THIS SECTION

- [Syntax | 142](#)
- [Hierarchy Level | 151](#)
- [Description | 152](#)
- [Options | 152](#)
- [Required Privilege Level | 152](#)
- [Release Information | 152](#)

## Syntax

```
unit logical-unit-number {
    accept-source-mac {
        mac-address mac-address {
            policer {
                input cos-policer-name;
                output cos-policer-name;
            }
        }
    }
    accounting-profile name;
    advisory-options {
        downstream-rate rate;
        upstream-rate rate;
    }
    allow-any-vci;
    atm-scheduler-map (map-name | default);
    auto-configure {
```



```

        interface-name;
        source-address;
        user-prefix user-prefix-string;
    }
}
dynamic-profile profile-name {
    network ip-address {
        range name {
            low lower-limit;
            high upper-limit;
        }
    }
}
}
}
}
inet6 {
    address-source address;
    auto-configure {
        address-ranges {
            authentication {
                password password-string;
                username-include {
                    auth-server-realm realm-string;
                    delimiter delimiter-character;
                    domain-name domain-name;
                    interface-name;
                    source-address;
                    user-prefix user-prefix-string;
                }
            }
        }
        dynamic-profile profile-name {
            network ip-address {
                range name {
                    low lower-limit;
                    high upper-limit;
                }
            }
        }
    }
}
}

```



```

    }
}
demux-destination family;
demux-source family;
demux-options {
    underlying-interface interface-name;
}
description text;
etree-ac-role (leaf | root);
interface {
    l2tp-interface-id name;
    (dedicated | shared);
}
dialer-options {
    activation-delay seconds;
    callback;
    callback-wait-period time;
    deactivation-delay seconds;
    dial-string [dial-string-numbers];
    idle-timeout seconds;
    incoming-map {
        caller caller-id | accept-all;
        initial-route-check seconds;
        load-interval seconds;
        load-threshold percent;
        pool pool-name;
        redial-delay time;
        watch-list {
            [routes];
        }
    }
}
}
disable;
disable-mlppp-inner-ppp-pfc;
dlci dlci-identifier;
drop-timeout milliseconds;
dynamic-call-admission-control {
    activation-priority priority;
    bearer-bandwidth-limit kilobits-per-second;
}
encapsulation type;
epd-threshold cells plp1 cells;
family family-name {

```

... the family *subhierarchy* appears after the main [edit interfaces *interface-name* unit *logical-unit-number*] *hierarchy* ...

```

}
fragment-threshold bytes;
host-prefix-only;
inner-vlan-id-range start start-id end end-id;
input-vlan-map {
    (pop | pop-pop | pop-swap | push | push-push | swap |
    swap-push | swap-swap);
    inner-tag-protocol-id tpid;
    inner-vlan-id number;
    tag-protocol-id tpid;
    vlan-id number;
}
interleave-fragments;
inverse-arp;
layer2-policer {
    input-policer policer-name;
    input-three-color policer-name;
    output-policer policer-name;
    output-three-color policer-name;
}
link-layer-overhead percent;
minimum-links number;
mrru bytes;
multicast-dlci dlci-identifier;
multicast-vci vpi-identifier.vci-identifier;
multilink-max-classes number;
multipoint;
oam-liveness {
    up-count cells;
    down-count cells;
}
oam-period (disable | seconds);
output-vlan-map {
    (pop | pop-pop | pop-swap | push | push-push | swap |
    swap-push | swap-swap);
    inner-tag-protocol-id tpid;
    inner-vlan-id number;
    tag-protocol-id tpid;
}
passive-monitor-mode;
peer-unit unit-number;

```

```

plp-to-clp;
point-to-point;
ppp-options {
    mru size;
    mtu (size | use-lower-layer);
    chap {
        access-profile name;
        default-chap-secret name;
        local-name name;
        passive;
    }
    compression {
        acfc;
        pfc;
    }
    dynamic-profile profile-name;
    ipcp-suggest-dns-option;
    lcp-restart-timer milliseconds;
    loopback-clear-timer seconds;
    ncp-restart-timer milliseconds;
    pap {
        access-profile name;
        default-pap-password password;
        local-name name;
        local-password password;
        passive;
    }
}
pppoe-options {
    access-concentrator name;
    auto-reconnect seconds;
    (client | server);
    service-name name;
    underlying-interface interface-name;
}
pppoe-underlying-options {
    access-concentrator name;
    direct-connect;
    dynamic-profile profile-name;
    max-sessions number;
}
proxy-arp;
service-domain (inside | outside);

```

```

shaping {
    (cbr rate | rtvbr peak rate sustained rate burst length | vbr peak rate sustained rate
burst length);
    queue-length number;
}
short-sequence;
targeted-distribution;
transmit-weight number;
(traps | no-traps);
trunk-bandwidth rate;
trunk-id number;
tunnel {
    backup-destination address;
    destination address;
    key number;
    routing-instance {
        destination routing-instance-name;
    }
    source source-address;
    ttl number;
}
vci vpi-identifier.vci-identifier;
vci-range start start-vci end end-vci;
vpi vpi-identifier;
vlan-id number;
vlan-id-range number-number;
vlan-tags inner tpid.vlan-id outer tpid.vlan-id;
family family {
    accounting {
        destination-class-usage;
        source-class-usage {
            (input | output | input output);
        }
    }
    access-concentrator name;
    address address {
        ... the address subhierarchy appears after the main [edit interfaces interface-name
unit logical-unit-number family family-name] hierarchy ...
    }
    bundle interface-name;
    core-facing;
    demux-destination {
        destination-prefix;

```

```

}
demux-source {
    source-prefix;
}
direct-connect;
duplicate-protection;
dynamic-profile profile-name;
filter {
    group filter-group-number;
    input filter-name;
    input-list [filter-names];
    output filter-name;
    output-list [filter-names];
}
interface-mode (access | trunk);
ipsec-sa sa-name;
keep-address-and-control;
mac-validate (loose | strict);
max-sessions number;
mtu bytes;
multicast-only;
no-redirects;
policer {
    arp policer-template-name;
    input policer-template-name;
    output policer-template-name;
}
primary;
protocols [inet iso mpls];
proxy inet-address address;
receive-options-packets;
receive-ttl-exceeded;
remote (inet-address address | mac-address address);
rpf-check {
    fail-filter filter-name
    mode loose;
}
sampling {
    input;
    output;
}
service {
    input {

```

```

        post-service-filter filter-name;
        service-set service-set-name <service-filter filter-name>;
    }
    output {
        service-set service-set-name <service-filter filter-name>;
    }
}
service-name-table table-name
targeted-options {
    backup backup;
    group group;
    primary primary;
    weight ($junos-interface-target-weight | weight-value);
}
(translate-discard-eligible | no-translate-discard-eligible);
(translate-fecn-and-becn | no-translate-fecn-and-becn);
translate-plp-control-word-de;
unnumbered-address interface-name destination address destination-profile profile-name;
vlan-id number;
vlan-id-list [number number-number];
address address {
    arp ip-address (mac | multicast-mac) mac-address <publish>;
    broadcast address;
    destination address;
    destination-profile name;
    eui-64;
    primary-only;
    multipoint-destination address {
        dlci dlci-identifier;
        epd-threshold cells <plp1 cells>;
        inverse-arp;
        oam-liveness {
            up-count cells;
            down-count cells;
        }
        oam-period (disable | seconds);
        shaping {
            (cbr rate | rtvbr burst length peak rate sustained rate | vbr burst length
peak rate sustained rate);
            queue-length number;
        }
        vci vpi-identifier.vci-identifier;
    }
}

```

```

preferred;
primary;
(vrrp-group | vrrp-inet6-group) group-number {
    (accept-data | no-accept-data);
    advertise-interval seconds;
    authentication-type authentication;
    authentication-key key;
    fast-interval milliseconds;
    (preempt | no-preempt) {
        hold-time seconds;
    }
    priority number;
    track {
        interface interface-name {
            bandwidth-threshold bits-per-second priority-cost number;
        }
        priority-hold-time seconds;
        route ip-address/prefix-length routing-instance instance-name priority-
cost cost;
    }
    virtual-address [addresses];
    virtual-link-local-address ipv6-address;
    vrrp-inherit-from {
        active-interface interface-name;
        active-group group-number;
    }
}
}
}
}

```

## Hierarchy Level

```

[edit interfaces interface-name],
[edit logical-systems logical-system-name interfaces interface-name],
[edit interfaces interface-set interface-set-name interface interface-name]

```

## Description

Configure a logical interface on the physical device. You must configure a logical interface to be able to use the physical device.

## Options

*logical-unit-number*—Number of the logical unit.

- **Range:** 0 through 1,073,741,823 for demux, PPPoE, and pseudowire static interfaces. 0 through 16,385 for all other static interface types.

*etree-ac-role* (leaf | root)—To configure an interface as either leaf or root.

The remaining statements are explained separately. Search for a statement in [CLI Explorer](#) or click a linked statement in the Syntax section for details.

## Required Privilege Level

*interface*—To view this statement in the configuration.

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

## Release Information

Statement introduced before Junos OS Release 7.4.

Range increased for static pseudowire interfaces to 1,073,741,823 in Junos OS Release 18.3R1.

## RELATED DOCUMENTATION

[Configuring Logical Interface Properties](#)

[Junos OS Services Interfaces Library for Routing Devices](#)



## vlan-id-list (Interface in Bridge Domain)

### IN THIS SECTION

- [Syntax | 153](#)
- [Hierarchy Level | 153](#)
- [Description | 153](#)
- [Options | 153](#)
- [Required Privilege Level | 154](#)
- [Release Information | 154](#)

### Syntax

```
vlan-id-list [ number number-number ];
```

### Hierarchy Level

```
[edit interfaces interface-name unit logical-unit-number family bridge],  
[edit logical-systems logical-system-name interfaces interface-name unit logical-unit-number  
family bridge]
```

### Description

Configure a logical interface to forward packets and learn MAC addresses within each bridge domain configured with a VLAN ID that matches a VLAN ID specified in the list. VLAN IDs can be entered individually using a space to separate each ID, entered as an inclusive list separating the starting VLAN ID and ending VLAN ID with a hyphen, or a combination of both.

### Options

*number number*—Individual VLAN IDs separated by a space.

*number-number*—Starting VLAN ID and ending VLAN ID in an inclusive range.

- **Range:** 1 through 4095

### Required Privilege Level

interface—To view this statement in the configuration.

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

### Release Information

Statement introduced in Junos OS Release 9.2.

Statement introduced in Junos OS Release 15.1.

### RELATED DOCUMENTATION

---

[Configuring a Logical Interface for Trunk Mode](#)

---

[Configuring the VLAN ID List for a Trunk Interface](#)

---

[Tunnel Services Overview](#)

---

[Tunnel Interface Configuration on MX Series Routers Overview](#)

## CHAPTER 4

# OVSDB Monitoring Commands

**IN THIS CHAPTER**

- [show bridge domain | 155](#)
- [show ovssdb controller | 158](#)
- [show ovssdb interface | 161](#)
- [show ovssdb logical-switch | 164](#)
- [show ovssdb mac | 167](#)
- [show ovssdb statistics interface | 173](#)
- [show ovssdb virtual-tunnel-end-point | 175](#)
- [show vpls mac-table | 178](#)
- [Verifying That a Logical Switch and Corresponding Junos OS OVSDb-Managed VXLAN Are Working Properly | 186](#)

## show bridge domain

**IN THIS SECTION**

- [Syntax | 156](#)
- [Description | 156](#)
- [Options | 156](#)
- [Required Privilege Level | 156](#)
- [Sample Output | 156](#)
- [Release Information | 158](#)

Syntax

```
show bridge domain
<brief | detail | extensive>
<bridge-domain (all | domain-name)>
<instance instance-name>
<operational>
```

Description

(MX Series routers only) Display bridge domain information.

Options

<b>none</b>	Display information for all bridge domains.
<b>brief   detail   extensive</b>	(Optional) Display the specified level of output.
<b>bridge-domain (all   <i>domain-name</i>)</b>	(Optional) Display information about all bridge domains or the specified bridge domain.
<b>instance <i>instance-name</i></b>	(Optional) Display information for the specified routing instance.
<b>operational</b>	(Optional) Display information for the operational routing instances.

Required Privilege Level

view

Sample Output

show bridge domain

```
user@host> show bridge domain
Instance      Bridging Domain      Type      Active
Primary Table
vs1           vlan100              bridge
              bridge.0              2
```

vs1	vlan200	bridge	
	bridge.0		0

**show bridge domain brief**

```
user@host> show bridge domain brief
```

Instance	Bridging Domain	Type	Active
	Primary Table		
vs1	vlan100	bridge	2
	bridge.0		2
vs1	vlan200	bridge	0
	bridge.0		0

**show bridge domain detail**

```
user@host> show bridge domain detail
```

Routing Instance:vs1	
Bridging Domain:vlan100	
Router ID: 0.0.0.0	
Type: bridge	State: Active
Interfaces:	
ge-11/0/3.0	
ge-11/1/4.100	
ge-11/1/1.100	
ge-11/1/0.100	
xe-10/2/0.100	
xe-10/0/0.100	
Tables:	
bridge.0	: 2 macs (2 active)

```
Routing Instance:vs1
```

Bridging Domain:vlan200	
Router ID: 0.0.0.0	
Type: bridge	State: Active
Interfaces:	
ge-11/1/0.200	
ge-11/1/1.200	
ge-11/1/4.200	
xe-10/0/0.200	
xe-10/2/0.200	
Tables:	

```
bridge.0          : 0 macs (0 active)
```

## Release Information

Command introduced in Junos OS Release 8.4.

## show ovsdb controller

### IN THIS SECTION

- [Syntax | 158](#)
- [Description | 158](#)
- [Options | 159](#)
- [Required Privilege Level | 159](#)
- [Output Fields | 159](#)
- [Sample Output | 160](#)
- [Release Information | 161](#)

## Syntax

```
show ovsdb controller
<address ip-address>
```

## Description

Display information and connection status for software-defined networking (SDN) controllers to which the Juniper Networks device is connected.

# Options

- none** Display information about all SDN controllers to which the Juniper Networks device is connected.
- address *ip-address*** Display information about the SDN controller at the specified IP address.

# Required Privilege Level

admin

# Output Fields

Table 11 on page 159 lists the output fields for the `show ovssdb controller` command. Output fields are listed in the approximate order in which they appear.

**Table 11: show ovssdb controller Output Fields**

Field Name	Field Description
Controller IP address	IP address of the SDN controller to which the Juniper Networks device is connected.
Controller protocol	Protocol used by the Juniper Networks device to initiate the connection.
Controller port	Port to which the Juniper Networks device is connected.
Controller connection	State of the connection with the SDN controller.
Controller seconds-since-connect	Number of seconds since the connection with the SDN controller was established.
Controller seconds-since-disconnect	Number of seconds since the connection with the SDN controller was dropped.
Controller connection status	Status of the connection with the SDN controller.

## Sample Output

### show ovssdb controller

```
user@host> show ovssdb controller
VTEP controller information:
Controller IP address: 10.168.66.189
Controller protocol: ssl
Controller port: 6632
Controller connection: up
Controller seconds-since-connect: 56290
Controller seconds-since-disconnect: 0
Controller connection status: active

Controller IP address: 10.168.181.54
Controller protocol: ssl
Controller port: 6632
Controller connection: up
Controller seconds-since-connect: 56292
Controller seconds-since-disconnect: 0
Controller connection status: active

Controller IP address: 10.168.182.45
Controller protocol: ssl
Controller port: 6632
Controller connection: up
Controller seconds-since-connect: 56292
Controller seconds-since-disconnect: 0
Controller connection status: active
```

### show ovssdb controller address

```
user@host> show ovssdb controller address 10.168.182.45
VTEP controller information:
Controller IP address: 10.168.182.45
Controller protocol: ssl
Controller port: 6632
Controller connection: up
Controller seconds-since-connect: 56347
```



```
Controller seconds-since-disconnect: 0
Controller connection status: active
```

## Release Information

Command introduced in Junos OS Release 14.1R2.

## RELATED DOCUMENTATION

[Setting Up the OVSDb Protocol on Juniper Networks Devices that Support Manual Configuration of VXLANs | 24](#)

*Setting Up OVSDb on Juniper Networks Devices That Support the Dynamic Configuration of VXLANs*

[Understanding How to Set Up OVSDb Connections on a Juniper Networks Device | 7](#)

## show ovsdb interface

### IN THIS SECTION

- [Syntax | 161](#)
- [Description | 162](#)
- [Options | 162](#)
- [Required Privilege Level | 162](#)
- [Output Fields | 162](#)
- [Sample Output | 163](#)
- [Release Information | 163](#)

## Syntax

```
show ovsdb interface
<interface-name>
```

# Description

Display information about *Open vSwitch Database (OVSDB)*-managed interfaces configured by using the interfaces *interface-name* statement in the [edit protocols ovssdb] hierarchy.

# Options

- none** Display information about all OVSDB-managed interfaces.
- interface-name*** Display information about the specified OVSDB-managed interface.

# Required Privilege Level

admin

# Output Fields

Table 12 on page 162 lists the output fields for the show ovssdb interface command. Output fields are listed in the approximate order in which they appear.

Table 12: show ovssdb interface Output Fields

Field Name	Field Description
Interface	Name of interface.
VLAN ID	ID of Virtual Extensible LAN (VXLAN) with which the interface is associated. <b>NOTE:</b> This field is not supported by MX Series routers or EX9200 switches.
Bridge domain or VLAN	Bridge domain or VLAN under which the VXLAN is created. <b>NOTE:</b> This field is not supported by MX Series routers or EX9200 switches.

## Sample Output

### show ovsdb interface

```
user@host> show ovsdb interface
Interface          VLAN ID          Bridge-domain
ge-7/0/9.0
ge-7/0/9.1
irb.11
irb.12
irb.2
irb.3
xe-10/3/0.0
xe-10/3/0.1
```

### show ovsdb (Specific Interface)

```
user@host> show ovsdb interface ge-7/0/9.0
Interface          VLAN ID          Bridge-domain
ge-7/0/9.0
```

## Release Information

Command introduced in Junos OS Release 14.1R2.

## RELATED DOCUMENTATION

[Setting Up the OVSDb Protocol on Juniper Networks Devices that Support Manual Configuration of VXLANs | 24](#)

*Setting Up OVSDb on Juniper Networks Devices That Support the Dynamic Configuration of VXLANs*

## show ovssdb logical-switch

### IN THIS SECTION

- [Syntax | 164](#)
- [Description | 164](#)
- [Options | 165](#)
- [Required Privilege Level | 165](#)
- [Output Fields | 165](#)
- [Sample Output | 166](#)
- [Release Information | 167](#)

### Syntax

```
show ovssdb logical-switch  
<logical-switch-name>
```

### Description

**NOTE:** In the Open vSwitch Database (OVSSDB) schema for physical devices, the logical switch table stores information about the Layer 2 broadcast domain that you configured in a VMware NSX or Contrail environment. In the NSX environment, the Layer 2 broadcast domain is known as a *logical switch*, while in the Contrail environment, the domain is known as a *virtual network*. In the context of the `show ovssdb logical-switch` command, the term *logical switch* refers to the logical switch or virtual network that was configured in the NSX or Contrail environments, respectively, and the corresponding configuration that was pushed to the OVSSDB schema.

Display information about logical switches and the corresponding Virtual Extensible LANs (VXLANs), which were configured on the Juniper Networks device.

In the command output, each logical switch is identified by a universally unique identifier (UUID), which in the context of this command, is also known as a logical switch name.

The `show ovssdb logical-switch` command displays the state of the logical switch (Flags), which can be one of the following:

Created by Controller	A logical switch is configured. However, a corresponding VXLAN is not yet configured. In this state, the logical switch and corresponding VXLAN are not yet operational.
Created by L2ALD	A VXLAN is configured. However, a corresponding logical switch is not yet configured. In this state, the logical switch and corresponding VXLAN are not yet operational.
Created by both	A logical switch and a corresponding VXLAN are configured. In this state, the logical switch and corresponding VXLAN are operational.
Tunnel key mismatch	The VNIs specified in the logical switch and corresponding VXLAN configurations do not match. In this state, the logical switch and corresponding VXLAN are not yet operational.

Options

<code>none</code>	Display information about all logical switches that are present in the OVSSDB schema for physical devices.
<i><code>logical-switch-name</code></i>	Display information about the specified logical switch.

Required Privilege Level

admin

Output Fields

[Table 13 on page 166](#) lists the output fields for the `show ovssdb logical-switch` command. Output fields are listed in the approximate order in which they appear.

**Table 13: show ovssdb logical-switch Output Fields**

Field Name	Field Description
Logical Switch Name	UUID that is automatically generated and assigned to the logical switch. When you configure the corresponding VXLAN in the Junos OS CLI, you must specify the same UUID as the VXLAN name.
Flags	State of the logical switch. For possible states, see the Description section of this topic.
VNI	VNI that is configured for the logical switch and corresponding VXLAN.
Num of Remote MAC	The total number of remote media access control (MAC) addresses associated with the logical switch. These addresses are learned by software and hardware <i>virtual tunnel endpoints (VTEPs)</i> .
Num of Local MAC	The total number of local MAC addresses associated with the logical switch. <i>Local MAC addresses</i> are addresses learned on the local physical ports.

## Sample Output

### show ovssdb logical-switch

```

user@host> show ovssdb logical-switch
Logical switch information:
Logical Switch Name: 24a76aff-7e61-4520-a78d-3eca26ad7510
Flags: Created by both
VNI: 3
Num of Remote MAC: 13
Num of Local MAC: 12
Logical Switch Name: 9b4f880e-dac8-4612-a832-97ad9dec270f
Flags: Created by Controller
VNI: 50
Num of Remote MAC: 0
Num of Local MAC: 0
Logical Switch Name: bc0da2da-6c16-44bf-b655-442484294ded
Flags: Created by Controller
VNI: 51

```

```
Num of Remote MAC: 0
Num of Local MAC: 0
```

## show ovssdb logical-switch (Specific Logical Switch)

```
user@host> show ovssdb logical-switch 24a76aff-7e61-4520-a78d-3eca26ad7510
Logical switch information:
Logical Switch Name: 24a76aff-7e61-4520-a78d-3eca26ad7510
Flags: Created by both
VNI: 3
Num of Remote MAC: 13
Num of Local MAC: 12
```

## Release Information

Command introduced in Junos OS Release 14.1R2.

## RELATED DOCUMENTATION

*OVSSDB Schema for Physical Devices*

[Troubleshooting a Nonoperational Logical Switch and Corresponding Junos OS OVSSDB-Managed VXLAN | 188](#)

## show ovssdb mac

### IN THIS SECTION

- [Syntax | 168](#)
- [Description | 168](#)
- [Options | 168](#)
- [Required Privilege Level | 169](#)
- [Output Fields | 169](#)
- [Sample Output | 170](#)

## Syntax

```
show ovsdb mac
<address mac-address>
<local>
<logical-switch logical-switch-uuid>
<multicast>
<remote>
<unicast>
```

## Description

Display media access control (MAC) addresses, as well as information about the MAC addresses, learned by a Juniper Networks device that functions as a hardware *virtual tunnel endpoint (VTEP)*. Using the *Open vSwitch Database (OVSDB)* management protocol, this hardware VTEP can learn about MAC addresses directly or from other software or hardware VTEPs. The MAC addresses learned directly by the hardware VTEP are known as *local addresses*, while the addresses learned from other software or hardware VTEPs are known as *remote addresses*.

## Options

Use one or more of the following options to display a more specific list of MAC addresses and information about the MAC addresses. For example, to display a list of local unicast MAC addresses, you can issue the `show ovsdb mac local unicast` command.

<b>none</b>	Display all MAC addresses, which includes all local, remote, unicast, and multicast addresses associated with all logical switches.
<b>address <i>mac-address</i></b>	Display the specified MAC address.
<b>count</b>	(All Juniper Networks devices that support OVSDB except EX9200 switches) Display the number of MAC addresses learned by the Juniper Networks device. Using this option alone, the number includes all local, remote, unicast, and multicast MAC addresses associated with all logical switches in the logical switch table of the OVSDB schema for physical devices. You can use this option with one or more of the other



options to display a more specific count of MAC addresses. For example, to display the number of local and remote unicast MAC addresses, you can issue the `show ovssdb mac count local remote unicast` command.

<b>local</b>	Display all local MAC addresses.
<b>logical-switch</b> <i>logical-switch-uuid</i>	Display all MAC addresses associated with the specified logical switch in the logical switch table of the OVSSDB schema for physical devices.
<b>multicast</b>	Display all multicast MAC addresses.
<b>remote</b>	Display all remote MAC addresses.
<b>unicast</b>	Display all unicast MAC addresses.

### Required Privilege Level

admin

### Output Fields

Table 14 on page 169 lists the output fields for the `show ovssdb mac` command. Output fields are listed in the approximate order in which they appear.

**Table 14: show ovssdb mac Output Fields**

Field Name	Field Description
Logical Switch Name	Universally unique identifier (UUID) of the logical switch.
MAC Address	MAC addresses of <i>virtual machines (VMs)</i> .
IP Address	IP address of VMs.  <b>NOTE:</b> If the IP addresses of VMs are not published by the SDN controller, this field displays 0.0.0.0.
Encapsulation	Encapsulation type.

Table 14: show ovssdb mac Output Fields (Continued)

Field Name	Field Description
VTEP Address	IP address of the hardware or software VTEP from which the MAC address was learned. Further, this VTEP can forward VM traffic to the associated host.
MAC Count	<p><b>NOTE:</b> This field is supported by all Juniper Networks devices that support OVSSDB except EX9200 switches.</p> <p>Number of all or specified MAC addresses learned by the Juniper Networks device.</p>

## Sample Output

### show ovssdb mac

```
user@host> show ovssdb mac
```

```
Logical Switch Name: 24a76aff-7e61-4520-a78d-3eca26ad7510
```

Mac Address	IP Address	Encapsulation	Vtep Address
02:00:00:00:03:01	0.0.0.0	Vxlan over Ipv4	10.255.18.22
02:00:00:00:03:02	0.0.0.0	Vxlan over Ipv4	10.255.18.22
02:00:00:00:03:03	0.0.0.0	Vxlan over Ipv4	10.255.18.22
02:00:00:00:03:04	0.0.0.0	Vxlan over Ipv4	10.255.18.22
02:00:00:00:03:05	0.0.0.0	Vxlan over Ipv4	10.255.18.22
04:00:00:00:03:05	0.0.0.0	Vxlan over Ipv4	10.255.18.22
06:00:00:00:03:01	0.0.0.0	Vxlan over Ipv4	10.255.18.22
06:00:00:00:03:02	0.0.0.0	Vxlan over Ipv4	10.255.18.22
06:00:00:00:03:03	0.0.0.0	Vxlan over Ipv4	10.255.18.22
06:00:00:00:03:04	0.0.0.0	Vxlan over Ipv4	10.255.18.22
06:00:00:00:03:05	0.0.0.0	Vxlan over Ipv4	10.255.18.22
40:b4:f0:06:6f:f0	0.0.0.0	Vxlan over Ipv4	10.255.18.22
ff:ff:ff:ff:ff:ff	0.0.0.0	Vxlan over Ipv4	10.100.100.1

```
Logical Switch Name: bf6d4fd4-f5f6-430c-8c37-4033ef1c55ab
```

Mac Address	IP Address	Encapsulation	Vtep Address
02:00:00:00:11:01	0.0.0.0	Vxlan over Ipv4	10.1.1.29
02:00:00:00:11:02	0.0.0.0	Vxlan over Ipv4	10.1.1.29
02:00:00:00:11:03	0.0.0.0	Vxlan over Ipv4	10.1.1.29

02:00:00:00:11:04	0.0.0.0	Vxlan over Ipv4	10.1.1.29
02:00:00:00:11:05	0.0.0.0	Vxlan over Ipv4	10.1.1.29
04:00:00:00:11:05	0.0.0.0	Vxlan over Ipv4	10.1.1.29
06:00:00:00:11:01	0.0.0.0	Vxlan over Ipv4	10.1.1.29
06:00:00:00:11:02	0.0.0.0	Vxlan over Ipv4	10.1.1.29
06:00:00:00:11:03	0.0.0.0	Vxlan over Ipv4	10.1.1.29
06:00:00:00:11:04	0.0.0.0	Vxlan over Ipv4	10.1.1.29
06:00:00:00:11:05	0.0.0.0	Vxlan over Ipv4	10.1.1.29
40:b4:f0:06:6f:f0	0.0.0.0	Vxlan over Ipv4	10.1.1.29
00:23:9c:5e:a7:f0	0.0.0.0	Vxlan over Ipv4	10.255.18.22
08:00:00:00:11:01	0.0.0.0	Vxlan over Ipv4	10.255.18.22
08:00:00:00:11:02	0.0.0.0	Vxlan over Ipv4	10.255.18.22
08:00:00:00:11:03	0.0.0.0	Vxlan over Ipv4	10.255.18.22
08:00:00:00:11:04	0.0.0.0	Vxlan over Ipv4	10.255.18.22
08:00:00:00:11:05	0.0.0.0	Vxlan over Ipv4	10.255.18.22
ff:ff:ff:ff:ff:ff	0.0.0.0	Vxlan over Ipv4	10.110.110.1
...			

### show ovssdb mac address

```
user@host> show ovssdb mac address 02:00:00:00:03:01
```

Mac	IP	Encapsulation	Vtep
Address	Address		Address
02:00:00:00:03:01	0.0.0.0	Vxlan over Ipv4	10.255.18.22

### show ovssdb mac logical-switch

```
user@host> show ovssdb mac logical-switch bf6d4fd4-f5f6-430c-8c37-4033ef1c55ab
```

Logical Switch Name: bf6d4fd4-f5f6-430c-8c37-4033ef1c55ab

Mac	IP	Encapsulation	Vtep
Address	Address		Address
02:00:00:00:11:01	0.0.0.0	Vxlan over Ipv4	10.1.1.29
02:00:00:00:11:02	0.0.0.0	Vxlan over Ipv4	10.1.1.29
02:00:00:00:11:03	0.0.0.0	Vxlan over Ipv4	10.1.1.29
02:00:00:00:11:04	0.0.0.0	Vxlan over Ipv4	10.1.1.29
02:00:00:00:11:05	0.0.0.0	Vxlan over Ipv4	10.1.1.29
04:00:00:00:11:05	0.0.0.0	Vxlan over Ipv4	10.1.1.29
06:00:00:00:11:01	0.0.0.0	Vxlan over Ipv4	10.1.1.29
06:00:00:00:11:02	0.0.0.0	Vxlan over Ipv4	10.1.1.29

06:00:00:00:11:03	0.0.0.0	Vxlan over Ipv4	10.1.1.29
06:00:00:00:11:04	0.0.0.0	Vxlan over Ipv4	10.1.1.29
06:00:00:00:11:05	0.0.0.0	Vxlan over Ipv4	10.1.1.29
40:b4:f0:06:6f:f0	0.0.0.0	Vxlan over Ipv4	10.1.1.29
00:23:9c:5e:a7:f0	0.0.0.0	Vxlan over Ipv4	10.255.18.22
08:00:00:00:11:01	0.0.0.0	Vxlan over Ipv4	10.255.18.22
08:00:00:00:11:02	0.0.0.0	Vxlan over Ipv4	10.255.18.22
08:00:00:00:11:03	0.0.0.0	Vxlan over Ipv4	10.255.18.22
08:00:00:00:11:04	0.0.0.0	Vxlan over Ipv4	10.255.18.22
08:00:00:00:11:05	0.0.0.0	Vxlan over Ipv4	10.255.18.22
ff:ff:ff:ff:ff:ff	0.0.0.0	Vxlan over Ipv4	10.110.110.1

### show ovsdb mac local unicast

```
user@host> show ovsdb mac local unicast
```

```
Logical Switch Name: 24a76aff-7e61-4520-a78d-3eca26ad7510
```

Mac Address	IP Address	Encapsulation	Vtep Address
02:00:00:00:03:01	0.0.0.0	Vxlan over Ipv4	10.255.181.72
02:00:00:00:03:02	0.0.0.0	Vxlan over Ipv4	10.255.181.72
02:00:00:00:03:03	0.0.0.0	Vxlan over Ipv4	10.255.181.72
02:00:00:00:03:04	0.0.0.0	Vxlan over Ipv4	10.255.181.72
02:00:00:00:03:05	0.0.0.0	Vxlan over Ipv4	10.255.181.72
04:00:00:00:03:05	0.0.0.0	Vxlan over Ipv4	10.255.181.72
06:00:00:00:03:01	0.0.0.0	Vxlan over Ipv4	10.255.181.72
06:00:00:00:03:02	0.0.0.0	Vxlan over Ipv4	10.255.181.72
06:00:00:00:03:03	0.0.0.0	Vxlan over Ipv4	10.255.181.72
06:00:00:00:03:04	0.0.0.0	Vxlan over Ipv4	10.255.181.72
06:00:00:00:03:05	0.0.0.0	Vxlan over Ipv4	10.255.181.72
40:b4:f0:06:6f:f0	0.0.0.0	Vxlan over Ipv4	10.255.181.72

...

### show ovsdb mac (Count of All Local, Remote, Unicast, and Multicast MAC Addresses for All Logical Switches)

```
user@host> show ovsdb mac count
```

```
MAC count: 6877
```

## Release Information

Command introduced in Junos OS Release 14.1R2.

## RELATED DOCUMENTATION

| *OVSDB Schema for Physical Devices*

## show ovsdb statistics interface

### IN THIS SECTION

- [Syntax | 173](#)
- [Description | 173](#)
- [Options | 174](#)
- [Required Privilege Level | 174](#)
- [Output Fields | 174](#)
- [Sample Output | 174](#)
- [Release Information | 175](#)

## Syntax

```
show ovsdb statistics interface  
  <interface-name>
```

## Description

Display statistics for *Open vSwitch Database (OVSDB)*-managed interfaces configured by using the `interfaces interface-name` statement in the `[edit protocols ovsdb]` hierarchy.

When an interface is configured as OVSDB-managed, the collection of statistics for that interface begins, and the statistics displayed at any given time reflects the data collected up to that point.

## Options

- none**                      Display statistics for all configured OVSDb-managed interfaces.
- interface-name***        Display statistics for the specified interface.

## Required Privilege Level

admin

## Output Fields

[Table 15 on page 174](#) lists the output fields for the `show ovssdb statistics interface` command. Output fields are listed in the approximate order in which they appear.

**Table 15: show ovssdb statistics interface Output Fields**

Field Name	Field Description
Num of rx pkts	Number of packets received by the interface.
Num of tx pkts	Number of packets sent by the interface.
Num of rx bytes	Number of bytes received by the interface.
Num of tx bytes	Number of bytes sent by the interface.

## Sample Output

### show ovssdb statistics interface

```

user@host> show ovssdb statistics interface
Interface Name: ge-7/0/9.0
Num of rx pkts: 945                      Num of tx pkts: 113280890
Num of rx bytes: 56700                    Num of tx bytes: 57531319540
Interface Name: ge-7/0/10.0
Num of rx pkts: 459                      Num of tx pkts: 473840856

```

```

Num of rx bytes: 84747           Num of tx bytes: 45830738532
Interface Name: ge-7/0/11.0
Num of rx pkts: 305             Num of tx pkts: 367483456
Num of rx bytes: 98974          Num of tx bytes: 33495468092

```

### show ovssdb statistics interface (Specific Interface)

```

user@host> show ovssdb statistics interface ge-7/0/9.0
Interface Name: ge-7/0/9.0
Num of rx pkts: 945             Num of tx pkts: 113280890
Num of rx bytes: 56700          Num of tx bytes: 57531319540

```

### Release Information

Command introduced in Junos OS Release 14.1R2.

### RELATED DOCUMENTATION

[interfaces \(OVSSDB\)](#) | [126](#)

## show ovssdb virtual-tunnel-end-point

### IN THIS SECTION

- [Syntax](#) | [176](#)
- [Description](#) | [176](#)
- [Options](#) | [176](#)
- [Required Privilege Level](#) | [176](#)
- [Output Fields](#) | [176](#)
- [Sample Output](#) | [177](#)
- [Release Information](#) | [178](#)

Syntax

```
show ovsdb virtual-tunnel-end-point
address <ip-address>
encapsulation <encapsulation-type>
```

Description

Display information about the following entities that the Juniper Networks device has learned:

- Other Juniper Networks devices that function as hardware *virtual tunnel endpoints (VTEPs)*
- Software VTEPs
- Service nodes
- Top-of-rack service nodes (TSNs)

Options

<b>none</b>	Display information about all VTEPs, service nodes, and TSNs that the Juniper Networks device has learned.
<b>address</b> <i>ip-address</i>	Display information about the entity with the specified IP address.
<b>encapsulation</b> <i>encapsulation-type</i>	Display information about all entities with the specified encapsulation type.

Required Privilege Level

admin

Output Fields

[Table 16 on page 177](#) lists the output fields for the `show ovsdb virtual-tunnel-end-point` command. Output fields are listed in the approximate order in which they appear.



**Table 16: show ovssdb virtual-tunnel-end-point Output Fields**

Field Name	Field Description
Encapsulation	Encapsulation type of entity.
IP Address	IP address of entity.
Num of MACs	Number of media access control (MAC) addresses learned by the entity.

## Sample Output

### show ovssdb virtual-tunnel-end-point

```
user@host> show ovssdb virtual-tunnel-end-point
```

Encapsulation	Ip Address	Num of MAC's
VXLAN over IPv4	10.255.181.43	24
VXLAN over IPv4	10.255.181.50	12
VXLAN over IPv4	10.255.181.72	24

### show ovssdb virtual-tunnel-end-point address (Specific Address)

```
user@host> show ovssdb virtual-tunnel-end-point address 10.255.181.43
```

Encapsulation	Ip Address	Num of MAC's
VXLAN over IPv4	10.255.181.43	24

### show ovssdb virtual-tunnel-end-point encapsulation (Specific Encapsulation)

```
user@host> show ovssdb virtual-tunnel-end-point encapsulation vxlan-over-ipv4
```

Encapsulation	Ip Address	Num of MAC's
VXLAN over IPv4	10.255.181.43	24

VXLAN over IPv4	10.255.181.50	12
VXLAN over IPv4	10.255.181.72	24

### show ovsdb virtual-tunnel-end-point address (Specific Address) encapsulation (Specific Encapsulation)

```
user@host> show ovsdb virtual-tunnel-end-point address 10.255.181.43 encapsulation vxlan-over-ipv4
```

Encapsulation	Ip Address	Num of MAC's
VXLAN over IPv4	10.255.181.43	24

### Release Information

Command introduced in Junos OS Release 14.1R2.

## show vpls mac-table

#### IN THIS SECTION

- [Syntax | 178](#)
- [Description | 179](#)
- [Options | 179](#)
- [Required Privilege Level | 179](#)
- [Output Fields | 180](#)
- [Sample Output | 181](#)
- [Release Information | 186](#)

### Syntax

```
show vpls mac-table
<age>
```

```
<brief | detail | extensive | summary>
<bridge-domain bridge-domain-name>
<instance instance-name>
<interface interface-name>
<logical-system (all | logical-system-name)>
<mac-address>
<vlan-id vlan-id-number>
```

### Description

Display learned virtual private LAN service (VPLS) media access control (MAC) address information.

### Options

<b>none</b>	Display all learned VPLS MAC address information.
<b>age</b>	(Optional) Display age of a single mac-address.
<b>brief   detail   extensive   summary</b>	(Optional) Display the specified level of output.
<b>bridge-domain <i>bridge-domain-name</i></b>	(Optional) Display learned VPLS MAC addresses for the specified bridge domain.
<b>instance <i>instance-name</i></b>	(Optional) Display learned VPLS MAC addresses for the specified instance.
<b>interface <i>interface-name</i></b>	(Optional) Display learned VPLS MAC addresses for the specified instance.
<b>logical-system (all   <i>logical-system-name</i>)</b>	(Optional) Display learned VPLS MAC addresses for all logical systems or for the specified logical system.
<b><i>mac-address</i></b>	(Optional) Display the specified learned VPLS MAC address information..
<b>vlan-id <i>vlan-id-number</i></b>	(Optional) Display learned VPLS MAC addresses for the specified VLAN.

### Required Privilege Level

view

## Output Fields

Table 17 on page 180 describes the output fields for the `show vpls mac-table` command. Output fields are listed in the approximate order in which they appear.

**Table 17: show vpls mac-table Output fields**

Field Name	Field Description
Age	Age of a single mac-address.
Routing instance	Name of the routing instance.
Bridging domain	Name of the bridging domain.
MAC address	MAC address or addresses learned on a logical interface.
MAC flags	Status of MAC address learning properties for each interface: <ul style="list-style-type: none"> <li>• S—Static MAC address configured.</li> <li>• D—Dynamic MAC address learned.</li> <li>• SE—MAC accounting is enabled.</li> <li>• NM—Nonconfigured MAC.</li> </ul>
Logical interface	Name of the logical interface.
MAC count	Number of MAC addresses learned on a specific routing instance or interface.
Learning interface	Logical interface or logical Label Switched Interface (LSI) the address is learned on.
Base learning interface	Base learning interface of the MAC address. This field is introduced in Junos OS Release 14.2.
Learn VLAN ID/ VLAN	VLAN ID of the routing instance or bridge domain in which the MAC address was learned.

**Table 17: show vpls mac-table Output fields (Continued)**

Field Name	Field Description
VXLAN ID/VXLAN	VXLAN Network Identifier (VNI)
Layer 2 flags	Debugging flags signifying that the MAC address is present in various lists.
Epoch	Spanning Tree Protocol epoch number identifying when the MAC address was learned. Used for debugging.
Sequence number	Sequence number assigned to this MAC address. Used for debugging.
Learning mask	Mask of Packet Forwarding Engines where this MAC address was learned. Used for debugging.
IPC generation	Creation time of the logical interface when this MAC address was learned. Used for debugging.

## Sample Output

### show vpls mac-table

```
user@host> show vpls mac-table
MAC flags (S -static MAC, D -dynamic MAC,
          SE -Statistics enabled, NM -Non configured MAC)
```

```
Routing instance : vpls_ldp1
```

```
VLAN : 223
```

```
MAC          MAC      Logical
address      flags    interface
00:00:5e:00:53:5d  D      ge-0/2/5.400
```

```
MAC flags (S -static MAC, D -dynamic MAC,
          SE -Statistics enabled, NM -Non configured MAC)
```

```
Routing instance : vpls_red
```

```
VLAN : 401
```

MAC address	MAC flags	Logical interface
00:00:5e:00:53:12	D	lsi.1051138
00:00:5e:00:53:f0	D	lsi.1051138

### show vpls mac-table (with Layer 2 Services over GRE Interfaces)

```
user@host> show vpls mac-table
MAC flags (S -static MAC, D -dynamic MAC, L -locally learned
          SE -Statistics enabled, NM -Non configured MAC, R -Remote PE MAC)

Routing instance : vpls_4site:1000
Bridging domain : __vpls_4site:1000__, MAC          MAC      Logical
address          flags      interface
00:00:5e:00:53:f4 D,SE    ge-4/2/0.1000
00:00:5e:00:53:33 D,SE    lsi.1052004
00:00:5e:00:53:32 D,SE    lsi.1048840
00:00:5e:00:53:14 D,SE    lsi.1052005
00:00:5e:00:53:f7 D,SE    gr-1/2/10.10
```

### show vpls mac-table (with VXLAN enabled)

```
user@host> show vpls mac-table
MAC flags (S -static MAC, D -dynamic MAC, L -locally learned
          SE -Statistics enabled, NM -Non configured MAC, R -Remote PE MAC)

Routing instance : vpls_4site:1000
Bridging domain : __vpls_4site:1000__, VLAN : 4094,4093
VXLAN: Id : 300, Multicast group: 233.252.0.1
MAC          MAC      Logical
address      flags      interface
00:00:5e:00:53:f4 D,SE    ge-4/2/0.1000
00:00:5e:00:53:33 D,SE    lsi.1052004
00:00:5e:00:53:32 D,SE    lsi.1048840
00:00:5e:00:53:14 D,SE    lsi.1052005
00:00:5e:00:53:f7 D,SE    vtep.1052010
00:00:5e:00:53:3f D,SE    vtep.1052011
```

**show vpls mac-table age (for GE interface)**

```
user@host> show vpls mac-table age 00:00:5e:00:53:1a instance vpls_instance_1
MAC Entry Age information
Current Age: 4 seconds
```

**show vpls mac-table age (for AE interface)**

```
user@host> show vpls mac-table age 000:00:5e:00:53:1a instance vpls_instance_1
MAC Entry Age information
Current Age on FPC1: 102 seconds
Current Age on FPC2: 94 seconds
```

**show vpls mac-table count**

```
user@host> show vpls mac-table count
0 MAC address learned in routing instance __example_private1__
```

MAC address count per interface within routing instance:

Logical interface	MAC count
lc-0/0/0.32769	0
lc-0/1/0.32769	0
lc-0/2/0.32769	0
lc-2/0/0.32769	0
lc-0/3/0.32769	0
lc-2/1/0.32769	0
lc-9/0/0.32769	0
lc-11/0/0.32769	0
lc-2/2/0.32769	0
lc-9/1/0.32769	0
lc-11/1/0.32769	0
lc-2/3/0.32769	0
lc-9/2/0.32769	0
lc-11/2/0.32769	0
lc-11/3/0.32769	0
lc-9/3/0.32769	0

MAC address count per learn VLAN within routing instance:

Learn VLAN ID	MAC count
---------------	-----------

```

0
0

1 MAC address learned in routing instance vpls_ldp1

MAC address count per interface within routing instance:
Logical interface      MAC count
lsi.1051137            0
ge-0/2/5.400           1

MAC address count per learn VLAN within routing instance:
Learn VLAN ID          MAC count
0                       1

1 MAC address learned in routing instance vpls_red

MAC address count per interface within routing instance:
Logical interface      MAC count
ge-0/2/5.300           1

MAC address count per learn VLAN within routing instance:
Learn VLAN ID          MAC count
0                       1

```

### show vpls mac-table detail

```

user@host> show vpls mac-table detail
MAC address: 00:00:5e:00:53:5d
Routing instance: vpls_ldp1
Learning interface: ge-0/2/5.400
Layer 2 flags: in_ifd, in_ifl, in_vlan, kernel
Epoch: 0                      Sequence number: 1
Learning mask: 0x1             IPC generation: 0

MAC address: 00:00:5e:00:53:5d
Routing instance: vpls_red
Learning interface: ge-0/2/5.300
Layer 2 flags: in_ifd, in_ifl, in_vlan, kernel
Epoch: 0                      Sequence number: 1
Learning mask: 0x1             IPC generation: 0

```



**show vpls mac-table extensive**

```
user@host> show vpls mac-table extensive
```

```
MAC address: 00:00:5e:00:53:00
```

```
Routing instance: vpls_1
```

```
Bridging domain: __vpls_1__, VLAN : NA
```

```
Learning interface: lsi.1049165
```

```
Base learning interface: lsi.1049165
```

```
Layer 2 flags: in_hash,in_ifd,in_ifl,in_vlan,in_rtt,kernel,in_ifbd
```

```
Epoch: 0                      Sequence number: 1
```

```
Learning mask: 0x00000001
```

```
MAC address: 00:00:5e:00:53:01
```

```
Routing instance: vpls_1
```

```
Bridging domain: __vpls_1__, VLAN : NA
```

```
Learning interface: lsi.1049165
```

```
Base learning interface: lsi.1049165
```

```
Layer 2 flags: in_hash,in_ifd,in_ifl,in_vlan,in_rtt,kernel,in_ifbd
```

```
Epoch: 0                      Sequence number: 1
```

```
Learning mask: 0x00000001
```

```
MAC address: 00:00:5e:00:53:02
```

```
Routing instance: vpls_1
```

```
Bridging domain: __vpls_1__, VLAN : NA
```

```
Learning interface: lsi.1049165
```

```
Base learning interface: lsi.1049165
```

```
Layer 2 flags: in_hash,in_ifd,in_ifl,in_vlan,in_rtt,kernel,in_ifbd
```

```
Epoch: 0                      Sequence number: 1
```

```
Learning mask: 0x00000001
```

```
MAC address: 00:00:5e:00:53:03
```

```
Routing instance: vpls_1
```

```
Bridging domain: __vpls_1__, VLAN : NA
```

```
Learning interface: lsi.1049165
```

```
Base learning interface: lsi.1049165
```

```
Layer 2 flags: in_hash,in_ifd,in_ifl,in_vlan,in_rtt,kernel,in_ifbd
```

```
Epoch: 0                      Sequence number: 1
```

```
Learning mask: 0x00000001
```

## Release Information

Command introduced in Junos OS Release 8.5.

## Verifying That a Logical Switch and Corresponding Junos OS OVSDb-Managed VXLAN Are Working Properly

### IN THIS SECTION

- [Purpose | 186](#)
- [Action | 186](#)
- [Meaning | 187](#)

### Purpose

Verify the following:

- A logical switch, which is configured in an NSX environment, or a virtual network, which is configured in a Contrail environment, is learning MAC addresses in their respective environments.
- The corresponding OVSDb-managed Virtual Extensible LAN (VXLAN), which is configured on a Juniper Networks device, is learning MAC addresses in the Junos OS environment.
- The logical switch or virtual network and OVSDb-managed VXLAN are exchanging the MAC addresses learned in their respective environments so that virtual and physical servers can communicate.

### Action

To verify that a logical switch or virtual network and its corresponding OVSDb-managed VXLAN are learning and exchanging MAC addresses in their respective environments, enter the `show ovssdb logical-switch operational mode` command.

```
user@device> show ovssdb logical-switch
Logical switch information:
Logical Switch Name: 28805c1d-0122-495d-85df-19abd647d772
```

```
Flags: Created by both
VNI: 100
Num of Remote MAC: 1
Num of Local MAC: 0
```

**NOTE:** In the Open vSwitch Database (OVSDB) schema for physical devices, the logical switch table stores information about the Layer 2 broadcast domain that you configured in a VMware NSX or Contrail environment. In the NSX environment, the Layer 2 broadcast domain is known as a *logical switch*, while in the Contrail environment, the domain is known as a *virtual network*. In the context of the `show ovssdb logical-switch` command, the term *logical switch* refers to the logical switch or virtual network that was configured in the NSX or Contrail environments, respectively, and the corresponding configuration that was pushed to the OVSDB schema.

## Meaning

The output in the Flags field (Created by both) indicates that the logical switch or virtual network and its corresponding OVSDB-managed VXLAN are both properly configured. In this state, the logical switch or virtual network and the VXLAN are learning and exchanging MAC addresses in their respective environments.

If the output in the Flags field displays a state other than Created by both, see ["Troubleshooting a Nonoperational Logical Switch and Corresponding Junos OS OVSDB-Managed VXLAN"](#) on page 188.

## RELATED DOCUMENTATION

| [show ovssdb logical-switch](#) | 164

## CHAPTER 5

# Troubleshooting OVSDB

**IN THIS CHAPTER**

- [Troubleshooting a Nonoperational Logical Switch and Corresponding Junos OS OVSDB-Managed VXLAN | 188](#)

## Troubleshooting a Nonoperational Logical Switch and Corresponding Junos OS OVSDB-Managed VXLAN

**IN THIS SECTION**

- [Problem | 188](#)
- [Cause | 189](#)
- [Solution | 189](#)

### Problem

#### Description

The Flags field in the `show ovssdb logical-switch operational mode` command output is one of the following:

- Created by Controller
- Created by L2ALD
- Tunnel key mismatch

## Cause

- If the Flags field displays `Created by Controller`, a logical switch is configured in the NSX environment or a virtual network is configured in the Contrail environment. However, an equivalent VXLAN is not configured or is improperly configured on the Juniper Networks device.
- If the Flags field displays `Created by L2ALD`, a VXLAN is configured on the Juniper Networks device. However, an equivalent logical switch is not configured in the NSX environment or an equivalent virtual network is not configured in the Contrail environment.
- If the Flags field displays `Tunnel key mismatch`, the VXLAN network identifier (VNI) specified in the logical switch configuration or the VXLAN identifier specified in the virtual network configuration do not match the VNI in the equivalent VXLAN configuration.

## Solution

If the Flags field displays `Created by Controller`, take the following action:

- On a QFX Series switch, verify that the `set switch-options ovssdb-managed` configuration command was issued in the Junos OS CLI. Issuing this command and committing the configuration enable the Juniper Networks device to dynamically create OVSDb-managed VXLANs.

Another possible cause is that the L2ALD daemon has become nonfunctional. If this is the case, wait for a few seconds, reissue the `show ovssdb logical-switch operational mode` command, and recheck the setting of the Flags field.

Another possible cause is that the Juniper Networks device dynamically configured the VXLAN and its associated logical interface, but there is an error in the configuration of these entities themselves or in an entity that was committed in the same transaction. If there is an issue with one or more of the configurations in a transaction, all configurations in the transaction, even the ones that are correctly configured, remain uncommitted and in a queue until you troubleshoot and resolve the configuration issues. As a result, the Juniper Networks device was unable to commit all configurations in the transaction. Starting with Junos OS Release 14.1X53-D26 for QFX5100 switches, Junos OS Release 15.1X53-D210 for QFX5110 and QFX5200 switches, and 18.1R1 for QFX5210 switches, you can enter the `show ovssdb commit failures operational mode` command to determine which configurations in a transaction are erroneous. After resolving the errors, enter the `clear ovssdb commit failures` command to remove the transaction from the queue and then retry committing all configurations in the transaction. Issues that can cause commitment errors include but are not limited to the detection of the same VXLAN name or VXLAN network identifier (VNI) in a dynamically configured VXLAN and in a VXLAN that was previously configured using the Junos OS CLI.

- On all other Juniper Networks devices that support VXLAN and OVSDb, determine whether a VXLAN equivalent to the logical switch configuration or virtual network configuration exists on the

device. If the VXLAN is not configured, configure it using the procedure in ["Configuring OVSDB-Managed VXLANs" on page 26](#). If a VXLAN is configured, check the VXLAN name to make sure that it is the same as the universally unique identifier (UUID) of the logical switch (NSX) or virtual network (Contrail) configuration. Also, check the VNI to make sure that the value is the same as the value in the logical switch (NSX) or virtual network (Contrail) configuration.

If the Flags field displays Created by L2ALD, take the following action:

- On a QFX Series switch, two issues exist. First, despite the fact that the Juniper Networks device dynamically creates OVSDB-managed VXLANs, this VXLAN was configured by using the Junos OS CLI. Second, a corresponding logical switch (NSX) or virtual network (Contrail) was not configured. To resolve both issues, configure a logical switch in the NSX environment or a virtual network in the Contrail environment. After the software-defined networking (SDN) controller pushes relevant logical switch or virtual network information to the Juniper Networks device, the device dynamically creates a corresponding VXLAN and deletes the VXLAN configured using the Junos OS CLI.
- On all other Juniper Networks devices that support VXLAN and OVSDB, determine whether an equivalent logical switch is configured in the NSX environment or a virtual network is configured in the Contrail environment. If a logical switch or virtual network is not configured, configure one, keeping in mind that a UUID is automatically generated for the logical switch or virtual network and that this UUID must be used as the name of the VXLAN. That is, the VXLAN name must be reconfigured with the logical switch or virtual network UUID.

Another possibility is that the logical switch or virtual network configuration might exist, but the UUID of the entity might not match the VXLAN name. In the NSX or Contrail environment, check for a logical switch or virtual network, respectively, that has the same configuration as the VXLAN but has a different UUID.

If the Flags field displays Tunnel key mismatch, take the following action:

- For a QFX Series switch, check the configuration of the VNI in the NSX environment or the VXLAN identifier in the Contrail environment to see whether it was changed after the Juniper Networks device dynamically created the equivalent VXLAN. If it was changed, update the VNI on the QFX Series switch using the Junos OS CLI.
- On all other Juniper Networks devices that support VXLAN and OVSDB, check the value of the VNI in the NSX environment or the VXLAN identifier in the Contrail environment and the Junos OS CLI. Change the incorrect value.

Release History Table

Release	Description
14.1X53-D26	Starting with Junos OS Release 14.1X53-D26 for QFX5100 switches, Junos OS Release 15.1X53-D210 for QFX5110 and QFX5200 switches, and 18.1R1 for QFX5210 switches, you can enter the <code>show ovssdb commit failures</code> operational mode command to determine which configurations in a transaction are erroneous.

RELATED DOCUMENTATION

<i>Understanding Dynamically Configured VXLANs in an OVSSDB Environment</i>
<a href="#">Understanding How to Manually Configure OVSSDB-Managed VXLANs   10</a>
<a href="#">show ovssdb logical-switch   164</a>
<i>show ovssdb commit failures</i>
<i>clear ovssdb commit failures</i>

# 2

PART

## VXLAN (Without a Controller)

---

[Using VXLAN Without a Controller | 193](#)

[VXLAN Configuration Statements | 219](#)

[VXLAN Monitoring Commands | 229](#)

---



## CHAPTER 6

# Using VXLAN Without a Controller

**IN THIS CHAPTER**

- Understanding VXLANs | 193
- PIM NSR and Unified ISSU Support for VXLAN Overview | 202
- Example: Manually Configuring VXLANs on MX Series Routers | 203

## Understanding VXLANs

**IN THIS SECTION**

- VXLAN Benefits | 194
- How Does VXLAN Work? | 195
- VXLAN Implementation Methods | 196
- Using QFX5100, QFX5110, QFX5120, QFX5200, QFX5210, EX4300-48MP, and EX4600 Switches with VXLANs | 196
- Changing the UDP Port on QFX5100, QFX5110, QFX5200, QFX5210, and EX4600 Switches | 197
- Controlling Transit Multicast Traffic on QFX5100, QFX5110, QFX5200, QFX5210, and EX4600 Switches | 198
- Using an MX Series Router, EX9200 Switch, or QFX10000 Switch as a VTEP | 198
- Manual VXLANs Require PIM | 199
- Load Balancing VXLAN Traffic | 199
- VLAN IDs for VXLANs | 200
- Enabling QFX5120 Switches to Tunnel Traffic on Core-Facing Layer 3 Tagged and IRB Interfaces | 200
- Using ping and traceroute with a VXLAN | 201
- Supported VXLAN Standards | 201

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

## VXLAN Benefits

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

- You can theoretically create as many as 16 million VXLANs in an administrative domain (as opposed to 4094 VLANs on a Juniper Networks device).
  - MX Series routers and EX9200 switches support as many as 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 Series switches support 4000 VXLANs and 2000 remote VTEPs.
  - QFX5100, QFX5110, QFX5200, QFX5210, and EX4600 switches support 4000 VXLANs, 4000 multicast groups, and 2000 remote VTEPs.
  - EX4300-48MP switches support 4000 VXLANs.
- 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?

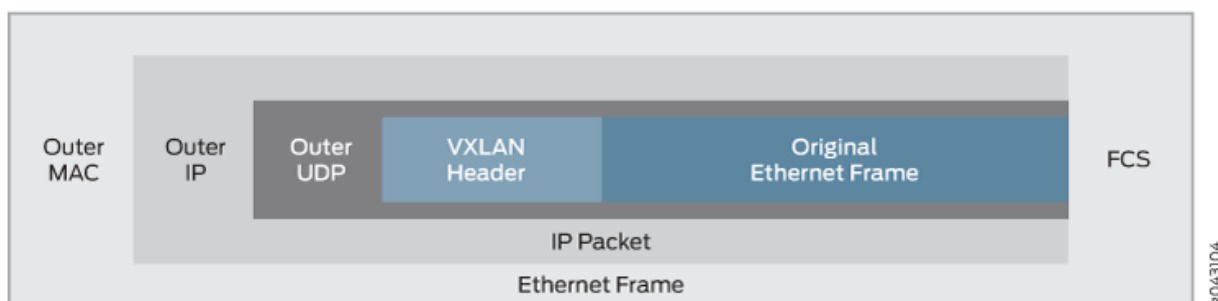
VXLAN is often described as an overlay technology because it allows you to stretch Layer 2 connections over an intervening Layer 3 network by encapsulating (tunneling) Ethernet frames in a VXLAN packet that includes IP addresses. Devices that support VXLANs are called *virtual tunnel endpoints (VTEPs)*—they 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 9 on page 195 shows the VXLAN packet format.

**Figure 9: 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, QFX5110, QFX5120, QFX5200, QFX5210, EX4300-48MP, and EX4600 Switches with VXLANs

You can configure the switches to perform all of the following roles:

- (All switches except EX4300-48MP) 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 199](#) for more information.)
- (All switches except EX4300-48MP) 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.
- (EX4300-48MP switches) Act as a Layer 2 gateway between virtualized and nonvirtualized networks in a campus network. For example, you can use the switch to connect a network that uses VXLANs to one that uses VLANs.
- (All switches except EX4300-48MP) 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.

- (QFX5110 and QFX5120 switches with EVPN-VXLAN) Act as a Layer 3 gateway to route traffic between different VXLANs in the same data center.
- (QFX5110 and QFX5120 switches with EVPN-VXLAN) 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 a QFX5110 or QFX5120 switch to be a Layer 3 VXLAN gateway in an EVPN-VXLAN environment, 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.

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, QFX5110, QFX5200, QFX5210, and EX4600 Switches

Starting with Junos OS Release 14.1X53-D25 on QFX5100 switches, Junos OS Release 15.1X53-D210 on QFX5110 and QFX5200 switches, Junos OS Release 18.1R1 on QFX5210 switches, and Junos OS Release 18.2R1 on EX4600 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, QFX5110, QFX5200, QFX5210, and EX4600 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. Starting with Junos OS Release 14.1X53-D30 for QFX5100 switches, Junos OS Release 15.1X53-D210 for QFX5110 and QFX5200 switches, Junos OS Release 18.1R1 for QFX5210 switches, and Junos OS Release 18.2R1 for EX4600 switches, 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. Junos OS Release 18.1R1 for QFX5210 switches

## Load Balancing VXLAN Traffic

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.

## VLAN IDs for VXLANs

When configuring a VLAN ID for a VXLAN on any Juniper Networks device that supports VXLANs except QFX10000 switches, we strongly recommend using a VLAN ID of 3 or higher. If you use a VLAN ID of 1 or 2, replicated broadcast, multicast, and unknown unicast (BUM) packets for these VXLANs might be untagged, which in turn might result in the packets being dropped by a device that receives the packets.

## Enabling QFX5120 Switches to Tunnel Traffic on Core-Facing Layer 3 Tagged and IRB Interfaces

**NOTE:** This section applies only to QFX5120 switches running Junos OS Releases 18.4R1, 18.4R2, 18.4R2-S1 through 18.4R2-S3, 19.1R1, 19.1R2, 19.2Rx, and 19.3Rx.

When a QFX5120 switch attempts to tunnel traffic on core-facing Layer 3 tagged interfaces or IRB interfaces, the switch drops the packets. To avoid this issue, you can configure a simple two-term filter-based firewall on the Layer 3 tagged or IRB interface.

**NOTE:** QFX5120 switches support a maximum of 256 two-term filter-based firewalls.

For example:

```
set interfaces et-0/0/3 unit 0 family inet filter input vxlan100
set firewall family inet filter vxlan100 term 1 from destination-address 192.168.0.1/24 then
accept
set firewall family inet filter vxlan100 term 2 then routing-instance route1
```

Term 1 matches and accepts traffic that is destined for the QFX5210 switch, which is identified by the source VTEP IP address (192.168.0.1/24) assigned to the switch's loopback interface. For term 1, note that when specifying an action, you can alternatively count traffic instead of accepting it.



Term 2 matches and forwards all other data traffic to a routing instance (route 1), which is configured interface et-0/0/3.

In this example, note that interface et-0/0/3 is referenced by routing instance route1. As a result, you must include the `set firewall family inet filter vxlan100 term 2 then routing-instance route1` command. Without this command, the firewall filter will not work properly.

### Using ping and traceroute with a VXLAN

On QFX5100 and QFX5110 switches, you can use the `ping` and `traceroute` commands to troubleshoot traffic flow through a VXLAN tunnel by including the `overlay` parameter and various options. You use these options to force the `ping` or `traceroute` packets to follow the same path as data packets through the VXLAN tunnel. In other words, you make the underlay packets (`ping` and `traceroute`) take the same route as the overlay packets (data traffic). See ["ping overlay" on page 268](#) and ["traceroute overlay" on page 275](#) for more information.

### Supported VXLAN Standards

RFCs and Internet drafts that define standards for VXLAN:

- RFC 7348, *Virtual eXtensible Local Area Network (VXLAN): A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks*
- Internet draft draft-ietf-nvo3-vxlan-gpe, *Generic Protocol Extension for VXLAN*

Release History Table

Release	Description
14.1X53-D30	Starting with Junos OS Release 14.1X53-D30 for QFX5100 switches, Junos OS Release 15.1X53-D210 for QFX5110 and QFX5200 switches, Junos OS Release 18.1R1 for QFX5210 switches, and Junos OS Release 18.2R1 for EX4600 switches, 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
14.1X53-D25	Starting with Junos OS Release 14.1X53-D25 on QFX5100 switches, Junos OS Release 15.1X53-D210 on QFX5110 and QFX5200 switches, Junos OS Release 18.1R1 on QFX5210 switches, and Junos OS Release 18.2R1 on EX4600 switches, you can configure the UDP port used as the destination port for VXLAN traffic.

### RELATED DOCUMENTATION

*Examples: Manually Configuring VXLANs on QFX Series and EX4600 Switches*

*Understanding EVPN with VXLAN Data Plane Encapsulation*

## PIM NSR and Unified ISSU Support for VXLAN Overview

Starting in Junos OS Release 16.2R1, Protocol Independent Multicast (PIM) nonstop active routing (NSR) support for Virtual Extensible LAN (VXLAN) is supported on MX Series routers.

The Layer 2 address learning daemon (l2ald) passes VXLAN parameters (VXLAN multicast group addresses and the source interface for a VXLAN tunnel [vtep-source-interface]) to the routing protocol process on the primary Routing Engine. The routing protocol process forms PIM joins with the multicast routes through the pseudo-VXLAN interface based on these configuration details.

Because the l2ald daemon does not run on the backup Routing Engine, the configured parameters are not available to the routing protocol process in the backup Routing Engine when NSR is enabled. The PIM NSR mirroring mechanism provides the VXLAN configuration details to the backup Routing Engine, which enables creation of the required states. The routing protocol process matches the multicast routes on the backup Routing Engine with PIM states, which maintains the multicast routes in the Forwarding state.

In response to Routing Engine switchover, the multicast routes remain in the Forwarding state on the new primary Routing Engine. This prevents traffic loss during Routing Engine switchover. When the l2ald process becomes active, it refreshes VXLAN configuration parameters to PIM.

**NOTE:** For this feature, NSR support is available for VXLAN in PIM sparse mode.

This feature does not introduce any new CLI commands. You can issue the following `show` commands on the backup Routing Engine to monitor the PIM joins and multicast routes on the backup Routing Engine:

- `show pim join extensive`
- `show multicast route extensive`

### Unified ISSU Support

Starting in Junos OS Release 17.2 R1, unified in-service software upgrade is supported for VXLAN using PIM on MX Series routers. ISSU enables you to upgrade your Junos OS software on your MX Series router with no disruption on the control plane and with minimal disruption of traffic. Unified ISSU is supported only on dual Routing Engine platforms. The graceful Routing Engine switchover (GRES) and nonstop active routing (NSR) features must both be enabled. Unified ISSU allows you to eliminate

network downtime, reduce operating costs, and deliver higher levels of services. See [Getting Started with Unified In-Service Software Upgrade](#).

**NOTE:** Unified ISSU is not supported on the QFX series switches.

To enable GRES, include the `graceful-switchover` statement at the `[edit chassis redundancy]` hierarchy level.

To enable NSR, include the `nonstop-routing` statement at the `[edit routing-options]` hierarchy level and the `commit synchronize` statement at the `[edit system]` hierarchy level.

**Release History Table**

Release	Description
17.2R1	Starting in Junos OS Release 17.2 R1, unified in-service software upgrade is supported for VXLAN using PIM on MX Series routers.
16.2R1	Starting in Junos OS Release 16.2R1, Protocol Independent Multicast (PIM) nonstop active routing (NSR) support for Virtual Extensible LAN (VXLAN) is supported on MX Series routers.

**RELATED DOCUMENTATION**

- [\*show pim join\*](#)
- [\*show multicast route\*](#)
- [Understanding Graceful Routing Engine Switchover](#)

## Example: Manually Configuring VXLANs on MX Series Routers

**IN THIS SECTION**

- [Requirements | 204](#)
- [Overview | 204](#)
- [Configuring VXLAN on MX Series Routers | 206](#)
- [Verification | 215](#)

Virtual Extensible Local Area Network (VXLAN) is a Layer 3 encapsulation protocol that enables MX Series routers to push Layer 2 or Layer 3 packets through a VXLAN tunnel to a virtualized data center or the Internet. Communication is established between two virtual tunnel endpoints (VTEPs). VTEPs encapsulate the virtual machine traffic into a VXLAN header and strip off the encapsulation.

This example shows how to configure VXLAN on MX Series routers using switch options in a default bridge domain.

## Requirements

This example uses the following hardware and software components:

- An MX Series router
- A VXLAN capable peer router
- Junos OS Release 14.1

## Overview

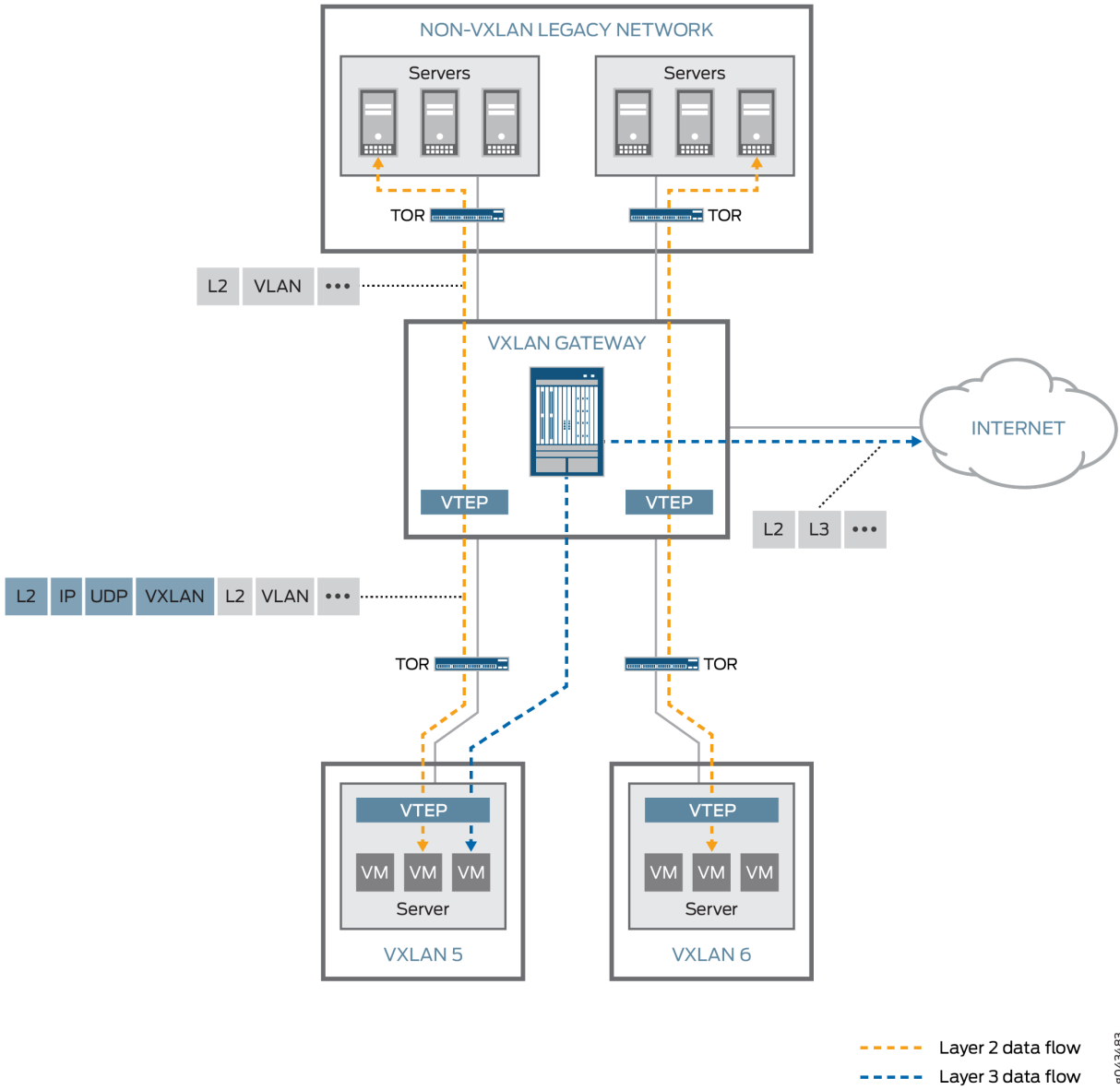
### IN THIS SECTION

- [Topology | 205](#)

In this example, VXLAN is configured to run on a default bridge domain. VTEP interfaces sources are configured to the loopback address, and VLAN groups are configured under bridge domains with VXLAN enabled. Interfaces are configured for VLAN tagging and encapsulation, and IRB is enabled. OSPF and PIM protocols are configured to facilitate unicast and multicast routing. The chassis is configured for GRES and enhanced IP services.

Topology

Figure 10: VXLAN Topology



8043483

## Configuring VXLAN on MX Series Routers

### IN THIS SECTION

- [CLI Quick Configuration | 206](#)
- [Configuring VXLAN | 207](#)
- [Results | 212](#)

### 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.

```
set switch-options vtep-source-interface lo0.0
set bridge-domains vlan-5 vxlan vni 100
set bridge-domains vlan-5 vxlan multicast-group 233.252.0.1
set bridge-domains vlan-5 vlan-id 100
set bridge-domains vlan-5 routing-interface irb.0
set bridge-domains vlan-5 interface xe-1/0/0.0
set bridge-domains vlan-6 vxlan vni 200
set bridge-domains vlan-6 vxlan multicast-group 233.252.0.1
set bridge-domains vlan-6 vlan-id 200
set bridge-domains vlan-6 routing-interface irb.1
set bridge-domains vlan-6 interface xe-2/0/0.0
set interfaces xe-1/0/0 vlan-tagging
set interfaces xe-1/0/0 encapsulation flexible-ethernet-services
set interfaces xe-1/0/0 unit 0 encapsulation vlan-bridge
set interfaces xe-1/0/0 unit 0 vlan-id 100
set interfaces xe-2/0/0 vlan-tagging
set interfaces xe-2/0/0 encapsulation flexible-ethernet-services
set interfaces xe-2/0/0 unit 0 encapsulation vlan-bridge
set interfaces xe-2/0/0 unit 0 vlan-id 200
set interface irb unit 0 family inet address 10.5.5.1/24
set interface irb unit 1 family inet address 10.6.6.1/24
set interfaces lo0 unit 0 family inet address 10.3.3.3/32
set protocols ospf area 0.0.0.0 interface ge-8/3/8.0
set protocols ospf area 0.0.0.0 interface lo0.0
```

```

set protocols ospf area 0.0.0.0 interface xe-0/1/3.0
set protocols ospf area 0.0.0.0 interface ge-8/3/2.0
set protocols pim rp static address 10.2.1.3
set protocols pim interface lo0.0 mode bidirectional-sparse
set protocols pim interface ge-8/3/8.0 mode bidirectional-sparse
set protocols pim interface xe-0/1/3.0 mode bidirectional-sparse
set protocols pim interface ge-8/3/2.0 mode bidirectional-sparse
set chassis redundancy graceful-switchover
set chassis aggregated-devices ethernet device-count 10
set chassis fpc 1 pic 0 tunnel-services bandwidth 10g
set chassis network-services enhanced-ip

```

## Configuring VXLAN

### Step-by-Step Procedure

The following example show how to set up a basic VXLAN configuration with default bridge domains and switch options. To configure VXLAN on an MX Series router, follow these steps:

1. Configure VTEP interface sources under switch-options for the default-switch.

```

[edit]
user@router# set switch-options vtep-source-interface lo0.0

```

2. Set up a VLAN group named vlan-5 and set its VXLAN Network Identifier (VNI) to 100.

```

[edit]
user@router# set bridge-domains vlan-5 vxlan vni 100

```

3. Configure the vlan-5 multicast group address for VXLAN.

```

[edit]
user@router# set bridge-domains vlan-5 vxlan multicast-group 233.252.0.1/32

```

4. Set the VLAN ID to 100 for vlan-5.

```

[edit]
user@router# set bridge-domains vlan-5 vlan-id 100

```

5. Configure integrated bridging and routing (IRB) for vlan-5.

```
[edit]
user@router# set bridge-domains vlan-5 routing-interface irb.0
```

6. Assign the xe-1/0/0.0 interface to vlan-5.

```
[edit]
user@router# set bridge-domains vlan-5 interface xe-1/0/0.0
```

7. Set up a VLAN group named vlan-6 and set its VXLAN Network Identifier (VNI) to 200.

```
[edit]
user@router# set bridge-domains vlan-6 vxlan vni 200
```

8. Configure the vlan-6 multicast group address for VXLAN.

```
[edit]
user@router# set bridge-domains vlan-6 vxlan multicast-group 233.252.0.1
```

9. Set the VLAN ID to 100 for vlan-6.

```
[edit]
user@router# set bridge-domains vlan-6 vlan-id 200
```

10. Configure IRB for vlan-6.

```
[edit]
user@router# set bridge-domains vlan-6 routing-interface irb.1
```

11. Assign the xe-2/0/0.0 interface to vlan-6.

```
[edit]
user@router# set bridge-domains vlan-6 interface xe-2/0/0.0
```



12. Set up VLAN tagging for xe-1/0/0.

```
[edit]
user@router# set interfaces xe-1/0/0 vlan-tagging
```

13. Configure flexible Ethernet service encapsulation on xe-1/0/0.

```
[edit]
user@router# set interfaces xe-1/0/0 encapsulation flexible-ethernet-services
```

14. Set up VLAN bridging encapsulation for xe-1/0/0 unit 0.

```
[edit]
user@router# set interfaces xe-1/0/0 unit 0 encapsulation vlan-bridge
```

15. Set the xe-1/0/0 unit 0 VLAN ID to 100.

```
[edit]
user@router# set interfaces xe-1/0/0 unit 0 vlan-id 100
```

16. Configure VLAN tagging for xe-2/0/0

```
[edit]
user@router# set interfaces xe-2/0/0 vlan-tagging
```

17. Set up flexible Ethernet service encapsulation on xe-2/0/0.

```
[edit]
user@router# set interfaces xe-2/0/0 encapsulation flexible-ethernet-services
```

18. Configure VLAN bridging encapsulation for xe-2/0/0 unit 0.

```
[edit]
user@router# set interfaces xe-2/0/0 unit 0 encapsulation vlan-bridge
```

19. Set the xe-2/0/0 unit 0 VLAN ID to 200.

```
[edit]
user@router# set interfaces xe-2/0/0 unit 0 vlan-id 200
```

20. Configure the IRB unit 0 family inet address.

```
[edit]
user@router# set interface irb unit 0 family inet address 10.5.5.1/24
```

21. Configure the IRB unit 1 family inet address.

```
[edit]
user@router# set interface irb unit 1 family inet address 10.6.6.1/24
```

22. Set the family inet address for the loopback unit 0.

```
[edit]
user@router# set interfaces lo0 unit 0 family inet address 10.3.3.3/32
```

23. Set up OSPF for the ge-8/3/8.0 interface.

```
[edit]
user@router# set protocols ospf area 0.0.0.0 interface ge-8/3/8.0
```

24. Configure OSPF for the loopback interface.

```
[edit]
user@router# set protocols ospf area 0.0.0.0 interface lo0.0
```

25. Set up OSPF for the xe-0/1/3.0 interface.

```
[edit]
user@router# set protocols ospf area 0.0.0.0 interface xe-0/1/3.0
```

26. Configure OSPF for the ge-8/3/2.0 interface.

```
[edit]
user@router# set protocols ospf area 0.0.0.0 interface ge-8/3/2.0
```

27. Set up the static address for the protocol independent multicast (PIM) rendezvous point (RP).

```
[edit]
user@router# set protocols pim rp static address 10.2.1.3
```

28. Configure the loopback interface to bidirectional sparse mode for the PIM protocol.

```
[edit]
user@router# set protocols pim interface lo0.0 mode bidirectional-sparse
```

29. Set the ge-8/3/8.0 interface to bidirectional sparse mode for the PIM protocol.

```
[edit]
user@router# set protocols pim interface ge-8/3/8.0 mode bidirectional-sparse
```

30. Configure the xe-0/1/3.0 interface to bidirectional sparse mode for the PIM protocol.

```
[edit]
user@router# set protocols pim interface xe-0/1/3.0 mode bidirectional-sparse
```

31. Set the ge-8/3/2.0 interface to bidirectional sparse mode for the PIM protocol.

```
[edit]
user@router# set protocols pim interface ge-8/3/2.0 mode bidirectional-sparse
```

32. Configure redundant graceful switchover on the chassis.

```
[edit]
user@router# set chassis redundancy graceful-switchover
```

33. Set the aggregated ethernet device count to 10.

```
[edit]
user@router# set chassis aggregated-devices ethernet device-count 10
```

34. Configure the tunnel services bandwidth for FPC 1/PIC 0.

```
[edit]
user@router# set chassis fpc 1 pic 0 tunnel-services bandwidth 10g
```

35. Enable enhanced IP for network services on the chassis.

```
[edit]
user@router# set chassis network-services enhanced-ip
```

## Results

From configuration mode, confirm your configuration by entering the following commands. If the output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
user@router# show switch-options
```

```
switch-options {
  vtep-source-interface lo0.0;
}
```

```
user@router# show bridge-domains
```

```
bridge-domains {
  vlan-5 {
    vxlan {
      vni          100;
      multicast-group 233.252.0.1;
    }
  }
}
```

```

        vlan-id          100;
        routing-interface irb.0;
        interface        xe-1/0/0.0;
    }
    vlan-6 {
        vxlan {
            vni          200;
            multicast-group 233.252.0.1;
        }
        vlan-id          200;
        routing-interface irb.1;
        interface        xe-2/0/0.0;
    }
}

```

**user@router# show interfaces**

```

interfaces {
    xe-1/0/0 {
        vlan-tagging;
        encapsulation flexible-ethernet-services;
        unit 0 {
            encapsulation vlan-bridge;
            vlan-id 100;
        }
    }
    xe-2/0/0 {
        vlan-tagging;
        encapsulation flexible-ethernet-services;
        unit 0 {
            encapsulation vlan-bridge;
            vlan-id 200;
        }
    }
    irb {
        unit 0 {
            family inet {
                address 10.5.5.1/24;
            }
        }
    }
}

```

```

        unit 1 {
            family inet {
                address 10.6.6.1/24;
            }
        }
    }
    lo0 {
        unit 0 {
            family inet {
                address 10.3.3.3/32;
            }
        }
    }
}

```

```
user@router# show protocols ospf
```

```

area 0.0.0.0 {
    interface ge-8/3/8.0;
    interface lo0.0;
    interface xe-0/1/3.0;
    interface ge-8/3/2.0;
}

```

```
user@router# show protocols pim
```

```

rp {
    static {
        address 10.2.1.3;
    }
}

```

```
}  
}
```

```
user@router# show chassis
```

```
redundancy {  
    graceful-switchover;  
}  
aggregated-devices {  
    ethernet {  
        device-count 10;  
    }  
}  
fpc 1 {  
    pic 0 {  
        tunnel-services {  
            bandwidth 10g;  
        }  
    }  
}  
network-services enhanced-ip;
```

## Verification

### IN THIS SECTION

- [Verifying Reachability | 215](#)
- [Verifying VXLAN | 216](#)

Confirm that the configuration is working properly.

### Verifying Reachability

#### Purpose

Verify that the network is up and running with the proper interfaces and routes installed.

## Action

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

Interface	Admin	Link	Proto	Local	Remote
irb	up	up			
irb.0	up	up	inet	10.5.5.1/24	
				multiservice	
irb.1	up	up	inet	10.6.6.1/24	
				multiservice	

```
user@router> ping 10.5.5.1/24
PING 10.5.5.1 (10.5.5.1): 56 data bytes
64 bytes from 10.5.5.1: icmp_seq=0 ttl=64 time=0.965 ms
64 bytes from 10.5.5.1: icmp_seq=1 ttl=64 time=0.960 ms
64 bytes from 10.5.5.1: icmp_seq=2 ttl=64 time=0.940 ms
^C
--- 10.1.1.1 ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max/stddev = 0.940/0.955/0.965/0.011 ms
```

## Meaning

Use the `show interfaces terse irb` command to verify that the IRB interface has been properly configured. The `irb.0` and `irb.1` interfaces should display the proper multiservice inet addresses.

Use the `ping` command to confirm that the network is connected to the IRB multiservice address.

## Verifying VXLAN

### Purpose

Verify that VXLAN is working and the proper protocols are enabled.

## Action

```
user@router> show interfaces vtep
Physical interface: vtep, Enabled, Physical link is Up
```



```

Interface index: 133, SNMP ifIndex: 575
Type: Software-Pseudo, Link-level type: VxLAN-Tunnel-Endpoint, MTU: 1600, Speed: Unlimited
Device flags   : Present Running
Interface flags: SNMP-Traps
Link type      : Full-Duplex
Link flags     : None
Last flapped   : Never
  Input packets : 0
  Output packets: 0

Logical interface vtep.32768 (Index 334) (SNMP ifIndex 607)
Flags: Up SNMP-Traps Encapsulation: ENET2
VXLAN Endpoint Type: Source, VXLAN Endpoint Address: 10.255.187.32, L2 Routing Instance:
default-switch, L3 Routing Instance: default
  Input packets : 0
  Output packets: 0

```

```

user@router> show l2-learning vxlan-tunnel-end-point remote mac-table
MAC flags (S -static MAC, D -dynamic MAC, L -locally learned, C -Control MAC
          SE -Statistics enabled, NM -Non configured MAC, R -Remote PE MAC)

Logical system   : <default>
Routing instance : default-switch
  Bridging domain : vlan-5+100, VLAN : 100, VNID : 100
  Bridging domain : vlan-6+200, VLAN : 200, VNID : 200

```

```

user@router> show l2-learning vxlan-tunnel-end-point source

```

Logical System Name	Id	SVTEP-IP	IFL	L3-Idx	
<default>	0	10.255.187.32	lo0.0	0	
L2-RTT		Bridge Domain		VNID	MC-Group-IP
default-switch		vlan-5+100		100	233.252.0.1
default-switch		vlan-6+200		200	233.252.0.1

## Meaning

Use the show interface vtep command to displays information about VXLAN endpoint configuration. Make sure the routing instance is assigned to the default-switch..

Use the `show l2-learning vxlan-tunnel-end-point remote mac-table` command to confirm that the bridging domain VLAN groups were configured correctly.

Use the `show l2-learning vxlan-tunnel-end-point source` command to confirm the multicast IP addresses for bridging domain VLAN groups.

## RELATED DOCUMENTATION

---

*Understanding VXLANs*

---

*show bridge mac-table*

---

*show vpls mac-table*

## CHAPTER 7

# VXLAN Configuration Statements

**IN THIS CHAPTER**

- [encapsulate-inner-vlan | 219](#)
- [multicast-group | 221](#)
- [ovsdb-managed | 222](#)
- [unreachable-vtep-aging-timer | 223](#)
- [vni | 225](#)
- [vxlan | 226](#)

## encapsulate-inner-vlan

**IN THIS SECTION**

- [Syntax | 219](#)
- [Hierarchy Level | 220](#)
- [Description | 220](#)
- [Default | 220](#)
- [Required Privilege Level | 220](#)
- [Release Information | 220](#)

### Syntax

```
encapsulate-inner-vlan
```

## Hierarchy Level

```
[edit routing-instances routing-instance-name vlan vlan-name vxlan],  
[edit vlan vlan-name vxlan]
```

## Description

Configure the switch to preserve the original VLAN tag (in the inner Ethernet packet) when performing Virtual Extensible LAN (VXLAN) encapsulation.

**NOTE:** ARP suppression is automatically disabled when you encapsulate VLAN traffic for EVPN-VXLAN by configuring `encapsulate-inner-vlan` and its equivalent converse statement `decapsulate-accept-inner-vlan`. As a result, you may observe ARP packets being sent by the device.

## Default

The original tag is dropped when the packet is encapsulated.

## Required Privilege Level

routing—To view this statement in the configuration.

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

## Release Information

Statement introduced in Junos OS Release 14.1R2.

## RELATED DOCUMENTATION

*Understanding VXLANs*

*Manually Configuring VXLANs on QFX Series and EX4600 Switches*

*Examples: Manually Configuring VXLANs on QFX Series and EX4600 Switches*

*decapsulate-accept-inner-vlan*

## multicast-group

### IN THIS SECTION

- [Syntax | 221](#)
- [Hierarchy Level | 221](#)
- [Description | 221](#)
- [Required Privilege Level | 221](#)
- [Release Information | 221](#)

### Syntax

```
multicast-group address
```

### Hierarchy Level

```
[edit vlanvlan-name vxlan]
```

### Description

Assign a multicast group address to a Virtual Extensible LAN (VXLAN). All members of a VXLAN must use the same multicast group address.

### Required Privilege Level

routing—To view this statement in the configuration.

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

### Release Information

Statement introduced in Junos OS Release 14.1R2.

## RELATED DOCUMENTATION

*Understanding VXLANs*

*Manually Configuring VXLANs on QFX Series and EX4600 Switches*

*Examples: Manually Configuring VXLANs on QFX Series and EX4600 Switches*

## ovsdb-managed

### IN THIS SECTION

- [Syntax | 222](#)
- [Hierarchy Level | 222](#)
- [Description | 223](#)
- [Required Privilege Level | 223](#)
- [Release Information | 223](#)

### Syntax

```
ovsdb-managed;
```

### Hierarchy Level

```
[edit bridge-domains bridge-domain-name vxlan],  
[edit routing-instances routing-instance-name bridge-domains bridge-domain-name vxlan],  
[edit routing-instances routing-instance-name switch-options],  
[edit routing-instances routing-instance-name vlans vlan-name vxlan],  
[edit routing-instances routing-instance-name vxlan],  
[edit switch-options],  
[edit vlans vlan-name vxlan]
```

## Description

Disable a Juniper Networks device from learning about other Juniper Networks devices that function as hardware *virtual tunnel endpoints (VTEPs)* in a specified Virtual Extensible LAN (VXLAN) and the media access control (MAC) addresses learned by the hardware VTEPs. Instead, the Juniper Networks device uses the *Open vSwitch Database (OVSDB)* management protocol to learn about the hardware VTEPs in the VXLAN and the MAC addresses learned by the hardware VTEPs.

The specified VXLAN must have a VXLAN network identifier (VNI) configured, using the `vni` statement in the `[edit bridge-domains bridge-domain-name vxlan]`, `[edit routing-instance routing-instance-name vxlan]`, or `[edit vlans vlan-name vxlan]` hierarchy.

Also, for OVSDB-managed VXLANs, the multicast scheme described in "[Understanding How Layer 2 BUM and Layer 3 Routed Multicast Traffic Are Handled with OVSDB](#)" on page 8 is used. Therefore, specifying the `multicast-group` statement in the `[edit bridge-domains bridge-domain-name vxlan]`, `[edit routing-instances routing-instance-name vxlan]`, or `[edit vlans vlan-name vxlan]` hierarchy has no effect.

## Required Privilege Level

admin—To view this statement in the configuration.

admin-control—To add this statement to the configuration.

## Release Information

Statement introduced in Junos OS Release 14.1R2.

## RELATED DOCUMENTATION

*Understanding Dynamically Configured VXLANs in an OVSDB Environment*

[Configuring OVSDB-Managed VXLANs](#) | 26

## unreachable-vtep-aging-timer

### IN THIS SECTION

● [Syntax](#) | 224

- [Hierarchy Level | 224](#)
- [Description | 224](#)
- [Required Privilege Level | 224](#)
- [Release Information | 224](#)

## Syntax

```
unreachable-vtep-aging-timer [300-1800]
```

## Hierarchy Level

```
[edit vlanvlan-name vxlan]
```

## Description

Configure the system to age out the address for the remote virtual tunnel endpoint (VTEP) if all the MAC addresses learned from that VTEP age out. The address for the remote VTEP expires the configured number of seconds after the last learned media access control (MAC) address expires.

## Required Privilege Level

routing—To view this statement in the configuration.

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

## Release Information

Statement introduced in Junos OS Release 14.1R2.

## RELATED DOCUMENTATION

*Understanding VXLANs*

*Manually Configuring VXLANs on QFX Series and EX4600 Switches*

*Examples: Manually Configuring VXLANs on QFX Series and EX4600 Switches*



## vni

### IN THIS SECTION

- [Syntax | 225](#)
- [Hierarchy Level | 225](#)
- [Description | 225](#)
- [Options | 225](#)
- [Required Privilege Level | 226](#)
- [Release Information | 226](#)

### Syntax

```
vni [0-16777214]
```

### Hierarchy Level

```
[edit routing-instances routing-instance-name vlans vlan-name vxlan]  
[edit vlans vlan-name vxlan]
```

### Description

Assign a numeric value to identify a Virtual Extensible LAN (VXLAN). All members of a VXLAN must use the same VNI.

### Options

**vni**            Value to specify in the `vni` attribute.

- **Range:** 0 through 16,777,215

**NOTE:** Starting in Junos OS Release 17.3R3-3, 18.1R3-S3, and 19.1R1, Junos OS supports a VNI value of 0.

## Required Privilege Level

routing—To view this statement in the configuration.

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

## Release Information

Statement introduced in Junos OS Release 14.1R2.

## RELATED DOCUMENTATION

*Understanding VXLANs*

*Manually Configuring VXLANs on QFX Series and EX4600 Switches*

*Examples: Manually Configuring VXLANs on QFX Series and EX4600 Switches*

## vxlan

### IN THIS SECTION

- [Syntax | 227](#)
- [Hierarchy Level | 227](#)
- [Description | 227](#)
- [Options | 227](#)
- [Required Privilege Level | 227](#)
- [Release Information | 228](#)

## Syntax

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

```

## Hierarchy Level

```
[edit vlan name]
[edit bridge-domains bridge-domain-name]
```

## Description

Configure support for Virtual Extensible LANs (VXLANs) on a Juniper Networks device.

## Options

**multicast-group *multicast-group*** Multicast group (IPv4 or IPv6 addresses) registered for VXLAN segment.

**mtu *mtu*** Specify the maximum transmission unit (MTU) size for a VXLAN packet . The range is from 100 to 65535 bytes.

The remaining statements are explained separately. See [CLI Explorer](#).

## Required Privilege Level

routing—To view this statement in the configuration.

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

## Release Information

Statement introduced in Junos OS Release 14.1X53-D10.

`ingress-node-replication` option added for EVPN VXLAN on QFX5100 switches in Junos OS Release 14.1X53-D30.

`multicast-group` *multicast-group* option added for MX series routers with MPC and MIC interfaces in Junos OS Release 17.2.

`mtu` option added for cRPD in Junos OS Release 21.2R1.

`riot-loopback` option added for EVPN-VXLAN on QFX5210 switches in Junos OS Release 21.3R1.

## RELATED DOCUMENTATION

---

*Understanding VXLANs*

---

*Using a RIOT Loopback Port to Route Traffic in an EVPN Network*

---

*Manually Configuring VXLANs on QFX Series and EX4600 Switches*

---

*Examples: Manually Configuring VXLANs on QFX Series and EX4600 Switches*

## CHAPTER 8

# VXLAN Monitoring Commands

**IN THIS CHAPTER**

- [Monitoring a Remote VTEP Interface | 229](#)
- [show bridge mac-table | 231](#)
- [show vpls mac-table | 239](#)
- [Verifying VXLAN Reachability | 246](#)
- [Verifying That a Local VXLAN VTEP Is Configured Correctly | 248](#)
- [Verifying MAC Learning from a Remote VTEP | 249](#)
- [Understanding Overlay ping and traceroute Packet Support | 250](#)
- [Example: Troubleshooting a VXLAN Overlay Network By Using Overlay Ping and Traceroute for MX Series Routers | 254](#)
- [ping overlay | 268](#)
- [traceroute overlay | 275](#)

## Monitoring a Remote VTEP Interface

**IN THIS SECTION**

- [Purpose | 229](#)
- [Action | 230](#)
- [Meaning | 230](#)

### Purpose

Monitor traffic details for a remote VTEP interface.

# Action

```

user@switch> show interface logical-name detail

M   Flags: Up SNMP-Traps Encapsulation: ENET2
      VXLAN Endpoint Type: Remote, VXLAN Endpoint Address: 10.1.1.2, L2 Routing Instance: default-
switch, L3 Routing Instance: default
      Traffic statistics:
        Input  bytes :          228851738624
        Output bytes :              0
        Input  packets:          714162415
        Output packets:              0
      Local statistics:
        Input  bytes :              0
        Output bytes :              0
        Input  packets:              0
        Output packets:              0
      Transit statistics:
        Input  bytes :          228851738624          0 bps
        Output bytes :              0          0 bps
        Input  packets:          714162415          0 pps
        Output packets:              0          0 pps
      Protocol eth-switch, MTU: 1600, Generation: 277, Route table: 5

```

# Meaning

The output shows traffic details for the remote VTEP interface. To get this information, you must supply the logical name of the remote VTEP interface (vtep.12345 in the above output), which you can learn by using the `show ethernet-switching table` command.

## RELATED DOCUMENTATION

<i>Understanding VXLANs</i>
<i>Manually Configuring VXLANs on QFX Series and EX4600 Switches</i>
<i>Examples: Manually Configuring VXLANs on QFX Series and EX4600 Switches</i>

## show bridge mac-table

### IN THIS SECTION

- [Syntax | 231](#)
- [Description | 231](#)
- [Options | 231](#)
- [Additional Information | 232](#)
- [Required Privilege Level | 232](#)
- [Output Fields | 232](#)
- [Sample Output | 234](#)
- [Release Information | 238](#)

### Syntax

```
show bridge mac-table
<age>
<brief | count | detail | extensive>
<bridge-domain (all | bridge-domain-name)>
<global-count>
<instance instance-name>
<interface interface-name>
<mac-address>
<instance instance-name>
<vlan-id (all-vlan | vlan-id)>
```

### Description

(MX Series routers only) Display Layer 2 MAC address information.

### Options

**none**                      Display all learned Layer 2 MAC address information.

<b>age</b>	(Optional) Display age of a single mac-address.
<b>brief   count   detail   extensive</b>	(Optional) Display the specified level of output.
<b>bridge-domain (all   <i>bridge-domain-name</i>)</b>	(Optional) Display learned Layer 2 MAC addresses for all bridging domains or for the specified bridging domain.
<b>global-count</b>	(Optional) Display the total number of learned Layer 2 MAC addresses on the system.
<b>instance <i>instance-name</i></b>	(Optional) Display learned Layer 2 MAC addresses for the specified routing instance.
<b>interface <i>interface-name</i></b>	(Optional) Display learned Layer 2 MAC addresses for the specified interface.
<b><i>mac-address</i></b>	(Optional) Display the specified learned Layer 2 MAC address information.
<b>vlan-id (all-vlan   <i>vlan-id</i>)</b>	(Optional) Display learned Layer 2 MAC addresses for all VLANs or for the specified VLAN.

## Additional Information

When Layer 2 protocol tunneling is enabled, the tunneling MAC address 01:00:0c:cd:cd:d0 is installed in the MAC table. When the Cisco Discovery Protocol (CDP), Spanning Tree Protocol (STP), or VLAN Trunk Protocol (VTP) is configured for Layer 2 protocol tunneling on an interface, the corresponding protocol MAC address is installed in the MAC table.

## Required Privilege Level

view

## Output Fields

[Table 18 on page 233](#) describes the output fields for the `show bridge mac-table` command. Output fields are listed in the approximate order in which they appear.



**Table 18: show bridge mac-table Output Fields**

Field Name	Field Description
Age	Age of a single mac-address.
Routing instance	Name of the routing instance.
Bridging domain	Name of the bridging domain.
MAC address	MAC address or addresses learned on a logical interface.
MAC flags	<p>Status of MAC address learning properties for each interface:</p> <ul style="list-style-type: none"> <li>• S—Static MAC address is configured.</li> <li>• D—Dynamic MAC address is configured.</li> <li>• L—Locally learned MAC address is configured.</li> <li>• C—Control MAC address is configured.</li> <li>• SE—MAC accounting is enabled.</li> <li>• NM—Non-configured MAC.</li> <li>• R—Remote PE MAC address is configured.</li> <li>• P—MAC Pinned interface is configured</li> </ul>
Logical interface	Name of the logical interface.
MAC count	Number of MAC addresses learned on the specific routing instance or interface.
Learning interface	Name of the logical interface on which the MAC address was learned.
Learning VLAN	VLAN ID of the routing instance or bridge domain in which the MAC address was learned.
VXLAN ID/VXLAN	VXLAN Network Identifier (VNI).

**Table 18: show bridge mac-table Output Fields (Continued)**

Field Name	Field Description
Layer 2 flags	Debugging flags signifying that the MAC address is present in various lists.
Epoch	Spanning Tree Protocol epoch number identifying when the MAC address was learned. Used for debugging.
Sequence number	Sequence number assigned to this MAC address. Used for debugging.
Learning mask	Mask of the Packet Forwarding Engines where this MAC address was learned. Used for debugging.
IPC generation	Creation time of the logical interface when this MAC address was learned. Used for debugging.

## Sample Output

### show bridge mac-table

```

user@host> show bridge mac-table
MAC flags (S -static MAC, D -dynamic MAC, L -locally learned, C -Control MAC
          SE -Statistics enabled, NM -Non configured MAC, R -Remote PE MAC)

Routing instance : default-switch
Bridging domain : test1, VLAN : 1
  MAC          MAC      Logical      NH      RTR
  address      flags    interface  Index   ID
01:00:0c:cc:cc:cc S,NM    NULL
01:00:0c:cc:cc:cd S,NM    NULL
01:00:0c:cd:cd:d0 S,NM    NULL
64:87:88:6a:17:d0 D        ae0.1
64:87:88:6a:17:f0 D        ae0.1

```

## show bridge mac-table (with Layer 2 Services over GRE Interfaces)

```
user@host> show bridge mac-table
MAC flags (S -static MAC, D -dynamic MAC, L -locally learned
          SE -Statistics enabled, NM -Non configured MAC, R -Remote PE MAC)

Routing instance : default-switch
Bridging domain : vlan-1, VLAN : 1
  MAC          MAC      Logical
  address      flags    interface
  00:01:01:00:01:f7 D,SE   gr-1/2/10.0
  00:03:00:32:01:f7 D,SE   gr-1/2/10.0
  00:00:21:11:11:10 DL      ge-1/0/0.0
  00:00:21:11:11:11 DL      ge-1/1/0.0

Routing instance : default-switch
Bridging domain : vlan-2, VLAN : 2
  MAC          MAC      Logical
  address      flags    interface
  00:02:01:33:01:f7 D,SE   gr-1/2/10.1
  00:00:21:11:21:10 DL      ge-1/0/0.1
  00:00:21:11:21:11 DL      ge-1/1/0.1
```

## show bridge mac-table (with VXLAN enabled)

```
user@host> show bridge mac-table
MAC flags (S -static MAC, D -dynamic MAC, L -locally learned
          SE -Statistics enabled, NM -Non configured MAC, R -Remote PE MAC)

Routing instance : default-switch
Bridging domain : vlan-1, VLAN : 1
VXLAN: Id : 100, Multicast group: 233.252.0.1
  MAC          MAC      Logical
  address      flags    interface
  00:01:01:00:01:f7 D,SE   vtep.1052010
  00:03:00:32:01:f7 D,SE   vtep.1052011
  00:00:21:11:11:10 DL      ge-1/0/0.0
  00:00:21:11:11:11 DL      ge-1/1/0.0
```

```

Routing instance : default-switch
Bridging domain : vlan-2, VLAN : 2, VXLAN : 200
VXLAN: Id : 200, Multicast group: 233.252.0.2
  MAC          MAC      Logical
  address      flags    interface
00:02:01:33:01:f7 D,SE    vtep.1052010
00:04:00:14:01:f7 D,SE    vtep.1052011
00:00:21:11:21:10 DL      ge-1/0/0.1
00:00:21:11:21:11 DL      ge-1/1/0.1

```

### show bridge mac-table age (for GE interface)

```

user@host> show vpls mac-table age 00:02:03:aa:bb:1a instance vpls_instance_1
MAC Entry Age information
Current Age: 4 seconds

```

### show bridge mac-table age (for AE interface)

```

user@host> show vpls mac-table age 00:02:03:aa:bb:1a instance vpls_instance_1
MAC Entry Age information
Current Age on FPC1: 102 seconds
Current Age on FPC2: 94 seconds

```

### show bridge mac-table count

```

user@host> show bridge mac-table count
2 MAC address learned in routing instance vs1 bridge domain vlan100

MAC address count per interface within routing instance:
  Logical interface      MAC count
  ge-11/0/3.0            1
  ge-11/1/4.100          0
  ge-11/1/1.100          0
  ge-11/1/0.100          0
  xe-10/2/0.100          1
  xe-10/0/0.100          0

```

MAC address count per learn VLAN within routing instance:

Learn VLAN ID	MAC count
0	2

0 MAC address learned in routing instance vs1 bridge domain vlan200

MAC address count per interface within routing instance:

Logical interface	MAC count
ge-11/1/0.200	0
ge-11/1/1.200	0
ge-11/1/4.200	0
xe-10/0/0.200	0
xe-10/2/0.200	0

MAC address count per learn VLAN within routing instance:

Learn VLAN ID	MAC count
0	0

## show bridge mac-table detail

```
user@host> show bridge mac-table detail
```

MAC address: 00:00:00:19:1c:db

Routing instance: vs1

Bridging domain: vlan100

Learning interface: ge-11/0/3.0      Learning VLAN: 0

Layer 2 flags: in\_ifd, in\_ifl, in\_vlan, kernel

Epoch: 4      Sequence number: 0

Learning mask: 0x800      IPC generation: 0

MAC address: 00:00:00:59:3a:2f

Routing instance: vs1

Bridging domain: vlan100

Learning interface: xe-10/2/0.100      Learning VLAN: 0

Layer 2 flags: in\_ifd, in\_ifl, in\_vlan, kernel

Epoch: 7      Sequence number: 0

Learning mask: 0x400      IPC generation: 0

## show bridge mac-table instance pbb-evpn

```
user@host> show bridge mac-table instance pbb-evpn
Routing instance : pbb-evpn
Bridging domain : isid-bd10000, ISID : 10000
  MAC          MAC      Logical      NH      RTR
  address      flags      interface  Index  ID
00:19:e2:b0:76:eb  D        cbp.1000
aa:bb:cc:dd:ee:f2  DC                    1048576 1048576
aa:bb:cc:dd:ee:f3  DC                    1048575 1048575
```

## show bridge mac-table

```
user@host>run show bridge mac-table
MAC flags (S -static MAC, D -dynamic MAC, L -locally learned, C -Control MAC
O -OVSDB MAC, SE -Statistics enabled, NM -Non configured MAC, R -Remote PE MAC, P -Pinned MAC)

Routing instance : VS-541
Bridging domain : 541, VLAN : 541
MAC MAC Logical NH RTR
address flags interface Index ID
00:00:01:00:00:01 DPRC xe-0/0/3.0
00:00:02:00:00:01 DP  xe-0/0/3.0
```

## Release Information

Command introduced in Junos OS Release 8.4.

Support for PBB-EVPN instance added in Junos OS Release 16.1

MAC Flag P to indicate a MAC Pinned interface introduced in Junos OS 16.2

## show vpls mac-table

### IN THIS SECTION

- [Syntax | 239](#)
- [Description | 239](#)
- [Options | 239](#)
- [Required Privilege Level | 240](#)
- [Output Fields | 240](#)
- [Sample Output | 242](#)
- [Release Information | 246](#)

### Syntax

```
show vpls mac-table
<age>
<brief | detail | extensive | summary>
<bridge-domain bridge-domain-name>
<instance instance-name>
<interface interface-name>
<logical-system (all | logical-system-name)>
<mac-address>
<vlan-id vlan-id-number>
```

### Description

Display learned virtual private LAN service (VPLS) media access control (MAC) address information.

### Options

- |             |   |
|-------------|---|
| <b>none</b> | Display all learned VPLS MAC address information. |
| <b>age</b>  | (Optional) Display age of a single mac-address.   |

<b>brief   detail   extensive   summary</b>	(Optional) Display the specified level of output.
<b>bridge-domain</b> <i>bridge-domain-name</i>	(Optional) Display learned VPLS MAC addresses for the specified bridge domain.
<b>instance</b> <i>instance-name</i>	(Optional) Display learned VPLS MAC addresses for the specified instance.
<b>interface</b> <i>interface-name</i>	(Optional) Display learned VPLS MAC addresses for the specified instance.
<b>logical-system</b> (all   <i>logical-system-name</i> )	(Optional) Display learned VPLS MAC addresses for all logical systems or for the specified logical system.
<b>mac-address</b>	(Optional) Display the specified learned VPLS MAC address information..
<b>vlan-id</b> <i>vlan-id-number</i>	(Optional) Display learned VPLS MAC addresses for the specified VLAN.

## Required Privilege Level

view

## Output Fields

[Table 19 on page 240](#) describes the output fields for the `show vpls mac-table` command. Output fields are listed in the approximate order in which they appear.

**Table 19: show vpls mac-table Output fields**

Field Name	Field Description
Age	Age of a single mac-address.
Routing instance	Name of the routing instance.
Bridging domain	Name of the bridging domain.
MAC address	MAC address or addresses learned on a logical interface.



**Table 19: show vpls mac-table Output fields (Continued)**

Field Name	Field Description
MAC flags	<p>Status of MAC address learning properties for each interface:</p> <ul style="list-style-type: none"> <li>• S—Static MAC address configured.</li> <li>• D—Dynamic MAC address learned.</li> <li>• SE—MAC accounting is enabled.</li> <li>• NM—Nonconfigured MAC.</li> </ul>
Logical interface	Name of the logical interface.
MAC count	Number of MAC addresses learned on a specific routing instance or interface.
Learning interface	Logical interface or logical Label Switched Interface (LSI) the address is learned on.
Base learning interface	Base learning interface of the MAC address. This field is introduced in Junos OS Release 14.2.
Learn VLAN ID/ VLAN	VLAN ID of the routing instance or bridge domain in which the MAC address was learned.
VXLAN ID/VXLAN	VXLAN Network Identifier (VNI)
Layer 2 flags	Debugging flags signifying that the MAC address is present in various lists.
Epoch	Spanning Tree Protocol epoch number identifying when the MAC address was learned. Used for debugging.
Sequence number	Sequence number assigned to this MAC address. Used for debugging.
Learning mask	Mask of Packet Forwarding Engines where this MAC address was learned. Used for debugging.

**Table 19: show vpls mac-table Output fields (Continued)**

Field Name	Field Description
IPC generation	Creation time of the logical interface when this MAC address was learned. Used for debugging.

## Sample Output

### show vpls mac-table

```
user@host> show vpls mac-table
MAC flags (S -static MAC, D -dynamic MAC,
          SE -Statistics enabled, NM -Non configured MAC)
```

```
Routing instance : vpls_ldp1
```

```
VLAN : 223
```

MAC address	MAC flags	Logical interface
00:00:5e:00:53:5d	D	ge-0/2/5.400

```
MAC flags (S -static MAC, D -dynamic MAC,
          SE -Statistics enabled, NM -Non configured MAC)
```

```
Routing instance : vpls_red
```

```
VLAN : 401
```

MAC address	MAC flags	Logical interface
00:00:5e:00:53:12	D	lsi.1051138
00:00:5e:00:53:f0	D	lsi.1051138

### show vpls mac-table (with Layer 2 Services over GRE Interfaces)

```
user@host> show vpls mac-table
MAC flags (S -static MAC, D -dynamic MAC, L -locally learned
          SE -Statistics enabled, NM -Non configured MAC, R -Remote PE MAC)
```

```
Routing instance : vpls_4site:1000
```

Bridging domain : __vpls_4site:1000__,		MAC	MAC	Logical
address	flags	interface		
00:00:5e:00:53:f4	D,SE	ge-4/2/0.1000		
00:00:5e:00:53:33	D,SE	lsi.1052004		
00:00:5e:00:53:32	D,SE	lsi.1048840		
00:00:5e:00:53:14	D,SE	lsi.1052005		
00:00:5e:00:53:f7	D,SE	gr-1/2/10.10		

### show vpls mac-table (with VXLAN enabled)

```
user@host> show vpls mac-table
MAC flags (S -static MAC, D -dynamic MAC, L -locally learned
          SE -Statistics enabled, NM -Non configured MAC, R -Remote PE MAC)

Routing instance : vpls_4site:1000
Bridging domain : __vpls_4site:1000__, VLAN : 4094,4093
VXLAN: Id : 300, Multicast group: 233.252.0.1
  MAC          MAC      Logical
  address      flags    interface
  00:00:5e:00:53:f4 D,SE    ge-4/2/0.1000
  00:00:5e:00:53:33 D,SE    lsi.1052004
  00:00:5e:00:53:32 D,SE    lsi.1048840
  00:00:5e:00:53:14 D,SE    lsi.1052005
  00:00:5e:00:53:f7 D,SE    vtep.1052010
  00:00:5e:00:53:3f D,SE    vtep.1052011
```

### show vpls mac-table age (for GE interface)

```
user@host> show vpls mac-table age 00:00:5e:00:53:1a instance vpls_instance_1
MAC Entry Age information
Current Age: 4 seconds
```

### show vpls mac-table age (for AE interface)

```
user@host> show vpls mac-table age 000:00:5e:00:53:1a instance vpls_instance_1
MAC Entry Age information
```

Current Age on FPC1: 102 seconds

Current Age on FPC2: 94 seconds

## show vpls mac-table count

```
user@host> show vpls mac-table count
```

0 MAC address learned in routing instance \_\_example\_private1\_\_

MAC address count per interface within routing instance:

Logical interface	MAC count
lc-0/0/0.32769	0
lc-0/1/0.32769	0
lc-0/2/0.32769	0
lc-2/0/0.32769	0
lc-0/3/0.32769	0
lc-2/1/0.32769	0
lc-9/0/0.32769	0
lc-11/0/0.32769	0
lc-2/2/0.32769	0
lc-9/1/0.32769	0
lc-11/1/0.32769	0
lc-2/3/0.32769	0
lc-9/2/0.32769	0
lc-11/2/0.32769	0
lc-11/3/0.32769	0
lc-9/3/0.32769	0

MAC address count per learn VLAN within routing instance:

Learn VLAN ID	MAC count
0	0

1 MAC address learned in routing instance vpls\_ldp1

MAC address count per interface within routing instance:

Logical interface	MAC count
lsi.1051137	0
ge-0/2/5.400	1

MAC address count per learn VLAN within routing instance:

Learn VLAN ID	MAC count
0	1

1 MAC address learned in routing instance vpls\_red

MAC address count per interface within routing instance:

Logical interface	MAC count
ge-0/2/5.300	1

MAC address count per learn VLAN within routing instance:

Learn VLAN ID	MAC count
0	1

### show vpls mac-table detail

```
user@host> show vpls mac-table detail
```

MAC address: 00:00:5e:00:53:5d

Routing instance: vpls\_ldp1

Learning interface: ge-0/2/5.400

Layer 2 flags: in\_ifd, in\_ifl, in\_vlan, kernel

Epoch: 0 Sequence number: 1

Learning mask: 0x1 IPC generation: 0

MAC address: 00:00:5e:00:53:5d

Routing instance: vpls\_red

Learning interface: ge-0/2/5.300

Layer 2 flags: in\_ifd, in\_ifl, in\_vlan, kernel

Epoch: 0 Sequence number: 1

Learning mask: 0x1 IPC generation: 0

### show vpls mac-table extensive

```
user@host> show vpls mac-table extensive
```

MAC address: 00:00:5e:00:53:00

Routing instance: vpls\_1

Bridging domain: \_\_vpls\_1\_\_, VLAN : NA

Learning interface: lsi.1049165

Base learning interface: lsi.1049165

Layer 2 flags: in\_hash,in\_ifd,in\_ifl,in\_vlan,in\_rtt,kernel,in\_ifbd

Epoch: 0 Sequence number: 1

Learning mask: 0x00000001

```

MAC address: 00:00:5e:00:53:01
Routing instance: vpls_1
  Bridging domain: __vpls_1__, VLAN : NA
  Learning interface: lsi.1049165
  Base learning interface: lsi.1049165
  Layer 2 flags: in_hash,in_ifd,in_ifl,in_vlan,in_rtt,kernel,in_ifbd
  Epoch: 0                               Sequence number: 1
  Learning mask: 0x00000001

```

```

MAC address: 00:00:5e:00:53:02
Routing instance: vpls_1
  Bridging domain: __vpls_1__, VLAN : NA
  Learning interface: lsi.1049165
  Base learning interface: lsi.1049165
  Layer 2 flags: in_hash,in_ifd,in_ifl,in_vlan,in_rtt,kernel,in_ifbd
  Epoch: 0                               Sequence number: 1
  Learning mask: 0x00000001

```

```

MAC address: 00:00:5e:00:53:03
Routing instance: vpls_1
  Bridging domain: __vpls_1__, VLAN : NA
  Learning interface: lsi.1049165
  Base learning interface: lsi.1049165
  Layer 2 flags: in_hash,in_ifd,in_ifl,in_vlan,in_rtt,kernel,in_ifbd
  Epoch: 0                               Sequence number: 1
  Learning mask: 0x00000001

```

## Release Information

Command introduced in Junos OS Release 8.5.

## Verifying VXLAN Reachability

### IN THIS SECTION

● Purpose | 247

- Action | 247
- Meaning | 247

Purpose

On the local VTEP, verify that there is connectivity with the remote VTEP.

Action

```
user@switch> show ethernet-switching vxlan-tunnel-end-point remote
Logical System Name      Id  SVTEP-IP      IFL  L3-Idx
<default>                0   10.1.1.2      lo0.0  0
RVTEP-IP                IFL-Idx  NH-Id
10.1.1.2                559      1728
VNID                     MC-Group-IP
100                     233.252.0.1
```

Meaning

The remote VTEP is reachable because its IP address appears in the output. The output also shows that the VXLAN (VNI 100) and corresponding multicast group are configured correctly on the remote VTEP.

RELATED DOCUMENTATION

- Understanding VXLANs*
- Manually Configuring VXLANs on QFX Series and EX4600 Switches*
- Examples: Manually Configuring VXLANs on QFX Series and EX4600 Switches*

# Verifying That a Local VXLAN VTEP Is Configured Correctly

IN THIS SECTION

- Purpose | 248
- Action | 248
- Meaning | 248

## Purpose

Verify that a local VTEP is correct.

## Action

```
user@switch> show ethernet-switching vxlan-tunnel-end-point source
```

Logical System Name	Id	SVTEP-IP	IFL	L3-Idx
<default>	0	10.1.1.1	lo0.0	0
L2-RTT	Bridge Domain		VNID	MC-Group-IP
default-switch	VLAN1+100		100	233.252.0.1

## Meaning

The output shows the correct tunnel source IP address (loopback address), VLAN, and multicast group for the VXLAN.

RELATED DOCUMENTATION

| *Understanding VXLANs*



## Verifying MAC Learning from a Remote VTEP

### IN THIS SECTION

- Purpose | 249
- Action | 249
- Meaning | 249

### Purpose

Verify that a local VTEP is learning MAC addresses from a remote VTEP.

### Action

```
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)

Ethernet switching table : 2 entries, 2 learned

Routing instance : default-switch

Vlan	MAC	MAC	Age	Logical
name	address	flags		interface
VLAN1	00:00:00:ff:ff:ff	D	-	vtep.12345
VLAN1	00:10:94:00:00:02	D	-	xe-0/0/0.0

### Meaning

The output shows the MAC addresses learned from the remote VTEP (in addition to those learned on the normal Layer 2 interfaces). It also shows the logical name of the remote VTEP interface (vtep.12345 in the above output).

### RELATED DOCUMENTATION

*Understanding VXLANs*

*Manually Configuring VXLANs on QFX Series and EX4600 Switches*

## Understanding Overlay ping and traceroute Packet Support

### IN THIS SECTION

- [Overlay ping and traceroute Functionality | 251](#)
- [Overlay OAM Packet Format for UDP Payloads | 251](#)

In a virtualized overlay network, existing ping and traceroute mechanisms do not provide enough information to determine whether or not connectivity is established throughout the network. The existing ping and traceroute commands can only verify the basic connectivity between two endpoints in the underlying physical network, but not in the overlay network. For example, you can issue the existing ping command on a Juniper Networks device that functions as a virtual tunnel endpoint (VTEP) to another Juniper Networks devices that also functions as a VTEP in a Virtual Extensible LAN (VXLAN) overlay. In this situation, the ping output might indicate that the connection between the source and destination VTEPs is up and running despite the fact that one of the endpoints (physical servers upon which applications directly run) is not reachable.

Starting with Junos OS Release 14.1X53-D30 for QFX5100 switches, Release 16.1 for EX9200 switches, and Release 16.2 for MX Series routers, overlay ping and traceroute are introduced as troubleshooting tools for overlay networks.

For ping and traceroute mechanisms to work in overlay networks, the ping and traceroute packets, also referred to collectively as Operations, Administration, and Management (OAM) packets, must be encapsulated with the same VXLAN UDP headers (outer headers) as the data packets forwarded over the overlay segment. This implementation ensures that transit nodes forward the OAM packets in the same way as a data packet for that particular overlay segment.

If any connectivity issues arise for a particular data flow, the overlay OAM packet corresponding to the flow would experience the same connectivity issues as the data packet for that flow.

When using ping overlay and traceroute overlay, keep the following in mind:

- The only tunnel type supported is VXLAN tunnels.
- The VTEPs in the overlay network that send and receive the overlay ping packets must be Juniper Networks devices that support overlay ping and traceroute.

### Overlay ping and traceroute Functionality

Overlay ping and traceroute packets are sent as User Datagram Protocol (UDP) echo requests and replies and are encapsulated in the VXLAN header. VTEPs, which initiate and terminate overlay tunnels, send and receive overlay OAM packets. Overlay ping and traceroute are supported only in VXLAN overlay networks in which the sending and receiving VTEPs are both Juniper Networks devices.

The overlay ping functionality validates both the data plane and the MAC address and IP address of the VTEPs. This additional validation is different from the more commonly known IP ping functionality where the actual destination replies to the echo request without the overlay segment context.

While tracing a route in a VXLAN overlay network, Juniper Networks devices that are along the route that support overlay traceroute additionally provide a timestamp. Third-party devices and Juniper Networks devices that do not support overlay traceroute do not provide this timestamp.

### Overlay OAM Packet Format for UDP Payloads

The format of overlay OAM packets depends on the type of payload that is carried in the tunnel. In the case of VXLAN tunnels, the inner packet is a Layer 2 packet.

**NOTE:** Only Layer 2 UDP payloads are supported.

Figure 11 on page 251 shows complete headers on a VXLAN-encapsulated overlay OAM packet.

Figure 11: VXLAN-Encapsulated Overlay OAM Packet



- Outer Ethernet header—Contains the source MAC (SMAC) and destination MAC (DMAC) addresses of directly connected nodes in the physical network. These addresses change at every hop.
- Outer IP header—Contains the source and destination IP addresses of the Juniper Networks devices that function as the VTEPs that initiate and terminate the tunnel.
- Outer UDP header—Contains the source port associated with the flow entropy and destination port. The source port is an internally calculated hash value. The destination port is the standard UDP port (4789) used for VXLAN.
- VXLAN header—Contains the VXLAN Network Identifier (VNI) or the segment ID of the VXLAN, and new router alert (RA) flag bits.

- Inner Ethernet header—Contains a control MAC address (00-00-5E-90-xx-xx) for both the SMAC and DMAC. This address is not forwarded out of the VTEP. Alternatively, the SMAC can be set to a non-control MAC address. However, if a non-control MAC address is used, the VTEP must not learn the SMAC from the overlay OAM packets.
- Inner IP header—Contains the source IP address that can be set to the IP address of the endpoint or source VTEP. The destination IP address can be set to the 127/8 address, which ensures that the overlay OAM packet is not forwarded out of the ports of the Juniper Networks device that is configured as a VTEP.
- Inner UDP header—Contains a new reserved value used in the destination port field in the inner UDP header. This value identifies the incoming UDP packet as an overlay OAM packet.
- Inner UDP payload—Contains all of the overlay OAM-specific message format and type, length, and value (TLV) definitions.

The OAM-specific message type is one of the following:

Value	What it means
1	Echo Request
2	Echo Reply

Reply Mode Values:-

Value	What it means
1	Do not reply
2	Reply via an IPv4/IPv6 UDP Packet
3	Reply via Overlay Segment

The TLV definition for VXLAN ping is as follows:

0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
	Type = 1(VXLAN ping IPv4)		Length																												
+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
	VXLAN VNI																														
+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
	IPv4 Sender Address																														
+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+

## Multiple Routing Instance Support

Starting in Junos OS Release 19.3R1, you can use the `ping overlay` and `traceroute overlay` commands to verify connectivity and detect fault in a static VxLAN tunnel with multiple routing instances. The ping and traceroute packets created for the `ping overlay` and `traceroute overlay` commands follow the same underlay network path as the data packets. This allow you to verify the connectivity between two VTEPs in the overlay VxLAN tunnel. The devices that are configured as the source and destination VTEP must both be running a Junos OS release that supports multiple routing instance, but the transit devices do not.

Release History Table

Release	Description
19.3R1	Starting in Junos OS Release 19.3R1, you can use the ping overlay and traceroute overlay commands to verify connectivity and detect fault in a static VxLAN tunnel with multiple routing instances.
14.1X53-D30	Starting with Junos OS Release 14.1X53-D30 for QFX5100 switches, Release 16.1 for EX9200 switches, and Release 16.2 for MX Series routers, overlay ping and traceroute are introduced as troubleshooting tools for overlay networks.

RELATED DOCUMENTATION

*Example: Troubleshooting a VXLAN Overlay Network By Using Overlay Ping and Traceroute on QFX Series Switches*

[ping overlay | 268](#)

[traceroute overlay | 275](#)

# Example: Troubleshooting a VXLAN Overlay Network By Using Overlay Ping and Traceroute for MX Series Routers

IN THIS SECTION

- [Requirements | 255](#)
- [Overview and Topology | 255](#)
- [Configuration | 259](#)
- [Verification | 259](#)

In a Virtual Extensible LAN (VXLAN) overlay network, the existing ping and traceroute commands can verify the basic connectivity between two Juniper Networks devices that function as virtual tunnel endpoints (VTEPs) in the underlying physical network. However, in between the two VTEPs, there could be multiple routes through intermediary devices, and the ping and traceroute packets might successfully reach their destinations, while a connectivity issue exists in another route along which the data packets are typically forwarded to reach their destination.

With the introduction of the `overlay` parameter and other options in Junos OS Release 16.2 for MX Series routers, you can use the `ping` and `traceroute` commands to troubleshoot a VXLAN.

For ping and traceroute mechanisms to work in a VXLAN, the ping and traceroute packets, also referred to as Operations, Administration, and Management (OAM) packets, must be encapsulated with the same VXLAN headers (outer headers) as the data packets forwarded over the VXLAN segment with possible connectivity issues. If any connectivity issues arise, the overlay OAM packet would experience the same issues as the data packet.

This example shows how to use overlay ping and traceroute on a VTEP to verify the following in a VXLAN:

- Scenario 1—Verify that a particular VXLAN is configured on another VTEP.
- Scenario 2—Verify that the MAC address of a particular endpoint is associated with a VXLAN on the remote VTEP.
- Scenario 3—Verify that no issues exist in a particular data flow between sending and receiving endpoints.

**NOTE:** When issuing the `ping overlay` and `traceroute overlay` commands, the source VTEP on which you issue the command and the destination VTEP that receives the ping packet must be Juniper Networks devices that support overlay ping and traceroute.

## Requirements

This example uses the following hardware and software components:

- Three physical servers on which applications directly run.
- Two MX Series routers running Junos OS Release 16.2 or later software. These routers function as VTEPs.
- Two Layer 3 routers, which can be Juniper Networks routers or routers provided by another vendor.

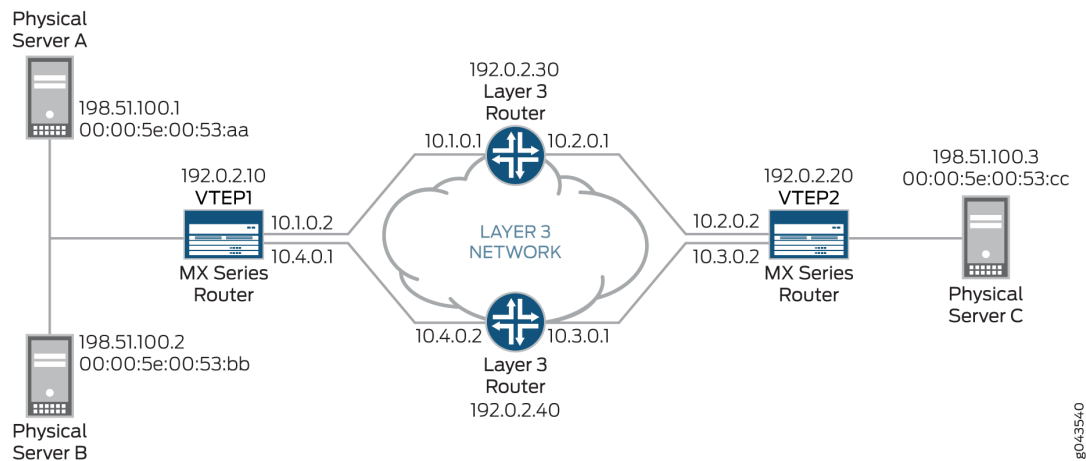
Before issuing the `ping overlay` and `traceroute overlay` commands, gather the information needed for each parameter— for example, IP addresses or MAC addresses— used for a particular scenario. See [Table 20 on page 257](#) to determine which parameters are used for each scenario.

## Overview and Topology

The VXLAN topology shown in [Figure 12 on page 256](#) includes physical servers A, B, and C on which applications directly run. The applications on physical servers A and B need to communicate with the applications on physical server C. These servers are on the same subnet, so the communication between

the applications occurs at the Layer 2 level, and VXLAN encapsulation or tunnels are used to transport their data packets over a Layer 3 network.

**Figure 12: Using Overlay Ping and Traceroute to Troubleshoot a VXLAN**



In this topology, there are two MX Series routers that function as VTEPs. VTEP1 initiates and terminates VXLAN tunnels for physical servers A and B, and VTEP2 does the same for physical server C. VTEP1 and VTEP2 are in VXLAN 100.

A data packet sent from physical server A is typically routed to the Layer 3 router with the IP address of 192.0.2.30 to reach physical server C.

In this VXLAN topology, a communication issue arises between physical servers A and C. To troubleshoot the issue with this data flow, you can initiate the ping overlay and traceroute overlay commands on VTEP1 (the source VTEP or tunnel-src) and specify that VTEP2 is the destination VTEP or tunnel-dst.

The ping overlay and traceroute overlay commands include several parameters. [Table 20 on page 257](#) explains the purpose and provides a value for each of the parameters used in the three scenarios.

[Table 20 on page 257](#) does not include all available ping overlay and traceroute overlay parameters. This example uses the default values of these omitted parameters.



**Table 20: Ping and Traceroute Overlay Parameter Values For Scenarios 1, 2, and 3**

ping overlay and traceroute overlay Parameters	Description	Scenario to Which Parameter Applies	Value
tunnel-type	Identifies type of tunnel that you are troubleshooting.	All	vxlan
vni	VXLAN network identifier (VNI) of VXLAN used in this example.	All	100
tunnel-src	IP address of VTEP1, on which you initiate overlay ping or traceroute.	All	192.0.2.10
tunnel-dst	IP address of VTEP2, which receives the overlay ping or traceroute packets.	All	192.0.2.20
mac	MAC address of physical server C, which is the destination endpoint.	Scenarios 2 and 3	00:00:5E:00:53:cc
count	Number of overlay ping requests that VTEP1 sends.  <b>NOTE:</b> The count parameter does not apply to overlay traceroute.	All	5
hash-source-mac	MAC address of physical server A, which is the source endpoint.	Scenario 3	00:00:5E:00:53:aa

**Table 20: Ping and Traceroute Overlay Parameter Values For Scenarios 1, 2, and 3 (Continued)**

ping overlay and traceroute overlay Parameters	Description	Scenario to Which Parameter Applies	Value
hash-destination-mac	MAC address of physical server C, which is the destination endpoint.  <b>NOTE:</b> When specifying this parameter for scenario 3, the MAC address must be the same MAC address as specified for the mac parameter.	Scenario 3	00:00:5E:00:53:cc
hash-source-address	IP address of physical server A.	Scenario 3	198.51.100.1
hash-destination-address	IP address of physical server C.	Scenario 3	198.51.100.3
hash-protocol	A value for the protocol used in the data flow.	Scenario 3	17
hash-source-port	A value for the outer TCP/UDP source port.	Scenario 3	4456
hash-destination-port	A value for the outer UDP destination port.	Scenario 3	4540

Table 20 on page 257 includes several hash parameters, which are used for scenario 3. For each of these parameters, you must specify a value associated with the data flow that you are troubleshooting. Based on the values that you specify, the system calculates a VXLAN UDP header source port hash, which is included in the VXLAN UDP header of the overlay ping and traceroute packets. Including the calculated hash in the VXLAN UDP header enables the overlay ping and traceroute packets to emulate data packets in the flow that you are troubleshooting.

**BEST PRACTICE:** When using the hash parameters, we recommend that you specify a value for each parameter. This practice ensures that the overlay ping and traceroute processes are successful and that the output for each command is accurate. If you do not specify a value for one or more of the hash parameters, the system sends an OAM request that might include incorrect hash values and generates a warning message.

## Configuration

## Verification

### IN THIS SECTION

- [Scenario-1: Verifying That VXLAN 100 Is Configured on VTEP2 | 259](#)
- [Scenario 2: Verifying That the MAC Address of the Destination Endpoint Is on VTEP2 | 262](#)
- [Scenario 3: Verifying a Data Flow | 266](#)

This section includes the following verification tasks:

### Scenario-1: Verifying That VXLAN 100 Is Configured on VTEP2

#### Purpose

Verify that a VXLAN with the VNI of 100 is configured on VTEP2. You can use either overlay ping or traceroute to perform this verification.

#### Action

##### Overlay Ping

On VTEP1, initiate an overlay ping:

```
user@switch> ping overlay tunnel-type vxlan vni 100 tunnel-src 192.0.2.10 tunnel-dst 192.0.2.20
count 5
ping-overlay protocol vxlan

vni 100
```

```

        tunnel src ip 192.0.2.10
tunnel dst ip 192.0.2.20
mac address 00:00:00:00:00:00
count 5
ttl 255

```

WARNING: following hash-parameters are missing -  
hash computation may not succeed

```

end-host smac
end-host dmac
end-host src ip
end-host dst ip
end-host protocol
end-host l4-src-port
end-host l4-dst-port

```

Request for seq 1, to 192.0.2.20, at 09-24 22:03:16 PDT.033 msecs

Response for seq 1, from 192.0.2.20, at 09-24 22:03:16 PDT.036 msecs, rtt 10 msecs

Overlay-segment not present at RVTEP 192.0.2.20

Request for seq 2, to 192.0.2.20, at 09-24 22:03:16 PDT.044 msecs

Response for seq 2, from 192.0.2.20, at 09-24 22:03:16 PDT.046 msecs, rtt 10 msecs

Overlay-segment not present at RVTEP 192.0.2.20

Request for seq 3, to 192.0.2.20, at 09-24 22:03:16 PDT.054 msecs

Response for seq 3, from 192.0.2.20, at 09-24 22:03:16 PDT.057 msecs, rtt 10 msecs

Overlay-segment not present at RVTEP 192.0.2.20

Request for seq 4, to 192.0.2.20, at 09-24 22:03:16 PDT.065 msecs

Response for seq 4, from 192.0.2.20, at 09-24 22:03:16 PDT.069 msecs, rtt 10 msecs

Overlay-segment not present at RVTEP 192.0.2.20

Request for seq 5, to 192.0.2.20, at 09-24 22:03:16 PDT.076 msecs

Response for seq 5, from 192.0.2.20, at 09-24 22:03:16 PDT.079 msecs, rtt 10 msecs

Overlay-segment not present at RVTEP 192.0.2.20

## Overlay Traceroute

On VTEP1, initiate an overlay traceroute:

```
user@switch> traceroute overlay tunnel-type vxlan vni 100 tunnel-src 192.0.2.10 tunnel-dst 192.0.2.20
```

```
traceroute-overlay protocol vxlan
```

```

          vni 100
    tunnel src ip 192.0.2.10
    tunnel dst ip 192.0.2.20
    mac address 00:00:00:00:00:00
    ttl 255

```

WARNING: following hash-parameters are missing -  
hash computation may not succeed

```

end-host smac
end-host dmac
end-host src ip
end-host dst ip
end-host protocol
end-host l4-src-port
end-host l4-dst-port

```

ttl	Address	Sender Timestamp	Receiver Timestamp	Response Time
1	10.1.0.2	09-25 00:51:10 PDT.599 msecs	*	10 msecs
2	192.0.2.20	09-25 00:51:10 PDT.621 msecs	09-25 00:51:10 PDT.635 msecs	21 msecs

Overlay-segment not present at RVTEP 192.0.2.20

## Meaning

The sample overlay ping output indicates the following:

- VTEP1 sent five ping requests to VTEP2, and VTEP2 responded to each request.



```
tunnel dst ip 192.0.2.20
mac address 00:00:5E:00:53:cc
count 5
ttl 255
```

WARNING: following hash-parameters are missing -  
hash computation may not succeed

```
end-host smac
end-host dmac
end-host src ip
end-host dst ip
end-host protocol
end-host l4-src-port
end-host l4-dst-port
```

Request for seq 1, to 192.0.2.20, at 09-24 23:53:54 PDT.089 msec

Response for seq 1, from 192.0.2.20, at 09-24 23:53:54 PDT.089 msec, rtt 6 msec

Overlay-segment present at RVTEP 192.0.2.20

End-System Not Present

Request for seq 2, to 192.0.2.20, at 09-24 23:53:54 PDT.096 msec

Response for seq 2, from 192.0.2.20, at 09-24 23:53:54 PDT.100 msec, rtt 10 msec

Overlay-segment present at RVTEP 192.0.2.20

End-System Not Present

Request for seq 3, to 192.0.2.20, at 09-24 23:53:54 PDT.107 msec

Response for seq 3, from 192.0.2.20, at 09-24 23:53:54 PDT.111 msec, rtt 10 msec

Overlay-segment present at RVTEP 192.0.2.20

End-System Not Present

Request for seq 4, to 192.0.2.20, at 09-24 23:53:54 PDT.118 msec

```
Response for seq 4, from 192.0.2.20, at 09-24 23:53:54 PDT.122 msecs, rtt 11 msecs

Overlay-segment present at RVTEP 192.0.2.20

End-System Not Present

Request for seq 5, to 192.0.2.20, at 09-24 23:53:54 PDT.129 msecs

Response for seq 5, from 192.0.2.20, at 09-24 23:53:54 PDT.133 msecs, rtt 10 msecs

Overlay-segment present at RVTEP 192.0.2.20

End-System Not Present
```

Overlay Traceroute

On VTEP1, initiate an overlay traceroute:

```
user@switch> traceroute overlay tunnel-type vxlan vni 100 tunnel-src 192.0.2.10 tunnel-dst
192.0.2.20 mac 00:00:5E:00:53:cc
traceroute-overlay protocol vxlan

vni 100
tunnel src ip 192.0.2.10
tunnel dst ip 192.0.2.20
mac address 00:00:5E:00:53:cc
ttl 255

WARNING: following hash-parameters are missing -
hash computation may not succeed

end-host smac
end-host dmac
end-host src ip
end-host dst ip
end-host protocol
end-host l4-src-port
end-host l4-dst-port

ttl Address Sender Timestamp Receiver Timestamp Response Time
```



```

1  10.1.0.1  09-25 00:56:17 PDT.663 msecs      *          10 msecs
2  192.0.2.20  09-25 00:56:17 PDT.684 msecs  09-25 00:56:17 PDT.689 msecs  11 msecs

```

Overlay-segment present at RVTEP 192.0.2.20

End-System not Present

## Meaning

The sample overlay ping output indicates the following:

- VTEP1 sent five ping requests to VTEP2, and VTEP2 responded to each request.
- VTEP2 verified that the VNI of 100 is configured (Overlay-segment present at RVTEP 192.0.2.20) but that the MAC address of physical server C is not in the forwarding table (End-System Not Present). VTEP2 included this information in its response to VTEP1.

The sample overlay traceroute output indicates the following:

- Upon receiving an overlay traceroute packet with a TTL value of 1 hop, the Layer 3 router responds to VTEP1.
- Upon receiving an overlay traceroute packet with a TTL value of 2 hops, VTEP2 responds to VTEP1.
- VTEP2 verified that the VNI of 100 is configured (Overlay-segment present at RVTEP 192.0.2.20) and that the MAC address of physical server C is in the forwarding table (End-System Present). VTEP2 included this information in its response to VTEP1.

**NOTE:** The asterisk (\*) in the Receiver Timestamp column of the overlay traceroute output indicates that the Layer 3 router that received the overlay traceroute packet is not a Juniper Networks device or is a Juniper Networks device that does not support overlay traceroute.

Given that the output of both overlay ping and traceroute indicates that the MAC address of physical server C is not known by VTEP2, you must further investigate to determine why this MAC address is not in the forwarding table of VTEP2.

### Scenario 3: Verifying a Data Flow

#### Purpose

Verify that there are no issues that might impede the flow of data from physical server A to physical server C. The networking devices that support this flow include VTEP1, the Layer 3 router with the IP address of 192.0.2.30, and VTEP2 (see [Figure 12 on page 256](#)).

Initially, use overlay ping, and if the overlay ping results indicate an issue, use overlay traceroute to determine which device in the path has an issue.

With both overlay ping and traceroute, use the hash parameters to specify information about the devices in this data flow so that the system can calculate a VXLAN UDP header source port hash, which is included in the VXLAN UDP header of the overlay ping and traceroute packets. With the calculated hash included in the VXLAN UDP header, the overlay ping and traceroute packets can emulate data packets in this flow, which should produce more accurate ping and traceroute results.

#### Action

##### Overlay Ping

On VTEP1, initiate an overlay ping:

```
user@switch> ping overlay tunnel-type vxlan vni 100 tunnel-src 192.0.2.10 tunnel-dst 192.0.2.20
mac 00:00:5E:00:53:cc count 5 hash-source-mac 00:00:5E:00:53:aa hash-destination-mac
00:00:5E:00:53:cc hash-source-address 198.51.100.1 hash-destination-address 198.51.100.3 hash-
protocol 17 hash-source-port 4456 hash-destination-port 4540
ping-overlay protocol vxlan

vni 100
tunnel src ip 192.0.2.10
tunnel dst ip 192.0.2.20
mac address 00:00:5E:00:53:cc
count 5
ttl 255

hash-parameters:
    input-ifd-idx 653
    end-host smac 00:00:5E:00:53:aa
    end-host dmac 00:00:5E:00:53:cc
    end-host src ip 198.51.100.1
    end-host dst ip 198.51.100.3
    end-host protocol 17
```

```
end-host l4-src-port 4456
end-host l4-dst-port 4540end-host vlan 150
Request for seq 1, to 192.0.2.20, at 09-24 19:15:33 PDT.352 msec
Request for seq 2, to 192.0.2.20, at 09-24 19:15:33 PDT.363 msec
Request for seq 3, to 192.0.2.20, at 09-24 19:15:33 PDT.374 msec
Request for seq 4, to 192.0.2.20, at 09-24 19:15:33 PDT.385 msec
Request for seq 5, to 192.0.2.20, at 09-24 19:15:33 PDT.396 msec
```

Overlay Traceroute

If needed, on VTEP1, initiate an overlay traceroute:

```
user@switch> traceroute overlay tunnel-type vxlan vni 100 tunnel-src 192.0.2.10 tunnel-dst
192.0.2.20 mac 00:00:5E:00:53:cc hash-source-mac 00:00:5E:00:53:aa hash-destination-mac
00:00:5E:00:53:cc hash-source-address 198.51.100.1 hash-destination-address 198.51.100.3 hash-
protocol 17 hash-source-port 4456 hash-destination-port 4540
traceroute-overlay protocol vxlan

vni 100
tunnel src ip 192.0.2.10
tunnel dst ip 192.0.2.20
mac address 00:00:5E:00:53:cc
ttl 255

hash-parameters:
input-ifd-idx 653
end-host smac 00:00:5E:00:53:aa
end-host dmac 00:00:5E:00:53:cc
end-host src ip 198.51.100.1
end-host dst ip 198.51.100.3
end-host protocol 17
end-host l4-src-port 4456
end-host l4-dst-port 4540

ttl Address      Sender Timestamp          Receiver Timestamp      Response Time
1  10.1.0.1      09-25 00:56:17 PDT.663 msec          *                       10 msec
```

## Meaning

The sample overlay ping output indicates that VTEP1 sent five ping requests to VTEP2, but VTEP2 did not respond to any of the requests. The lack of response from VTEP2 indicates that a connectivity issue exists along the path between VTEP1 and the Layer 3 router or the path between the Layer 3 router and VTEP2.

To further troubleshoot in which path the issue lies, overlay traceroute is used. The sample overlay traceroute output indicates the following:

- Upon receiving an overlay traceroute packet with a TTL value of 1 hop, the Layer 3 router responds to VTEP1, which indicates that the path between VTEP1 and the Layer 3 router is up.
- VTEP2 does not respond to the overlay traceroute packet, which indicates that the path between the Layer 3 router and VTEP2 might be down.

**NOTE:** The asterisk (\*) in the Receiver Timestamp column of the overlay traceroute output indicates that the Layer 3 router that received the overlay traceroute packet is not a Juniper Networks device or is a Juniper Networks device that does not support overlay traceroute.

Given that the overlay traceroute output indicates that there is a connectivity issue between the Layer 3 router and VTEP2, you must further investigate this path segment to determine the source of the issue.

## RELATED DOCUMENTATION

[Understanding Overlay ping and traceroute Packet Support | 250](#)

[ping overlay | 268](#)

[traceroute overlay | 275](#)

## ping overlay

### IN THIS SECTION

● [Syntax | 269](#)

● [Description | 269](#)

● [Options | 270](#)

- Required Privilege Level | 272
- Output Fields | 272
- Sample Output | 273
- Release Information | 274

## Syntax

```
ping overlay
<tunnel-type>
vni vni
tunnel-src ip-source-address
tunnel-dst ip-destination-address
<mac mac-address>
<count requests>
<ttl value>
<hash-source-mac source-mac-address>
<hash-destination-mac destination-mac-address>
<hash-source-address source-IP-address>
<hash-destination-address destination-IP-address>
<hash-vlan vlan-id>
<hash-input-interface input-interface>
<hash-protocol protocol-id>
<hash-source-port source-layer4-port>
<hash-destination-port destination-layer4-port>
```

## Description

Verify the presence of the Virtual Extensible LAN (VXLAN) tunnel endpoints (VTEPs), which can originate and terminate VXLAN tunnels, and service connectivity within the context of the overlay VXLAN segment. Use `ping overlay` as a fault detection tool to determine failure within an overlay VXLAN tunnel. Type **Ctrl+c** to interrupt a `ping overlay` command.

**NOTE:** The `ping overlay` command is not supported for IPv6.

**NOTE:** The `ping overlay` command is not supported when there are multiple virtual-switch routing instances.

## Options

**tunnel-type** (Optional) Specify the overlay tunnel type used in a virtualized environment such as: VXLAN, Network Virtualization using Generic Routing Encapsulation (NVGRE), MPLS over User Datagram Protocol (UDP), and MPLS over General Routing Encapsulation (GRE) tunnels.

**NOTE:** Only VXLAN overlay tunnel types are supported.

**vni *vni*** Specify the VNI of the VXLAN overlay segment.

**tunnel-src *ip-source-address*** Specify the IP address of the source entity at the end of the tunnel, such as the source VTEP.

**tunnel-dst *ip-destination-address*** Specify the IP address of the destination entity at the end of the tunnel, such as a remote VTEP.

**mac *mac-address*** (Optional) Include the physical or hardware address on the end host system you are trying to reach.

**count *requests*** (Optional) Number of ping requests to send.

For QFX and EX9200 switches, the range of values is **1** through **65,535**. The default value is **10**.

For MX Series routers, the range of values is **1** through **2,000,000,000**. The default value is **5**.

**ttl *value*** (Optional) Time-to-live (TTL) value to include in the ping request.

For QFX and EX9200 switches, the range of values is **1** through **255**. The default value is **255**.

For MX Series routers, the range of values is **0** through **255**. The default value is **255**.

**hash-source-mac *source-mac-address*** (Optional) Specify the MAC address of the source end host system.

**NOTE:** The hash parameters provide values that correspond to a particular data flow that the `ping overlay` command debugs. Based on the values that you specify, the system calculates a VXLAN UDP header source port hash, which is included in the VXLAN UDP header of the overlay ping and traceroute packets. Including the calculated hash in the VXLAN header enables the overlay ping and traceroute packets to emulate data packets in the flow that you are troubleshooting.

When using the hash parameters, we recommend that you specify a value for each parameter. The exception to this guideline is the `hash-vlan` parameter, which you do not have to use if the source endpoint is not a member of a VLAN. This practice ensures that the overlay ping and traceroute processes are successful and that the output for each command is accurate. If you do not specify a value for one or more of the hash parameters, the system sends an OAM request that might include incorrect hash values and generates a warning message.

Hash computation supports TCP and UDP protocols only.

<b>hash-destination-mac</b> <i>destination-mac-address</i>	(Optional) Specify the MAC address of the destination end host system.
<b>hash-source-address</b> <i>source-IP-address</i>	(Optional) Specify the IP address of the source end host system.
<b>hash-destination-address</b> <i>destination-IP-address</i>	(Optional) Specify the IP address of the destination end host system.
<b>hash-vlan</b> <i>vlan-id</i>	(Optional, QFX switches only) Specify the VLAN ID of the end host system.
<b>hash-input-interface</b> <i>input-interface</i>	(Optional, QFX switches only) Specify the ingress interface of the flow on the Juniper Networks device.
<b>hash-protocol</b> <i>protocol-id</i>	(Optional) Specify the TCP/UDP IP protocol ID. The range of values is <b>1</b> through <b>255</b> .
<b>hash-source-port</b> <i>source-layer4-port</i>	(Optional) Specify the Layer 4 source port. The range of values is <b>1</b> through <b>65,535</b> .
<b>hash-destination-port</b> <i>destination-layer4-port</i>	(Optional) Specify the Layer 4 destination port. The range of values is <b>1</b> through <b>65,535</b> .

## Required Privilege Level

network

## Output Fields

When you enter this command, you are provided feedback on the status of your request. An exclamation point (!) indicates that an echo reply was received. A period (.) indicates that an echo reply was not received within the timeout period. An **x** indicates that an echo reply was received with an error code. These packets are not counted in the received packets count. They are accounted for separately.

[Table 21 on page 272](#) lists the output fields for the `ping overlay` command. Output fields are listed in the approximate order in which they appear.

**Table 21: ping overlay Output Fields**

Field Name	Field Description
vni	The VNI of the VXLAN overlay segment.
tunnel src ip	The IP address of the source end of the tunnel.
tunnel dst ip	The IP address of the destination end of the tunnel.
mac address	The physical or hardware address on the end host system you are trying to reach.
count	Number of ping requests sent.
ttl	TTL value for maximum number of pings.
hash-parameters	The hash parameters provide the input-interface, source MAC address, destination MAC address, source IP address, destination IP address, and the VLAN of the two end hosts within an overlay segment. Hash parameters enable platform-specific hash computation to use as the source port in the outer UDP header.
Request/Response for seq <i>x</i> to/ from <i>address</i> at <i>timestamp</i>	Number of ping request and response counts for determining overlay segments in tunnel.



## Sample Output

### run ping overlay

```

user@host> ping overlay tunnel-type vxlan vni 100 tunnel-src 192.0.2.10 tunnel-dst 192.0.2.20
mac 00:00:5E:00:53:cc count 5 hash-source-mac 00:00:5E:00:53:aa hash-destination-mac
00:00:5E:00:53:cc hash-source-address 198.51.100.1 hash-destination-address 198.51.100.3 hash-
vlan 150 hash-input-interface xe-0/0/2 hash-protocol 17 hash-source-port 4456 hash-destination-
port 4540
vni 100
    tunnel src ip 192.0.2.10
    tunnel dst ip 192.0.2.20
    mac address 00:00:5E:00:53:cc
    count 5
    ttl 255

    hash-parameters:
        input-ifd-idx 653
        end-host smac 00:00:5E:00:53:aa
        end-host dmac 00:00:5E:00:53:cc
        end-host src ip 198.51.100.1
        end-host dst ip 198.51.100.3
        end-host protocol 17
        end-host l4-src-port 4456
        end-host l4-dst-port 4540
        end-host vlan 150

Request for seq 1, to 192.0.2.20, at 09-24 19:15:33 PDT.352 msecs

Response for seq 1, from 192.0.2.20, at 09-24 19:15:33 PDT.359 msecs, rtt 11 msecs

Overlay-segment present at RVTEP 192.0.2.20

End-System Present

Request for seq 2, to 192.0.2.20, at 09-24 19:15:33 PDT.363 msecs

Response for seq 2, from 192.0.2.20, at 09-24 19:15:33 PDT.370 msecs, rtt 10 msecs

Overlay-segment present at RVTEP 192.0.2.20

```

End-System Present

Request for seq 3, to 192.0.2.20, at 09-24 19:15:33 PDT.374 msec

Response for seq 3, from 192.0.2.20, at 09-24 19:15:33 PDT.381 msec, rtt 10 msec

Overlay-segment present at RVTEP 192.0.2.20

End-System Present

Request for seq 4, to 192.0.2.20, at 09-24 19:15:33 PDT.385 msec

Response for seq 4, from 192.0.2.20, at 09-24 19:15:33 PDT.392 msec, rtt 10 msec

Overlay-segment present at RVTEP 192.0.2.20

End-System Present

Request for seq 5, to 192.0.2.20, at 09-24 19:15:33 PDT.396 msec

Response for seq 5, from 192.0.2.20, at 09-24 19:15:33 PDT.403 msec, rtt 11 msec

Overlay-segment present at RVTEP 192.0.2.20

End-System Present

## Release Information

Command introduced in Junos OS Release 14.1X53-D30.

## RELATED DOCUMENTATION

[Understanding Overlay ping and traceroute Packet Support | 250](#)

*Example: Troubleshooting a VXLAN Overlay Network By Using Overlay Ping and Traceroute on QFX Series Switches*

[traceroute overlay | 275](#)

## traceroute overlay

### IN THIS SECTION

- [Syntax | 275](#)
- [Description | 275](#)
- [Options | 276](#)
- [Required Privilege Level | 278](#)
- [Output Fields | 278](#)
- [Sample Output | 279](#)
- [Release Information | 280](#)

### Syntax

```
traceroute overlay
<tunnel-type>
vni vni
tunnel-src source-ip-address
tunnel-dst destination-ip-address
<mac mac-address>
<ttl value>
<hash-input-interface input-interface>
<hash-source-mac source-mac-address>
<hash-destination-mac destination-mac-address>
<hash-source-address source-IP-address>
<hash-destination-address destination-IP-address>
<hash-vlan vlan-id>
<hash-protocol protocol-id>
<hash-source-port source-layer4-port>
<hash-destination-port destination-layer4-port>
```

### Description

Display the route that packets take between two Virtual Extensible LAN (VXLAN) tunnel endpoints (VTEPs) and within the context of a VXLAN overlay segment. Use `traceroute overlay` as an isolation and

debugging tool to locate points of failure within an overlay VXLAN tunnel. The output is useful for diagnosing a point of failure in the path from the device to the destination host, and for addressing network traffic latency and throughput problems.

**NOTE:** The `traceroute overlay` command is not supported for IPv6.

**NOTE:** The `traceroute overlay` command is not supported when there are multiple virtual-switch routing instances.

### Options

**tunnel-type** (Optional) Specify the overlay tunnel type used in a virtualized environment such as: VXLAN, Network Virtualization using Generic Routing Encapsulation (NVGRE), MPLS over User Datagram Protocol (UDP), and MPLS over General Routing Encapsulation (GRE) tunnels.

**NOTE:** Only VXLAN overlay tunnel types are supported.

**vni *vni*** Specify the VNI of the VXLAN overlay segment.

- **Range:** 1 through 16,777,215

**tunnel-src *source-ip-address*** Specify the IP address of the source entity at end of the tunnel, such as the source VTEP.

**tunnel-dst *destination-ip-address*** Specify the IP address of the destination entity at the end of the tunnel, such as the remote VTEP.

**mac *mac-address*** (Optional) Include the physical or hardware address on the end host you are trying to reach.

**ttl *value*** (Optional) Time-to-live (TTL) value to include as the maximum number of hops in the traceroute request.

For MX Series routers, the range of values is **0** through **255**. The default value is **255**.

- **Range:** (QFX Series, EX9200 switches) 1 through 255

- **Default:** 255

**hash-source-mac**  
*source-mac-address*

(Optional) Specify the MAC address of the source end host.

**NOTE:** The hash parameters provide values that correspond to a particular data flow that the traceroute overlay command debugs. Based on the values that you specify, the system calculates a VXLAN UDP header source port hash, which is included in the VXLAN UDP header of the overlay ping and traceroute packets. Including the calculated hash in the VXLAN header enables the overlay ping and traceroute packets to emulate data packets in the flow that you are troubleshooting.

When using the hash parameters, we recommend that you specify a value for each parameter. The exception to this guideline is the hash-vlan parameter, which you do not have to use if the source endpoint is not a member of a VLAN. This practice ensures that the overlay ping and traceroute processes are successful and that the output for each command is accurate. If you do not specify a value for one or more of the hash parameters, the system sends an OAM request that might include incorrect hash values and generates a warning message.

Hash computation supports TCP and UDP protocols only.

**hash-destination-mac**  
*destination-mac-address*

(Optional) Specify the MAC address of the destination end host.

**hash-source-address**  
*source-IP-address*

(Optional) Specify the IP address of the source end host.

**hash-destination-address**  
*destination-IP-address*

(Optional) Specify the IP address of the destination end host.

**hash-vlan** *vlan-id*

(Optional, QFX Series switches only) Specify the VLAN ID of the end host.

- **Range:** 1 through 4094

**hash-input-interface**  
*interface-name*

(Optional, QFX Series switches only) Specify the ingress interface of the flow on the Juniper Networks device.

**hash-protocol**  
*protocol-id*

(Optional) Specify the TCP/UDP IP protocol ID of the end host.

- **Range:** 1 through 255

**hash-source-port**  
*source-layer4-port*

(Optional) Specify the Layer 4 source port of the end host.

- **Range:** 1 through 65,535

hash-destination-  
port *destination-  
layer4-port*

- (Optional) Specify the Layer 4 destination port of the end host.
- **Range:** 1 through 65,535

**Required Privilege Level**

network

**Output Fields**

Use the `traceroute overlay` command to determine overlay segments within a VXLAN tunnel. The output is useful for diagnosing a point of failure in the path from the device to the destination host, and for addressing network traffic latency and throughput problems.

[Table 22 on page 278](#) lists the output fields for the `traceroute overlay` command. Output fields are listed in the approximate order in which they appear.

**Table 22: traceroute overlay Output Fields**

Field Name	Field Description
vni	The VNI of the VXLAN overlay segment.
tunnel src ip	The IP address of the source end of the tunnel.
tunnel dst ip	The IP address of the destination end of the tunnel.
mac address	The physical or hardware address of the end host you are trying to reach.
ttl	TTL value for the maximum number of hops in the traceroute request.
hash-parameters	The hash parameters provide the input interface, source MAC address, destination MAC address, source IP address, destination IP address, and the VLAN ID of the two end hosts within an overlay segment. Hash parameters enable platform-specific hash computation to use as the source port in the outer UDP header.

**Table 22: traceroute overlay Output Fields (Continued)**

Field Name	Field Description
ttl	Number of hops remaining in the traceroute message. The TTL is decremented at each hop.
Address	The sending IPv4 address.
Sender Timestamp	Timestamp in microseconds when hop was sent.
Receiver Timestamp	Timestamp in microseconds when hop was received.
Response Time	Time in microseconds for traceroute to respond.

## Sample Output

### run traceroute overlay

```

user@host> traceroute overlay tunnel-type vxlan vni 100 tunnel-src 192.0.2.10 tunnel-dst
192.0.2.20 mac 00:00:5E:00:53:cc hash-source-mac 00:00:5E:00:53:aa hash-destination-mac
00:00:5E:00:53:cc hash-source-address 198.51.100.1 hash-destination-address 198.51.100.3 hash-
vlan 150 hash-input-interface xe-0/0/2 hash-protocol 17 hash-source-port 4456 hash-destination-
port 4540
traceroute-overlay protocol vxlan

vni 100
tunnel src ip 192.0.2.10
tunnel dst ip 192.0.2.20
mac address 00:00:5E:00:53:cc
ttl 255

hash-parameters:
    input-ifd-idx 653
    end-host smac 00:00:5E:00:53:aa
    end-host dmac 00:00:5E:00:53:cc
    end-host src ip 198.51.100.1
    end-host dst ip 198.51.100.3

```

```
end-host protocol 17
end-host l4-src-port 4456
end-host l4-dst-port 4540
end-host vlan 150

ttl  Address      Sender Timestamp      Receiver Timestamp      Response Time
1    10.1.0.1      09-25 00:56:17 PDT.663 msecs      *                        10 msecs
2    192.0.2.20   09-25 00:56:17 PDT.684 msecs      09-25 00:56:17 PDT.689 msecs  11 msecs

Overlay-segment  present at RVTEP 192.0.2.20

End-System  Present
```

Release Information

Command introduced in Junos OS Release 14.1X53-D30.

RELATED DOCUMENTATION

<a href="#">Understanding Overlay ping and traceroute Packet Support   250</a>
<i>Example: Troubleshooting a VXLAN Overlay Network By Using Overlay Ping and Traceroute on QFX Series Switches</i>
<a href="#">ping overlay   268</a>