Engineering Simplicity

# Data Center EVPN-VXLAN Fabric Architecture Guide

Published
2024-12-20

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

## YEAR 2000 NOTICE

## END USER LICENSE AGREEMENT

# Table of Contents

# About This Guide

Dive in to the Data Center EVPN-VXLAN Fabric Architecture Guide, which gives network designers and operators the information they need to build a state-of-the-art, multiservice, cloud data center network.

1

CHAPTER

# Data Center Fabric Blueprint Architecture—Overview

# About This Architecture Guide

The purpose of this guide is to provide networking professionals with the concepts and tools needed to build multiservice data center fabric networks.

The intended audience for this guide includes system integrators, infrastructure professionals, partners, and customers that are currently using or considering upgrading to a high-end IP fabric data center fabric architecture.

# Terminology

**IN THIS SECTION**

This section provides a summary of commonly used terms, protocols, and building block technologies used in creating and maintaining data center networks.

## Glossary Terms

- **ARP**—Address Resolution Protocol. A protocol defined in RFC 826 for mapping a logical IP address to a physical MAC address.
- **Backbone Device**—A device in the WAN cloud that is directly connected to a spine device or devices in a data center. Backbone devices are required in this reference topology to provide physical connectivity between data centers that are interconnected using a data center interconnect (DCI).
- **Border Leaf**—A device that typically has the sole purpose of providing a connection to one or more external devices. The external devices, for example, multicast gateways or data center gateways, provide additional functionality to the IP fabric.
- **Border Spine**—A device that typically has two roles—a network underlay device and a border device that provides a connection to one or more external devices. The external devices, for example, multicast gateways or data center gateways, provide additional functionality to the IP fabric.

- **Bridged Overlay**—An Ethernet-based overlay service designed for data center environments that do not require routing within an EVPN/VXLAN fabric. IP routing can provided externally to the fabric as needed.
- **BUM**—Broadcast, Unknown Unicast, and Multicast. The BUM acronym collectively identifies the three traffic types.
- **Centrally-Routed Bridging Overlay**—A form of IRB overlay that provides routing at a central gateway and bridging at the edge of the overlay network. In an IRB overlay, a routed overlay and one or more bridged overlays connect at one or more locations through the use of IRB interfaces.
- **Clos Network**—A multistage network topology first developed by Charles Clos for telephone networks that provides multiple paths to a destination at each stage of the topology. Non-blocking networks are possible in a Clos-based topology.
- **Collapsed Spine fabric**—An EVPN fabric design in which the overlay functions are collapsed onto the spine layer rather than distributed between spine and leaf devices. This type of fabric has no leaf layer in the EVPN core; the spine devices connect directly to the access layer.
- **Contrail Command**—Contrail Enterprise Multicloud user interface. Provides a consolidated, easy-to-use software designed to automate the creation and management of data center networks.
- **Contrail Enterprise Multicloud**—A suite of products and software that combines Contrail Command as a single point of management for private and public clouds, QFX Series switches running Junos OS as an infrastructure for data center networking, and Contrail Insights (formerly known as AppFormix) for telemetry and network visualization.
- **DCI**—Data Center Interconnect. The technology used to interconnect separate data centers.
- **Default instance**—A global instance in a Juniper Networks device that hosts the primary routing table such as **inet.0** (default routing instance) and the primary MAC address table (default switching instance).
- **DHCP relay**—A function that allows a DHCP server and client to exchange DHCP messages over the network when they are not in the same Ethernet broadcast domain. DHCP relay is typically implemented at a default gateway.
- **EBGP**—External BGP. A routing protocol used to exchange routing information between autonomous networks. It has also been used more recently in place of a traditional Interior Gateway Protocols, such as IS-IS and OSPF, for routing within an IP fabric.
- **Edge-Routed Bridging Overlay**—A form of IRB overlay that provides routing and bridging at the edge of the overlay network.
- **End System**—An endpoint device that connects into the data center. An end system can be a wide range of equipment but is often a server, a router, or another networking device in the data center.

- **ESI**—Ethernet segment identifier. An ESI is a 10-octet integer that identifies a unique Ethernet segment in EVPN. In this blueprint architecture, LAGs with member links on different access devices are assigned a unique ESI to enable Ethernet multihoming.
- **Ethernet-connected Multihoming**—An Ethernet-connected end system that connects to the network using Ethernet access interfaces on two or more devices.
- **EVPN**—Ethernet Virtual Private Network. A VPN technology that supports bridged, routed, and hybrid network overlay services. EVPN is defined in RFC 7432 with extensions defined in a number of IETF draft standards.
- **EVPN Type 2 Route**—Advertises MAC addresses and the associated IP addresses from end systems to devices participating in EVPN.
- **GBP**—Group based policy. GBP uses firewall filter rules to provide micro and macro segmentation (security policies) for EVPN-VXLAN by using the reserved fields in the VXLAN header for match conditions.
- **IBGP**—Internal BGP. In this blueprint architecture, IBGP with Multiprotocol BGP (MP-IBGP) is used for EVPN signalling between the devices in the overlay.
- **IP Fabric**—An all-IP fabric network infrastructure that provides multiple symmetric paths between all devices in the fabric. We support an IP Fabric with IPv4 (an *IPv4 Fabric*) with all architectures described in this guide, and an IP Fabric with IPv6 (an *IPv6 Fabric*) with some architectures and on some platforms.
- **IP-connected Multihoming**—An IP-connected end system that connects to the network using IP access interfaces on two or more devices.
- **IRB**—Integrated Routing and Bridging. A technique that enables routing between VLANs and allows traffic to be routed or bridged based on whether the destination is outside or inside of a bridging domain. To activate IRB, you associate a logical interface (IRB interface) with a VLAN and configure the IRB interface with an IP address for the VLAN subnet.
- **Leaf Device**—An access level network device in an IP fabric topology. End systems connect to the leaf devices in this blueprint architecture.
- **MAC-VRF instance**—A routing instance type that enables you to configure multiple customer-specific EVPN instances instead of only one instance (the default instance). MAC-VRF instances use a consistent configuration style across platforms. Different MAC-VRF instances can support different Ethernet service types (VLAN-aware and VLAN-based services) on the same device in a data center.
- **Multiservice Cloud Data Center Network**—A data center network that optimizes the use of available compute, storage, and network access interfaces by allowing them to be shared flexibly across diverse applications, tenants, and use cases.

- **NDP**—Neighbor Discovery Protocol. An IPv6 protocol defined in RFC 4861 that combines the functionality of ARP and ICMP, and adds other enhanced capabilities.
- **NVE**—As defined in RFC 8365, *A Network Virtualization Overlay Solution Using Ethernet VPN (EVPN)*, a network virtualization edge is a device that terminates a VXLAN tunnel in a network virtualization overlay. For example, in a centrally routed bridging overlay, we consider spine and leaf devices to be NVE devices.
- **NVO**—A network virtualization overlay is a fabric in which we use EVPN as a control plane and VXLAN as a data plane.
- **Routed Overlay**—An IP-based overlay service where no Ethernet bridging is required. Also referred to as an IP VPN. In this blueprint architecture, the routed overlay is based on EVPN Type 5 routes and their associated procedures, and supported by VXLAN tunneling.
- **Spine Device**—A centrally-located device in an IP fabric topology that has a connection to each leaf device.
- **Storm Control**—Feature that prevents BUM traffic storms by monitoring BUM traffic levels and taking a specified action to limit BUM traffic forwarding when a specified traffic level is exceeded.
- **Symmetric IRB Routing**—An intersubnet forwarding model in EVPN-VXLAN fabrics where the VXLAN tunnel endpoint (VTEPs) symmetrically route the traffic for a tenant virtual routing and forwarding (VRF) instance using the same VXLAN network identifier (VNI) on both sides of the VXLAN tunnel. We support EVPN Type 5 routing using the symmetric IRB model by default. Type 2 routing is asymmetric by default; you can configure a device to use the symmetric IRB model for EVPN Type 2 routing in a particular VRF.
- **Underlay Network**—A network that provides basic network connectivity between devices.In this blueprint architecture, the underlay network is an IP Fabric that provides the basic IP connectivity, usually with IPv4. We also support an IP Fabric with an IPv6 underlay with some architecture designs on certain platforms.
- **VLAN trunking**—The ability for one interface to support multiple VLANs.
- **VNI**—VXLAN Network Identifier. Uniquely identifies a VXLAN virtual network. A VNI encoded in a VXLAN header can support 16 million virtual networks.
- **VTEP**—VXLAN Tunnel Endpoint. A loopback or virtual interface where traffic enters and exits a VXLAN tunnel. Tenant traffic is encapsulated into VXLAN packets at a source VTEP, and de-encapsulated when the traffic leaves the VXLAN tunnel at a remote VTEP.
- **VXLAN**—Virtual Extensible LAN. Network virtualization tunneling protocol defined in RFC 7348 used to build virtual networks over an IP-routed infrastructure. VXLAN is used to tunnel tenant traffic over the IP fabric underlay from a source endpoint at an ingress device to a destination endpoint at the

egress device. These tunnels are established dynamically by EVPN. Each VTEP device advertises its loopback address in the underlay network for VXLAN tunnel reachability between VTEP devices.

- **VXLAN Stitching**—A Data Center Interconnect (DCI) feature that supports Layer 2 interconnection EVPN-VXLAN fabric on a per-VXLAN VNI basis.

RELATED DOCUMENTATION

Contrail Enterprise Multicloud Components

# Introducing the Data Center Fabric Blueprint Architecture

**IN THIS SECTION**

## Data Center Fabric Blueprint Architecture Introduction

This section provides an introduction to the Data Center Fabric Blueprint Architecture.

It includes the following sections.

### The Next Act for Data Center Networks

For truly cloud-native workloads that have no dependency on Ethernet broadcast, multicast, segmentation, multitenancy, or workload mobility, the best solution is typically a simple IP fabric network. When a unique workload instance requires mobility, the current host system can advertise the unique IP address of the workload. This can be performed with EBGP route exchange between the host system and the ToR. However, support for BUM and multitenancy require more advanced network functions. This is where overlays are added.

Over its evolution the data center network was a function of the demands and expectations placed on it. As the nature of workloads changed, the network had to adapt. Each solution simplified a set of problems by trading one form of complexity and cost for another. The cloud network is no different. In the end, bits must be moved from point A to point B reliably, securely, and at the desired throughput. Where operators need a single network to serve more than one purpose (the multiservice cloud network), they can add network-layer segmentation and other functions to share the infrastructure across diverse groups of endpoints and tenants. Operational simplicity is achieved with a centralized controller that implements an intent model that is consistent with the cloud scale functions of the network layer. Technical simplicity is achieved using a reduced set of building blocks that are based on open standards and homogeneous across the end-to-end network.

This guide introduces a building block approach to creating multiservice cloud networks on the foundation of a modern IP fabric. The Juniper Networks solutions team will systematically review the set of building blocks required for an agile network, focus on specific, state-of-the-art, open standards-based technology that enables each function, and add new functionality to the guide as it becomes available in future software releases.

All the building blocks are fully synergistic and you can combine any of the building blocks with any other to satisfy a diverse set of use cases simultaneously — this is the hallmark of the cloud. You should consider how you can leverage the building blocks in this guide to achieve the use cases that are relevant to your network.

## Building Blocks

The guide organizes the technologies used to build multiservice cloud network architectures into modular building blocks. Each building block includes features that either must be implemented together to build the network, are often implemented together because they provide complementary functionality, or are presented together to provide choices for particular technologies.

This blueprint architecture includes required building blocks and optional building blocks. The optional building blocks can be added or removed to support the needs of a specific multiservice data center fabric network.

This guide walks you through the design and technology choices associated with each building block, and provides information designed to help you choose the building blocks that best meet the needs for your multiservice data center fabric network. The guide also provides the implementation procedures for each building block.

The currently-supported building blocks include:

- IP Fabric Underlay Network
- Network Virtualization Overlays

    - Centrally-Routed Bridging Overlay
    - Edge-Routed Bridging Overlay
    - Routed Overlay
- Multihoming
    - Multihoming of Ethernet-connected End Systems
    - Multihoming of IP-connected End Systems
- Data Center Interconnect (DCI)
- Service Chaining
- Multicast
- Ingress Virtual Machine Traffic Optimization
- DHCP Relay
- Proxy ARP
- Layer 2 Port Security

Additional building blocks will be added to this guide as support for the technology becomes available and is validated by the Juniper Networks testing team.

Each building block is discussed in more detail in "Data Center Fabric Blueprint Architecture Components" on page 9.

For information about the hardware and software that serve as a foundation to your building blocks, see "Data Center EVPN-VXLAN Fabric Reference Designs—Supported Hardware Summary" on page 51.

## RELATED DOCUMENTATION

Data Center Fabric Reference Design Overview and Validated Topology | **49**

Data Center Fabric Blueprint Architecture Components | **9**

# Data Center Fabric Blueprint Architecture Components

IN THIS SECTION

This section gives an overview of the building blocks used in this blueprint architecture. The implementation of each building block technology is explored in more detail later sections.

For information about the hardware and software that serve as a foundation to your building blocks, see the "Data Center EVPN-VXLAN Fabric Reference Designs—Supported Hardware Summary" on page 51.

The building blocks include:

## IP Fabric Underlay Network

The modern IP fabric underlay network building block provides IP connectivity across a Clos-based topology. Juniper Networks supports the following IP fabric underlay models:

- A 3-stage IP fabric, which is made up of a tier of spine devices and a tier of leaf devices. See Figure 1 on page 10.
- A 5-stage IP fabric, which typically starts as a single 3-stage IP fabric that grows into two 3-stage IP fabrics. These fabrics are segmented into separate points of delivery (PODs) within a data center. For this use case, we support the addition of a tier of super spine devices that enable communication between the spine and leaf devices in the two PODs. See Figure 2 on page 11.
- A collapsed spine IP fabric model, in which leaf layer functions are collapsed onto the spine devices. This type of fabric can be set up and operate similarly to either a 3-stage or 5-stage IP fabric except without a separate tier of leaf devices. You might use a collapsed spine fabric if you are moving incrementally to an EVPN spine-and-leaf model, or you have access devices or top-of-rack (TOR) devices that can't be used in a leaf layer because they don't support EVPN-VXLAN.

**Figure 1: Three-Stage IP Fabric Underlay**

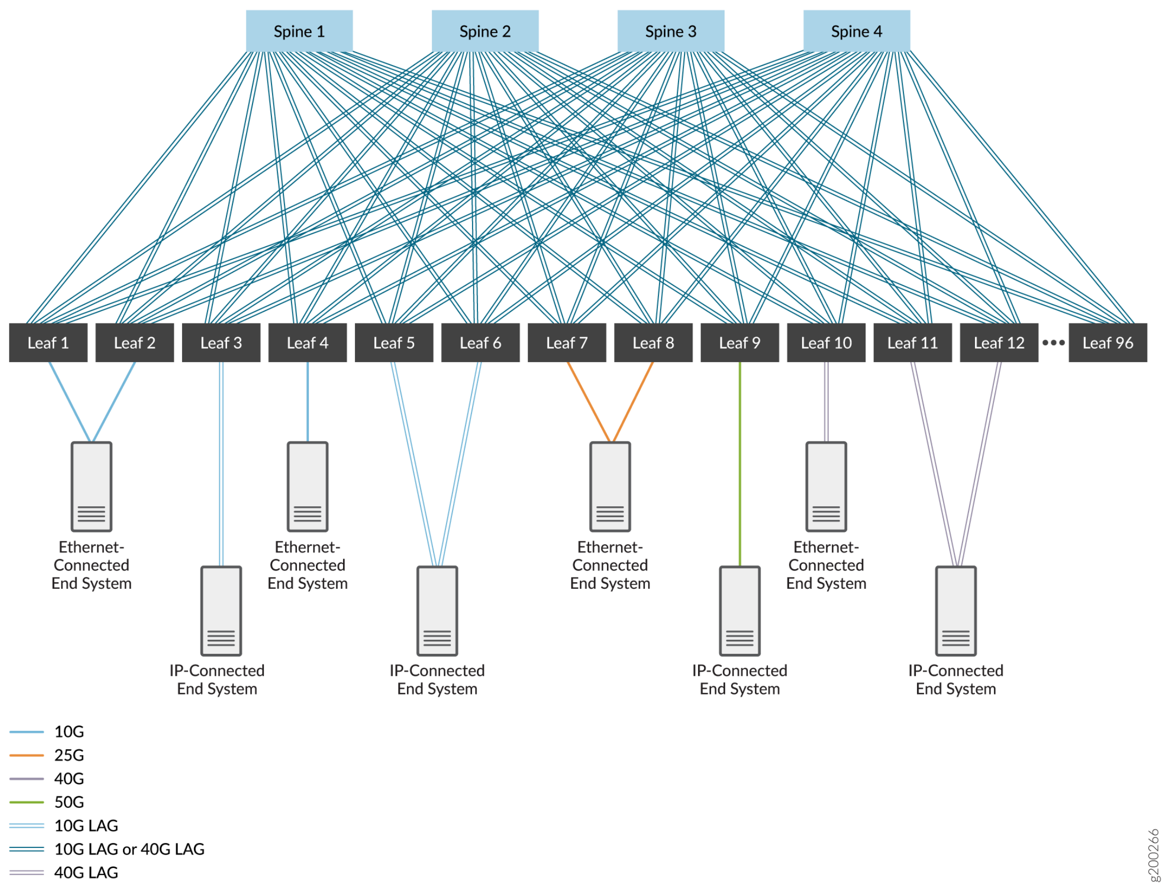
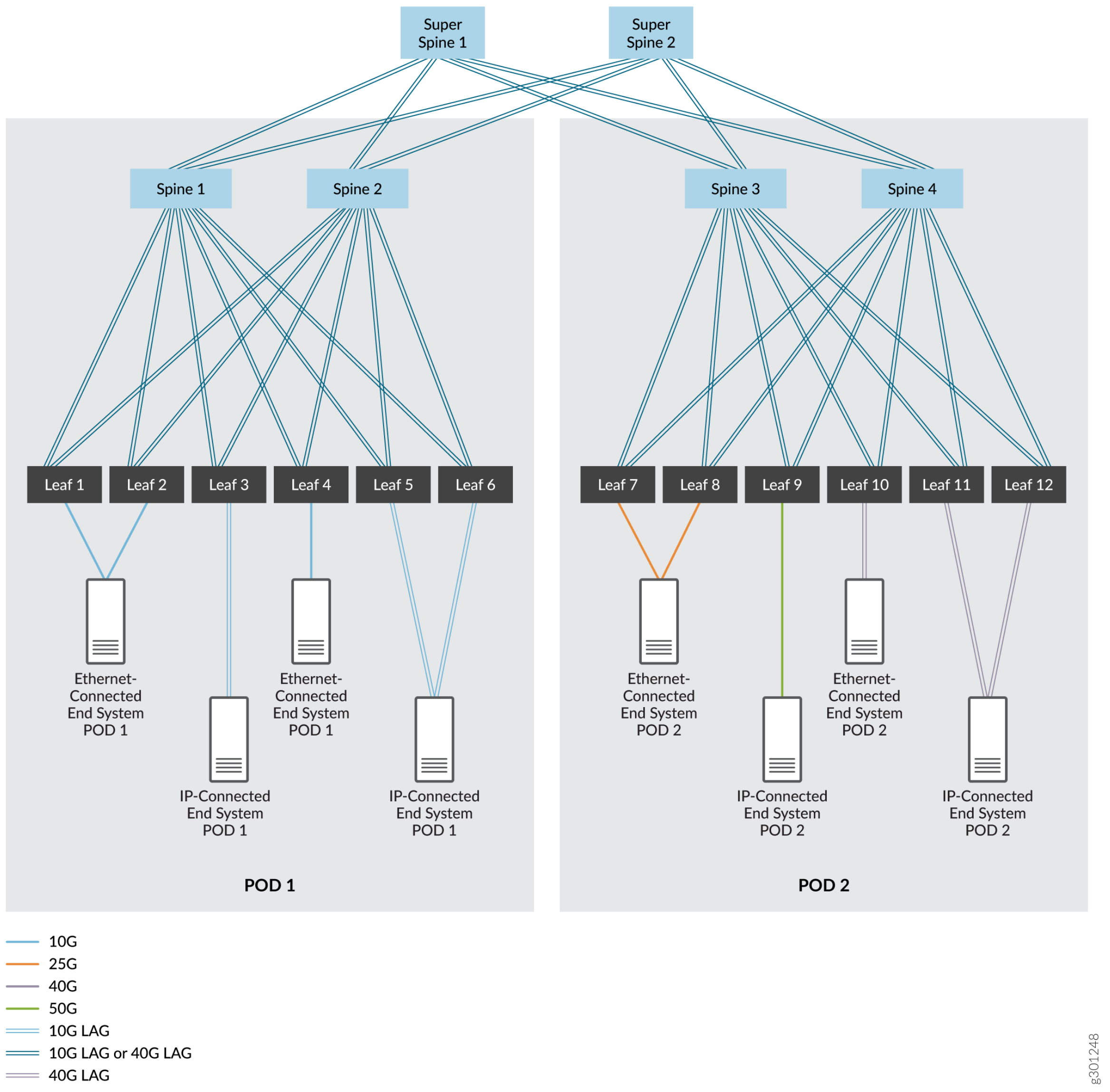


**Figure 2: Five-Stage IP Fabric Underlay**


Super Spine 1
Super Spine 2
Spine 1
Spine 2
Spine 3
Spine 4
Leaf 1
Leaf 2
Leaf 3
Leaf 4
Leaf 5
Leaf 6
Leaf 7
Leaf 8
Leaf 9
Leaf 10
Leaf 11
Leaf 12
Ethernet-Connected End System POD 1
IP-Connected End System POD 1
Ethernet-Connected End System POD 1
IP-Connected End System POD 1
Ethernet-Connected End System POD 2
IP-Connected End System POD 2
Ethernet-Connected End System POD 2
IP-Connected End System POD 2
POD 1
POD 2
10G
25G
40G
50G
10G LAG
10G LAG or 40G LAG
40G LAG
g301248

In these figures, the devices are interconnected using high-speed interfaces that are either single links or aggregated Ethernet interfaces. The aggregated Ethernet interfaces are optional—a single link between devices is typically used— but can be deployed to increase bandwidth and provide link level redundancy. We cover both options.

We chose EBGP as the routing protocol in the underlay network for its dependability and scalability. Each device is assigned its own autonomous system with a unique autonomous system number to support EBGP. You can use other routing protocols in the underlay network; the usage of those protocols is beyond the scope of this document.

The reference architecture designs described in this guide are based on an IP Fabric that uses EBGP for the underlay connectivity and IBGP for overlay peering (see "IBGP for Overlays" on page 14). You can alternatively configure the overlay peering using EBGP.

Starting in Junos OS Releases 21.2R2 and 21.4R1, we also support configuring an *IPv6 Fabric*. The IPv6 Fabric design in this guide uses EBGP for both underlay connectivity and overlay peering (see "EBGP for Overlays with IPv6 Underlays" on page 15).

**NOTE**: The IP Fabric can use IPv4 or IPv6 as follows:

- An IPv4 Fabric uses IPv4 interface addressing, and IPv4 underlay and overlay BGP sessions for end-to-end workload communication.
- An IPv6 Fabric uses IPv6 interface addressing, and IPv6 underlay and overlay BGP sessions for end-to-end workload communication.
- We don’t support an IP Fabric that mixes IPv4 and IPv6.

  However, both IPv4 Fabrics and IPv6 Fabrics support dual-stack workloads—the workloads can be either IPv4 or IPv6, or both IPv4 and IPv6.

Micro Bidirectional Forwarding Detection (BFD)—the ability to run BFD on individual links in an aggregated Ethernet interface—can also be enabled in this building block to quickly detect link failures on any member links in aggregated Ethernet bundles that connect devices.

For more information, see these other sections in this guide:

- Configuring spine and leaf devices in 3-stage and 5-stage IP fabric underlays: "IP Fabric Underlay Network Design and Implementation" on page 80.
- Implementing the additional tier of super spine devices in a 5-stage IP fabric underlay: "Five-Stage IP Fabric Design and Implementation" on page 255.
- Configuring an IPv6 underlay and supporting EBGP IPv6 overlay: "IPv6 Fabric Underlay and Overlay Network Design and Implementation with EBGP" on page 103.
- Setting up the underlay in a collapsed spine fabric model: "Collapsed Spine Fabric Design and Implementation" on page 285.

## IPv4 and IPv6 Workload Support

Because many networks implement a dual stack environment for workloads that includes IPv4 and IPv6 protocols, this blueprint provides support for both protocols. Steps to configure the fabric to support

IPv4 and IPv6 workloads are interwoven throughout this guide to allow you to pick one or both of these protocols.

**NOTE**: The IP protocol you use for workload traffic is independent of the IP protocol version (IPv4 or IPv6) that you configure for the IP Fabric underlay and overlay. (See "IP Fabric Underlay Network" on page 9.) An IPv4 Fabric or an IPv6 Fabric infrastructure can support both IPv4 and IPv6 workloads.

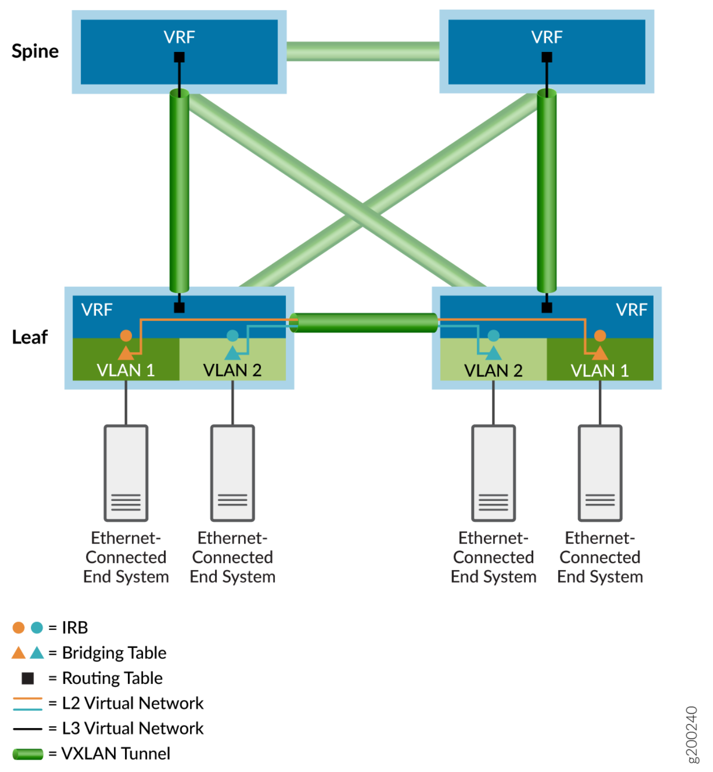
## Network Virtualization Overlays

A *network virtualization overlay* is a virtual network that is transported over an IP underlay network. This building block enables multitenancy in a network, allowing you to share a single physical network across multiple tenants, while keeping each tenant's network traffic isolated from the other tenants.

A tenant is a user community (such as a business unit, department, workgroup, or application) that contains groups of endpoints. Groups may communicate with other groups in the same tenancy, and tenants may communicate with other tenants if permitted by network policies. A group is typically expressed as a subnet (VLAN) that can communicate with other devices in the same subnet, and reach external groups and endpoints by way of a virtual routing and forwarding (VRF) instance.

As seen in the overlay example shown in Figure 3 on page 14, Ethernet bridging tables (represented by triangles) handle tenant bridged frames and IP routing tables (represented by squares) process routed packets. Inter-VLAN routing happens at the integrated routing and bridging (IRB) interfaces (represented by circles). Ethernet and IP tables are directed into virtual networks (represented by colored lines). To reach end systems attached to other VXLAN Tunnel Endpoint (VTEP) devices, tenant packets are encapsulated and sent over an EVPN-signalled VXLAN tunnel (represented by green tunnel icons) to the associated remote VTEP devices. Tunneled packets are de-encapsulated at the remote VTEP devices and forwarded to the remote end systems by way of the respective bridging or routing tables of the egress VTEP device.

**Figure 3: VXLAN Tunnels—Ethernet Bridging, IP Routing, and IRB**



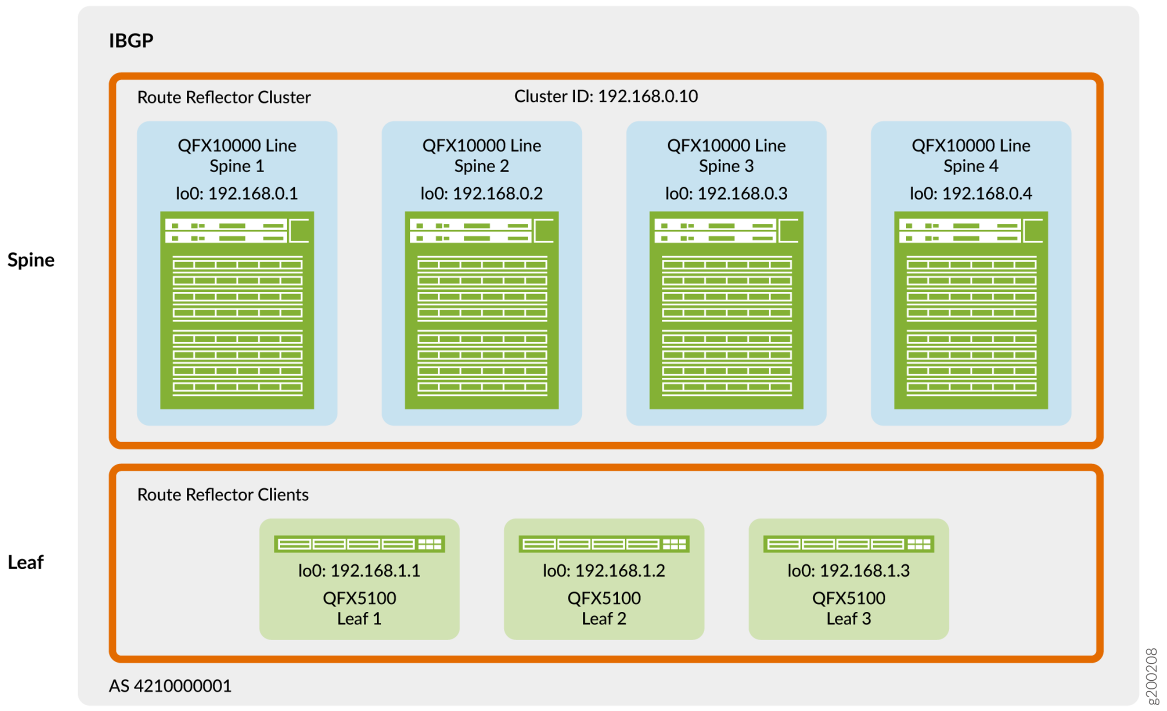
The next sections provide more details about overlay networks.

## IBGP for Overlays

Internal BGP (IBGP) is a routing protocol that exchanges reachability information across an IP network. When IBGP is combined with Multiprotocol BGP (MP-IBGP), it provides the foundation for EVPN to exchange reachability information between VTEP devices. This capability is required to establish inter-VTEP VXLAN tunnels and use them for overlay connectivity services.

Figure 4 on page 15 shows that the spine and leaf devices use their loopback addresses for peering in a single autonomous system. In this design, the spine devices act as a route reflector cluster and the leaf devices are route reflector clients. A route reflector satisfies the IBGP requirement for a full mesh without the need to peer all the VTEP devices directly with one another. As a result, the leaf devices peer only with the spine devices and the spine devices peer with both spine devices and leaf devices. Because the spine devices are connected to all the leaf devices, the spine devices can relay IBGP information between the indirectly peered leaf device neighbors.

**Figure 4: IBGP for Overlays**



You can place route reflectors almost anywhere in the network. However, you must consider the following:

- Does the selected device have enough memory and processing power to handle the additional workload required by a route reflector?
- Is the selected device equidistant and reachable from all EVPN speakers?
- Does the selected device have the proper software capabilities?

In this design, the route reflector cluster is placed at the spine layer. The QFX switches that you can use as a spine in this reference design have ample processing speed to handle route reflector client traffic in the network virtualization overlay.

For details about implementing IBGP in an overlay, see "Configure IBGP for the Overlay" on page 96.

## EBGP for Overlays with IPv6 Underlays

The original reference architecture use cases in this guide illustrate an IPv4 EBGP underlay design with IPv4 IBGP overlay device connectivity. See "IP Fabric Underlay Network" on page 9 and "IBGP for Overlays" on page 14. However, as network virtualization edge (NVE) devices start adopting IPv6 VTEPs to take advantage of the extended addressing range and capabilities of IPv6, we have expanded IP Fabric support to encompass IPv6.

Starting in Junos OS Release 21.2R2-S1, on supporting platforms you can alternatively use an IPv6 Fabric infrastructure with some reference architecture overlay designs. The IPv6 Fabric design comprises IPv6 interface addressing, an IPv6 EBGP underlay and an IPv6 *EBGP* overlay for workload connectivity. With an IPv6 Fabric, the NVE devices encapsulate the VXLAN header with an IPv6 outer header and tunnel the packets across the fabric end to end using IPv6 next hops. The workload can be IPv4 or IPv6.

Most of the elements you configure in the supported reference architecture overlay designs are independent of whether the underlay and overlay infrastructure uses IPv4 or IPv6. The corresponding configuration procedures for each of the supported overlay designs call out any configuration differences if the underlay and overlay uses the IPv6 Fabric design.

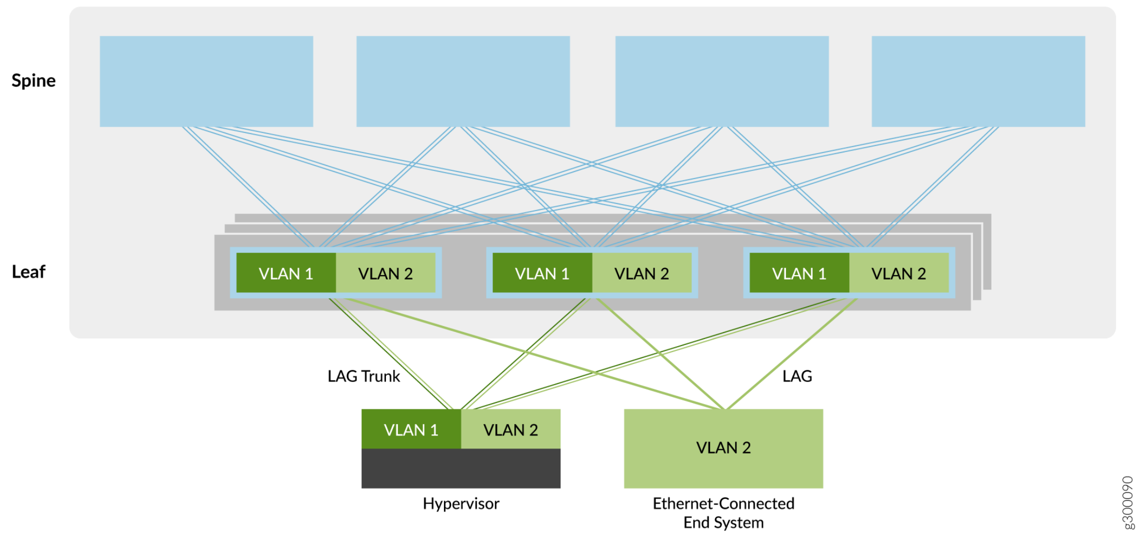
For more details, see the following references in this guide and other resources:

- Configuring an IPv6 Fabric using EBGP for underlay connectivity and overlay peering: "IPv6 Fabric Underlay and Overlay Network Design and Implementation with EBGP" on page 103.
- Starting releases in which different platforms support an IPv6 Fabric design when serving in particular roles in the fabric: "Data Center EVPN-VXLAN Fabric Reference Designs—Supported Hardware Summary" on page 51.
- Overview of IPv6 underlay and overlay peering support in EVPN-VXLAN fabrics on Juniper Networks devices: EVPN-VXLAN with an IPv6 Underlay.

## Bridged Overlay

The first overlay service type described in this guide is a *bridged overlay*, as shown in Figure 5 on page 17.

**Figure 5: Bridged Overlay**



In this overlay model, Ethernet VLANs are extended between leaf devices across VXLAN tunnels. These leaf-to-leaf VXLAN tunnels support data center networks that require Ethernet connectivity between leaf devices but do not need routing between the VLANs. As a result, the spine devices provide only basic underlay and overlay connectivity for the leaf devices, and do not perform routing or gateway services seen with other overlay methods.

Leaf devices originate VTEPs to connect to the other leaf devices. The tunnels enable the leaf devices to send VLAN traffic to other leaf devices and Ethernet-connected end systems in the data center. The simplicity of this overlay service makes it attractive for operators who need an easy way to introduce EVPN/VXLAN into their existing Ethernet-based data center.

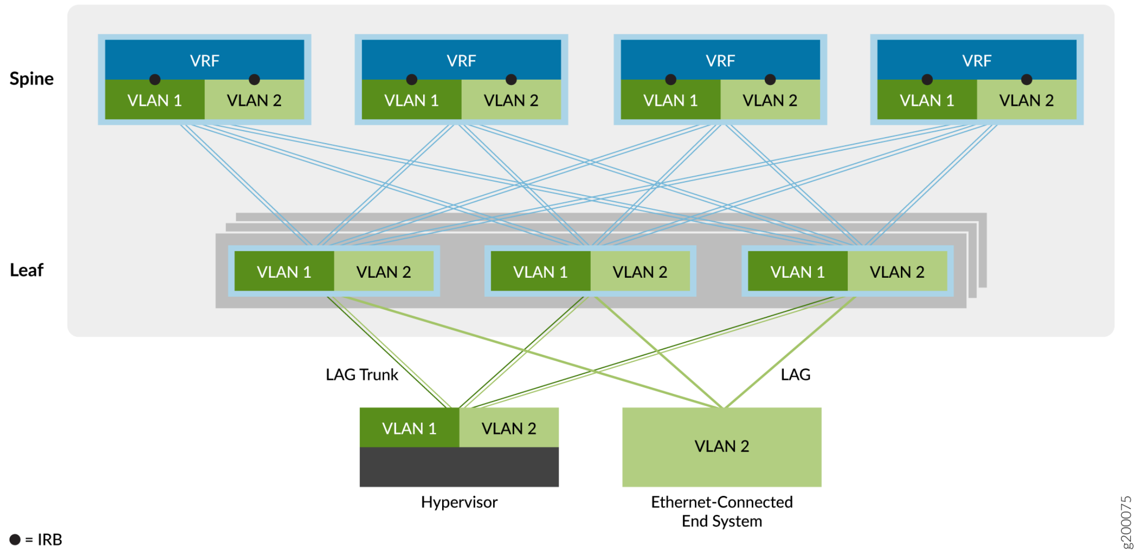
**NOTE**: You can add routing to a bridged overlay by implementing an MX Series router or SRX Series security device external to the EVPN/VXLAN fabric. Otherwise, you can select one of the other overlay types that incorporate routing (such as an "edge-routed bridging overlay" on page 206, a "centrally-routed bridging overlay" on page 142, or a "routed overlay" on page 237).

For information on implementing a bridged overlay, see "Bridged Overlay Design and Implementation" on page 112.

## Centrally Routed Bridging Overlay

The second overlay service type is the *centrally routed bridging (CRB) overlay*, as shown in Figure 6 on page 18.

**Figure 6: Centrally Routed Bridging Overlay**



In a CRB overlay, routing occurs at a central gateway of the data center network (the spine layer in this example) rather than at the VTEP device where the end systems are connected (the leaf layer in this example).

You can use this overlay model when you need routed traffic to go through a centralized gateway or when your edge VTEP devices lack the required routing capabilities.

As shown above, traffic that originates at the Ethernet-connected end systems is forwarded to the leaf VTEP devices over a trunk (multiple VLANs) or an access port (single VLAN). The VTEP device forwards the traffic to local end systems or to an end system at a remote VTEP device. An integrated routing and bridging (IRB) interface at each spine device helps route traffic between the Ethernet virtual networks.

The VLAN-aware bridging overlay service model enables you to easily aggregate a collection of VLANs into the same overlay virtual network. The Juniper Networks EVPN design supports three VLAN-aware Ethernet service model configurations in the data center, as follows:

- **Default instance VLAN-aware**—With this option, you implement a single, default switching instance that supports a total of 4094 VLANs. All leaf platforms included in this design ("Data Center EVPN-VXLAN Fabric Reference Designs—Supported Hardware Summary" on page 51) support the default instance style of VLAN-aware overlay.

  To configure this service model, see Configuring a VLAN-Aware Centrally-Routed Bridging Overlay in the Default Instance.

- **Virtual switch VLAN-aware**—With this option, multiple virtual switch instances support up to 4094 VLANs per instance. This Ethernet service model is ideal for overlay networks that require scalability beyond a single default instance. Support for this option is available currently on the QFX10000 line of switches.

To implement this scalable service model, see "Configuring a VLAN-Aware CRB Overlay with Virtual Switches or MAC-VRF Instances" on page 175.

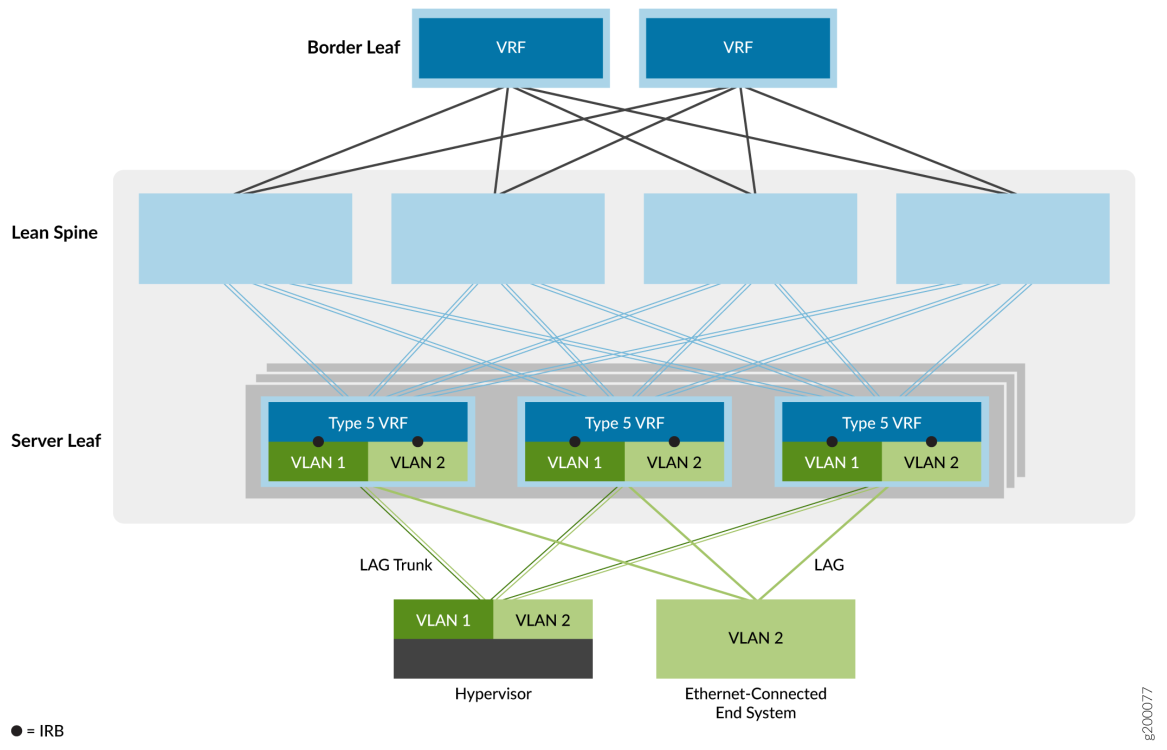
- **MAC-VRF instance VLAN-aware**—With this option, multiple MAC-VRF instances support up to 4094 VLANs per instance. This Ethernet service model is ideal for overlay networks that require scalability beyond a single default instance, and where you want more options to ensure VLAN isolation or interconnection among different tenants in the same fabric. Support for this option is available on the platforms that support MAC-VRF instances (see Feature Explorer: MAC VRF with EVPN-VXLAN).

  To implement this scalable service model, see "Configuring a VLAN-Aware CRB Overlay with Virtual Switches or MAC-VRF Instances" on page 175.

## Edge-Routed Bridging Overlay

The third overlay service option is the *edge-routed bridging (ERB) overlay*, as shown in Figure 7 on page 20.

**Figure 7: Edge-Routed Bridging Overlay**



In this Ethernet service model, the IRB interfaces are moved to leaf device VTEPs at the edge of the overlay network to bring IP routing closer to the end systems. Because of the special ASIC capabilities required to support bridging, routing, and EVPN/VXLAN in one device, ERB overlays are only possible on certain switches. For a list of switches that we support as leaf devices in an ERB overlay, see "Data Center EVPN-VXLAN Fabric Reference Designs—Supported Hardware Summary" on page 51.

This model allows for a simpler overall network. The spine devices are configured to handle only IP traffic, which removes the need to extend the bridging overlays to the spine devices.

This option also enables faster server-to-server, intra-data center traffic (also known as east-west traffic) where the end systems are connected to the same leaf device VTEP. As a result, routing happens much closer to the end systems than with CRB overlays.

**NOTE**: When you configure IRB interfaces that are included in EVPN Type-5 routing instances on QFX5110 or QFX5120 switches that function as leaf devices, the device automatically enables symmetric inter-IRB unicast routing for EVPN Type 5 routes.

For information on implementing the ERB overlay, see "Edge-Routed Bridging Overlay Design and Implementation" on page 206.

### Collapsed Spine Overlay

The overlay network in a collapsed spine architecture is similar to an ERB overlay. In a collapsed spine architecture, the leaf device functions are collapsed onto the spine devices. Because there is no leaf layer, you configure the VTEPS and IRB interfaces on the spine devices, which are at the edge of the overlay network like the leaf devices in an ERB model. The spine devices can also perform border gateway functions to route north-south traffic, or extend Layer 2 traffic across data center locations.

For a list of switches that we support with a collapsed spine architecture, see "Data Center EVPN-VXLAN Fabric Reference Designs—Supported Hardware Summary" on page 51.

### Comparison of Bridged, CRB, and ERB Overlays

To help you decide which overlay type is best suited for your EVPN environment, see Table 1 on page 21.

**NOTE**: We support mixing bridged overlay, CRB overlay, and ERB overlay configurations on the same device at the same time on devices that support these overlay types. You don't need to configure the device with separate logical systems for the device to operate in different types of overlays in parallel.

**Table 1: Comparison of Bridged, CRB, and ERB Overlays**

| Comparison Points | ERB Overlay | CRB Overlay | Bridged Overlay |
|---|---|---|---|
| Fully distributed tenant inter-subnet routing | ✓ | | |
| Minimal impact of IP gateway failure | ✓ | | |
| Dynamic routing to third-party nodes at leaf level | ✓ | | |
| Optimized for high volume of east-west traffic | ✓ | | |

**Table 1: Comparison of Bridged, CRB, and ERB Overlays** ***(Continued)***

| Comparison Points | ERB Overlay | CRB Overlay | Bridged Overlay |
|---|---|---|---|
| Better integration with raw IP fabrics | ✓ | | |
| IP VRF virtualization closer to the server | ✓ | | |
| Contrail vRouter multihoming required | ✓ | | |
| Easier EVPN interoperability with different vendors | ✓ | | |
| Symmetric inter-subnet routing | ✓ | ✓ | |
| VLAN ID overlapping per rack | ✓ | ✓ | ✓ |
| Simpler manual configuration and troubleshooting | | ✓ | ✓ |
| Service provider- and Enterprise-style interfaces | | ✓ | ✓ |
| Legacy leaf switch support (QFX5100) | | ✓ | ✓ |
| Centralized virtual machine traffic optimization (VMTO) control | | ✓ | |

**Table 1: Comparison of Bridged, CRB, and ERB Overlays *(Continued)***

| Comparison Points | ERB Overlay | CRB Overlay | Bridged Overlay |
|---|---|---|---|
| IP tenant subnet gateway on the firewall cluster | | | ✓ |

## IRB Addressing Models in Bridging Overlays

The configuration of IRB interfaces in CRB and ERB overlays requires an understanding of the models for the default gateway IP and MAC address configuration of IRB interfaces as follows:

- **Unique IRB IP Address**—In this model, a unique IP address is configured on each IRB interface in an overlay subnet.

  The benefit of having a unique IP address and MAC address on each IRB interface is the ability to monitor and reach each of the IRB interfaces from within the overlay using its unique IP address. This model also allows you to configure a routing protocol on the IRB interface.

  The downside of this model is that allocating a unique IP address to each IRB interface may consume many IP addresses of a subnet.

- **Unique IRB IP Address with Virtual Gateway IP Address**—This model adds a virtual gateway IP address to the previous model, and we recommend it for centrally routed bridged overlays. It is similar to VRRP, but without the in-band data plane signaling between the gateway IRB interfaces. The virtual gateway should be the same for all default gateway IRB interfaces in the overlay subnet and is active on all gateway IRB interfaces where it is configured. You should also configure a common IPv4 MAC address for the virtual gateway, which becomes the source MAC address on data packets forwarded over the IRB interface.

  In addition to the benefits of the previous model, the virtual gateway simplifies default gateway configuration on end systems. The downside of this model is the same as the previous model.

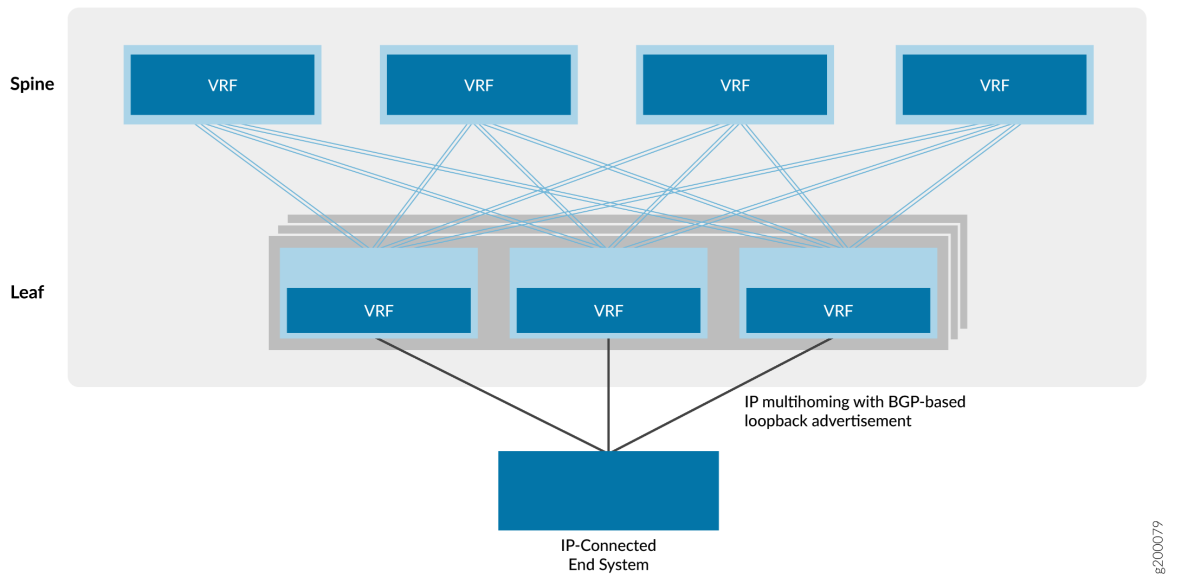
- **IRB with Anycast IP Address and MAC Address**—In this model, all default gateway IRB interfaces in an overlay subnet are configured with the same IP and MAC address. We recommend this model for ERB overlays.

  A benefit of this model is that only a single IP address is required per subnet for default gateway IRB interface addressing, which simplifies default gateway configuration on end systems.

## Routed Overlay using EVPN Type 5 Routes

The final overlay option is a *routed overlay*, as shown in Figure 8 on page 24.

**Figure 8: Routed Overlay**



This option is an IP-routed virtual network service. Unlike an MPLS-based IP VPN, the virtual network in this model is based on EVPN/VXLAN.

Cloud providers prefer this virtual network option because most modern applications are optimized for IP. Because all communication between devices happens at the IP layer, there is no need to use any Ethernet bridging components, such as VLANs and ESIs, in this routed overlay model.

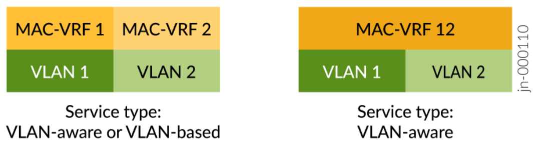
For information on implementing a routed overlay, see "Routed Overlay Design and Implementation" on page 237.

## MAC-VRF Instances for Multitenancy in Network Virtualization Overlays

MAC-VRF routing instances enable you to configure multiple EVPN instances with different Ethernet service types on a device acting as a VTEP in an EVPN-VXLAN fabric. Using MAC-VRF instances, you can manage multiple tenants in the data center with customer-specific VRF tables to isolate or group tenant workloads.

MAC-VRF instances also introduce support for the VLAN-based service type in addition to the prior support for the VLAN-aware service type. See Figure 9 on page 25.

**Figure 9: MAC-VRF Service Types**



- VLAN-based service—You can configure one VLAN and the corresponding VXLAN network identifier (VNI) in the MAC-VRF instance. To provision a new VLAN and VNI, you must configure a new MAC VRF instance with the new VLAN and VNI.

- VLAN-aware service—You can configure one or more VLANs and the corresponding VNIs in the same MAC-VRF instance. To provision a new VLAN and VNI, you can add the new VLAN and VNI configuration to the existing MAC-VRF instance, which saves some configuration steps over using a VLAN-based service.

MAC-VRF instances enable more flexible configuration options at both Layer 2 and Layer 3. For example:

**Figure 10: Flexible Configuration Options at both Layer 2 and Layer 3 with MAC-VRF Instances**

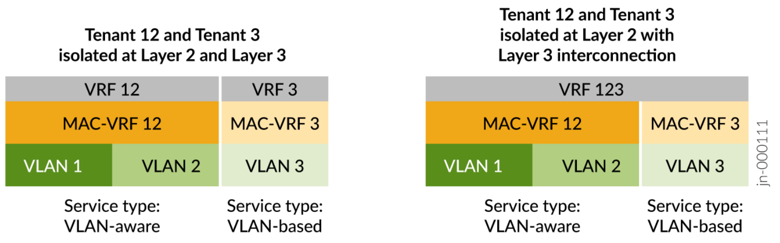


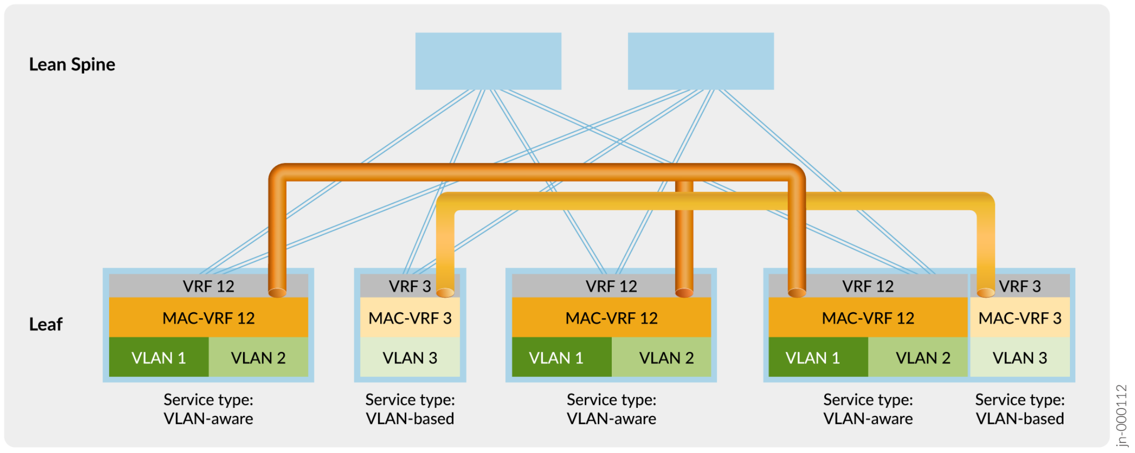
Figure 10 on page 25 shows that with MAC-VRF instances:

- You can configure different service types in different MAC-VRF instances on the same device.

- You have flexible tenant isolation options at Layer 2 (MAC-VRF instances) as well as at Layer 3 (VRF instances). You can configure a VRF instance that corresponds to the VLAN or VLANs in a single MAC-VRF instance. Or you can configure a VRF instance that spans the VLANs in multiple MAC-VRF instances.

Figure 11 on page 26 shows an ERB overlay fabric with a sample MAC-VRF configuration for tenant separation.

**Figure 11: ERB Overlay Fabric with MAC-VRF Instances for Tenant Separation**



In Figure 11 on page 26, the leaf devices establish VXLAN tunnels that maintain isolation at Layer 2 between Tenant 12 (VLAN 1 and VLAN 2) and Tenant 3 (VLAN 3) using MAC-VRF instances MAC-VRF 12 and MAC-VRF 3. The leaf devices also isolate the tenants at Layer 3 using VRF instances VRF 12 and VRF 3.

You can employ other options for sharing VLAN traffic between tenants that are isolated at Layer 2 and Layer 3 by the MAC-VRF and VRF configurations, such as:

- Establish a secure external interconnection between tenant VRFs through a firewall.
- Configure local route leaking between Layer 3 VRFs.

For more information about MAC-VRF instances and using them in an example customer use case, see EVPN-VXLAN DC IP Fabric MAC-VRF L2 Services.

MAC-VRF instances correspond to forwarding instances as follows:

- MAC-VRF instances on switches in the QFX5000 line (including those that run Junos OS or Junos OS Evolved) are all part of the default forwarding instance. On these devices, you can't configure overlapping VLANs in a MAC-VRF instance or across multiple MAC-VRF instances.
- On the QFX10000 line of switches, you can configure multiple forwarding instances, and map a MAC-VRF instance to a particular forwarding instance. You can also map multiple MAC-VRF instances to the same forwarding instance. If you configure each MAC-VRF instance to use a different forwarding instance, you can configure overlapping VLANs across the multiple MAC-VRF instances. You can't configure overlapping VLANs in a single MAC-VRF instance or across MAC-VRF instances that map to the same forwarding instance.

- In the default configuration, switches include a default VLAN with VLAN ID=1 associated with the default forwarding instance. Because VLAN IDs must be unique in a forwarding instance, if you want to configure a VLAN with VLAN ID=1 in a MAC-VRF instance that uses the default forwarding instance, you must reassign the VLAN ID of the default VLAN to a value other than 1. For example:

  ```
  set vlans default vlan-id 4094
  set routing-instances mac-vrf-instance-name vlans vlan-name vlan-id 1
  ```

The reference network virtualization overlay configuration examples in this guide include steps to configure the overlay using MAC-VRF instances. You configure an EVPN routing instance of type `mac-vrf`, and set a route distinguisher and a route target in the instance. You also include the desired interfaces (including a VTEP source interface), VLANs, and VLAN-to-VNI mappings in the instance. See the reference configurations in the following topics:

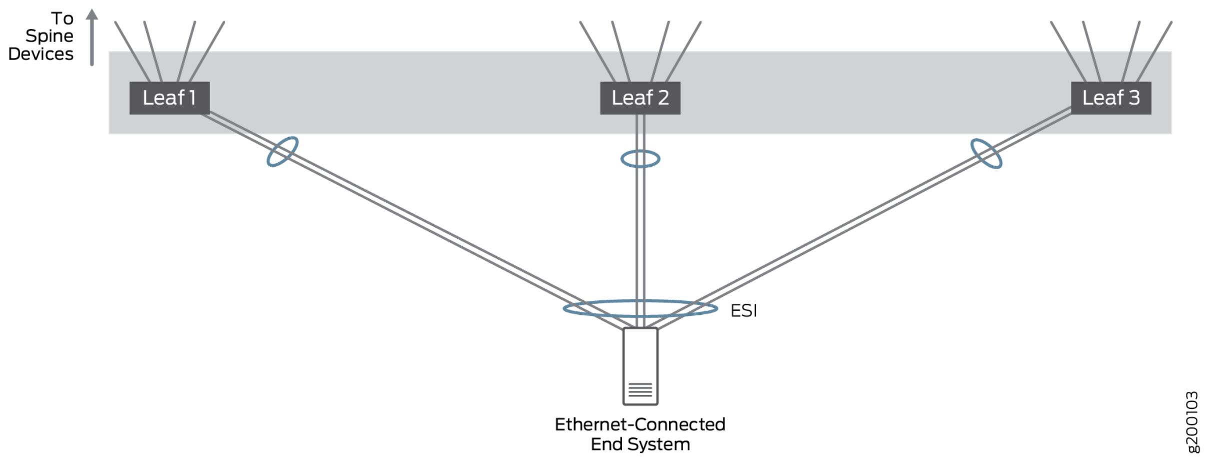
- "Bridged Overlay Design and Implementation" on page 112—You configure MAC-VRF instances on the leaf devices.
- "Centrally-Routed Bridging Overlay Design and Implementation" on page 142—You configure MAC-VRF instances on the spine devices. On the leaf devices, the MAC-VRF configuration is similar to the MAC-VRF leaf configuration in a bridged overlay design.
- "Edge-Routed Bridging Overlay Design and Implementation" on page 206—You configure MAC-VRF instances on the leaf devices.

**NOTE**: A device can have problems with VTEP scaling when the configuration uses multiple MAC-VRF instances. As a result, to avoid this problem, we require that you enable the shared tunnels feature on the QFX5000 line of switches running Junos OS with a MAC-VRF instance configuration. When you configure shared tunnels, the device minimizes the number of next-hop entries to reach remote VTEPs. You globally enable shared VXLAN tunnels on the device using the `shared-tunnels` statement at the `[edit forwarding-options evpn-vxlan]` hierarchy level. This setting requires you to reboot the device.
This statement is optional on the QFX10000 line of switches running Junos OS, which can handle higher VTEP scaling than QFX5000 switches.

On devices running Junos OS Evolved in EVPN-VXLAN fabrics, shared tunnels are enabled by default. Junos OS Evolved supports EVPN-VXLAN only with MAC-VRF configurations.

## Multihoming Support for Ethernet-Connected End Systems

**Figure 12: Ethernet-Connected End System Multihoming**



*Ethernet-connected multihoming* allows Ethernet-connected end systems to connect into the Ethernet overlay network over a single-homed link to one VTEP device or over multiple links multihomed to different VTEP devices. Ethernet traffic is load-balanced across the fabric between VTEPs on leaf devices that connect to the same end system.

We tested setups where an Ethernet-connected end system was connected to a single leaf device or multihomed to 2 or 3 leaf devices to prove traffic is properly handled in multihomed setups with more than two leaf VTEP devices; in practice, an Ethernet-connected end system can be multihomed to a large number of leaf VTEP devices. All links are active and network traffic can be load balanced over all of the multihomed links.

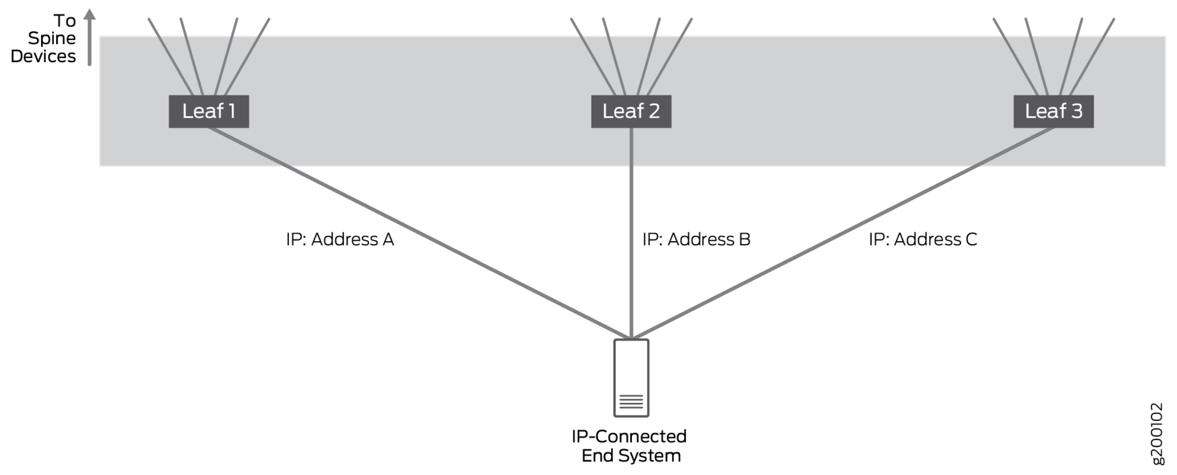
In this architecture, EVPN is used for Ethernet-connected multihoming. EVPN multihomed LAGs are identified by an Ethernet segment identifier (ESI) in the EVPN bridging overlay while LACP is used to improve LAG availability.

VLAN trunking allows one interface to support multiple VLANs. VLAN trunking ensures that virtual machines (VMs) on non-overlay hypervisors can operate in any overlay networking context.

For more information about Ethernet-connected multihoming support, see "Multihoming an Ethernet-Connected End System Design and Implementation" on page 199.

## Multihoming Support for IP-Connected End Systems

**Figure 13: IP-Connected End System Multihoming**



*IP-connected multihoming* endpoint systems to connect to the IP network over multiple IP-based access interfaces on different leaf devices.

We tested setups where an IP–connected end system was connected to a single leaf or multihomed to 2 or 3 leaf devices. The setup validated that traffic is properly handled when multihomed to multiple leaf devices; in practice, an IP-connected end system can be multihomed to a large number of leaf devices.

In multihomed setups, all links are active and network traffic is forwarded and received over all multihomed links. IP traffic is load balanced across the multihomed links using a simple hashing algorithm.

EBGP is used to exchange routing information between the IP-connected endpoint system and the connected leaf devices to ensure the route or routes to the endpoint systems are shared with all spine and leaf devices.

For more information about the IP-connected multihoming building block, see "Multihoming an IP-Connected End System Design and Implementation" on page 251.

## Border Devices

Some of our reference designs include border devices that provide connections to the following devices, which are external to the local IP fabric:

- A multicast gateway.

- A data center gateway for data center interconnect (DCI).
- A device such as an SRX router on which multiple services such as firewalls, Network Address Translation (NAT), intrusion detection and prevention (IDP), multicast, and so on are consolidated. The consolidation of multiple services onto one physical device is known as service chaining.
- Appliances or servers that act as firewalls, DHCP servers, sFlow collectors, and so on.

> **NOTE**: If your network includes legacy appliances or servers that require a 1-Gbps Ethernet connection to a border device, we recommend using a QFX10008 or a QFX5120 switch as the border device.

To provide the additional functionality described above, Juniper Networks supports deploying a border device in the following ways:

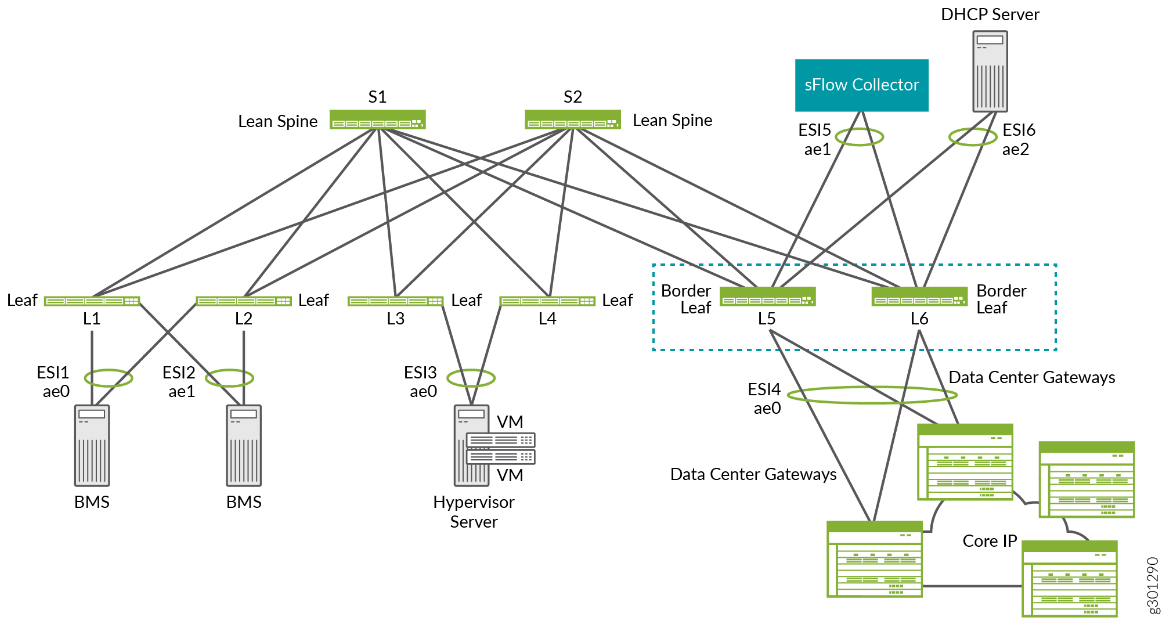
- As a device that serves as a border device only. In this dedicated role, you can configure the device to handle one or more of the tasks described above. For this situation, the device is typically deployed as a border leaf, which is connected to a spine device.

  For example, in the ERB overlay shown in Figure 14 on page 31, border leafs L5 and L6 provide connectivity to data center gateways for DCI, an sFlow collector, and a DHCP server.

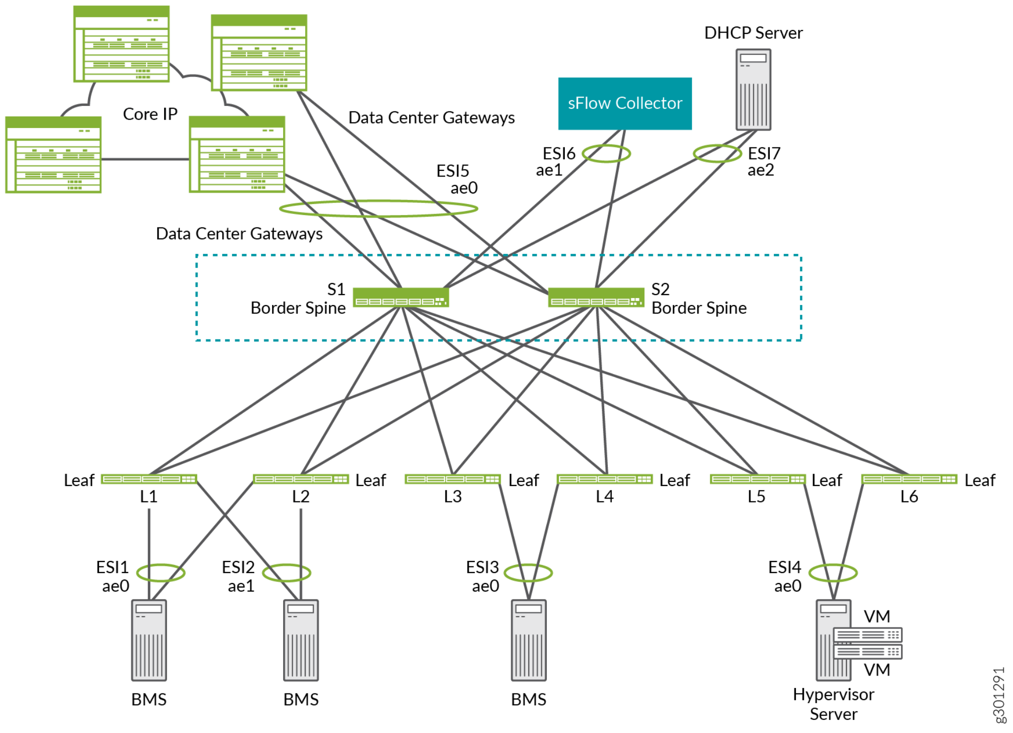
- As a device that has two roles—a network underlay device and a border device that can handle one or more of the tasks described above. For this situation, a spine device usually handles the two roles. Therefore, the border device functionality is referred to as a border spine.

  For example, in the ERB overlay shown in Figure 15 on page 31, border spines S1 and S2 function as lean spine devices. They also provide connectivity to data center gateways for DCI, an sFlow collector, and a DHCP server.

### Figure 14: Sample ERB Topology with Border Leafs



### Figure 15: Sample ERB Topology with Border Spines

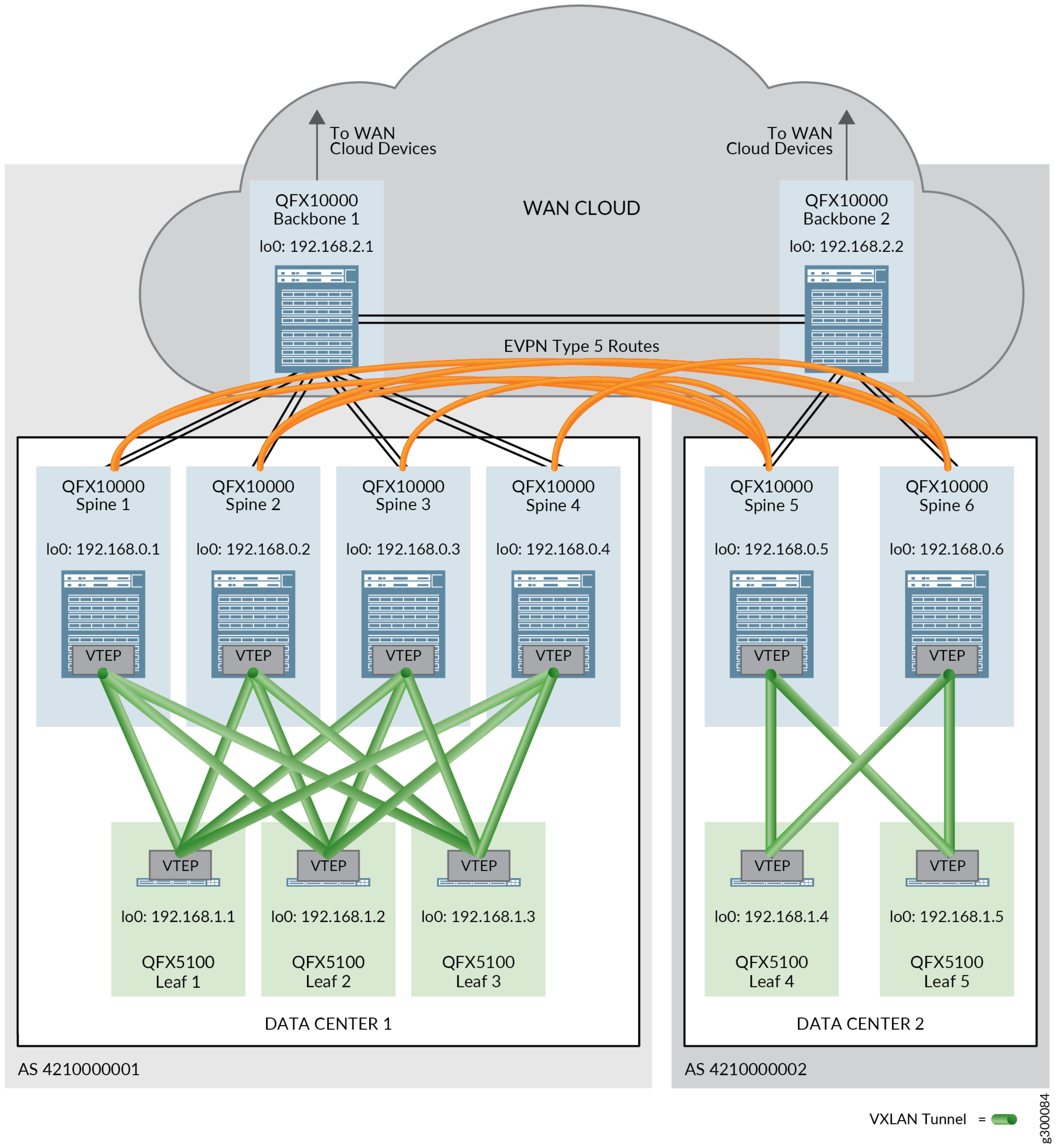


## Data Center Interconnect (DCI)

The data center interconnect (DCI) building block provides the technology needed to send traffic between data centers. The validated design supports DCI using EVPN Type 5 routes, IPVPN routes, and Layer 2 DCI with VXLAN stitching.

EVPN Type 5 or IPVPN routes are used in a DCI context to ensure inter-data center traffic between data centers using different IP address subnetting schemes can be exchanged. Routes are exchanged between spine devices in different data centers to allow for the passing of traffic between data centers.

**Figure 16: DCI Using EVPN Type 5 Routes Topology Overview**



Physical connectivity between the data centers is required before you can configure DCI. The physical connectivity is provided by backbone devices in a WAN cloud. A backbone device is connected to all spine devices in a single data center (POD), as well as to the other backbone devices that are connected to the other data centers.

For information about configuring DCI, see:

- "Data Center Interconnect Design and Implementation Using Type 5 Routes " on page 331
- "Data Center Interconnect Design and Implementation Using IPVPN" on page 357
- "Configure VXLAN Stitching for Layer 2 Data Center Interconnect" on page 361

## Service Chaining

In many networks, it is common for traffic to flow through separate hardware devices that each provide a service, such as firewalls, NAT, IDP, multicast, and so on. Each device requires separate operation and management. This method of linking multiple network functions can be thought of as physical service chaining.

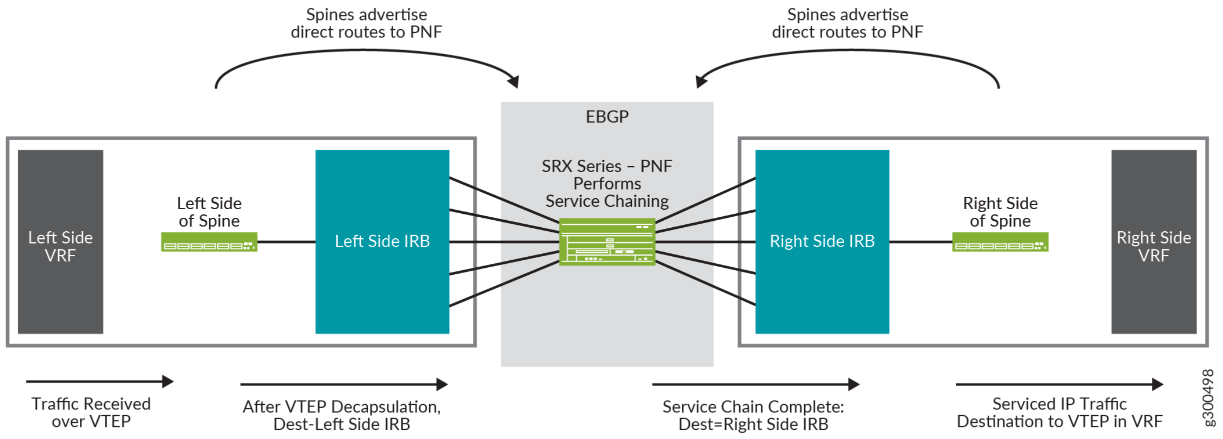
A more efficient model for service chaining is to virtualize and consolidate network functions onto a single device. In our blueprint architecture, we are using the SRX Series routers as the device that consolidates network functions and processes and applies services. That device is called a physical network function (PNF).

In this solution, service chaining is supported on both CRB overlay and ERB overlay. It works only for inter-tenant traffic.

### Logical View of Service Chaining

Figure 17 on page 34 shows a logical view of service chaining. It shows one spine with a right side configuration and a left side configuration. On each side is a VRF routing instance and an IRB interface. The SRX Series router in the center is the PNF, and it performs the service chaining.

**Figure 17: Service Chaining Logical View**


Spines advertise direct routes to PNF
Spines advertise direct routes to PNF
EBGP
SRX Series – PNF Performs Service Chaining
Left Side VRF
Left Side of Spine
Left Side IRB
Right Side IRB
Right Side of Spine
Right Side VRF
Traffic Received over VTEP
After VTEP Decapsulation, Dest-Left Side IRB
Service Chain Complete: Dest=Right Side IRB
Serviced IP Traffic Destination to VTEP in VRF
g300498

The flow of traffic in this logical view is:

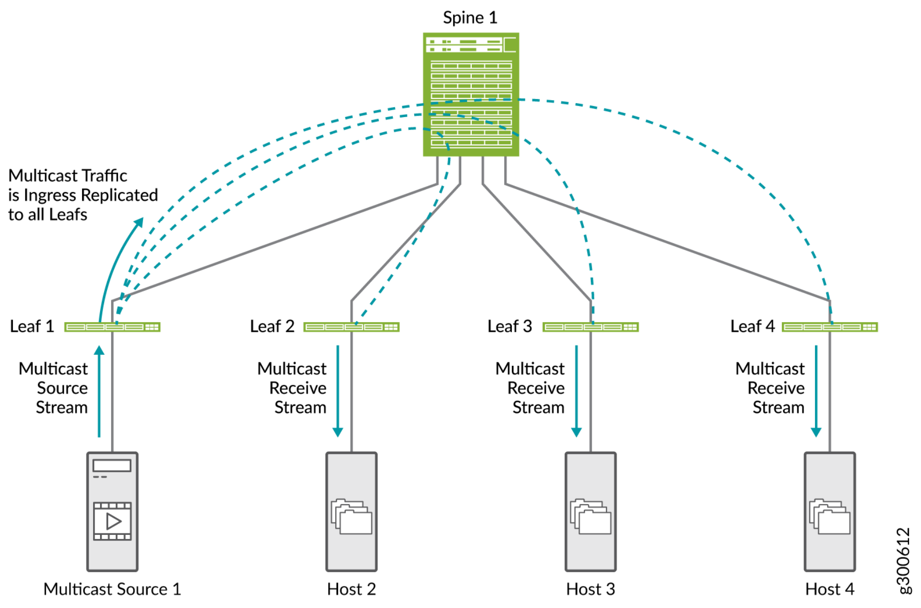
1. The spine receives a packet on the VTEP that is in the left side VRF.
2. The packet is decapsulated and sent to the left side IRB interface.
3. The IRB interface routes the packet to the SRX Series router, which is acting as the PNF.
4. The SRX Series router performs service chaining on the packet and forwards the packet back to the spine, where it is received on the IRB interface shown on the right side of the spine.
5. The IRB interface routes the packet to the VTEP in the right side VRF.

For information about configuring service chaining, see "Service Chaining Design and Implementation" on page 470.

## Multicast Optimizations

Multicast optimizations help to preserve bandwidth and more efficiently route traffic in a multicast scenario in EVPN VXLAN environments. Without any multicast optimizations configured, all multicast replication is done at the ingress of the leaf connected to the multicast source as shown in Figure 18 on page 36. Multicast traffic is sent to all leaf devices that are connected to the spine. Each leaf device sends traffic to connected hosts.

**Figure 18: Multicast without Optimizations**



There are a few types of multicast optimizations supported in EVPN VXLAN environments that can work together:

For information about configuring multicast features, see:

- "Multicast Optimization Design and Implementation" on page 495
- "Optimized Intersubnet Multicast (OISM) with Assisted Replication (AR) for Edge-Routed Bridging Overlays" on page 509

## IGMP Snooping

IGMP snooping in an EVPN-VXLAN fabric is useful to optimize the distribution of multicast traffic. IGMP snooping preserves bandwidth because multicast traffic is forwarded only on interfaces where there are IGMP listeners. Not all interfaces on a leaf device need to receive multicast traffic.

Without IGMP snooping, end systems receive IP multicast traffic that they have no interest in, which needlessly floods their links with unwanted traffic. In some cases when IP multicast flows are large, flooding unwanted traffic causes denial-of-service issues.

Figure 19 on page 37 shows how IGMP snooping works in an EVPN-VXLAN fabric. In this sample EVPN-VXLAN fabric, IGMP snooping is configured on all leaf devices, and multicast receiver 2 has previously sent an IGMPv2 join request.

1. Multicast Receiver 2 sends an IGMPv2 leave request.

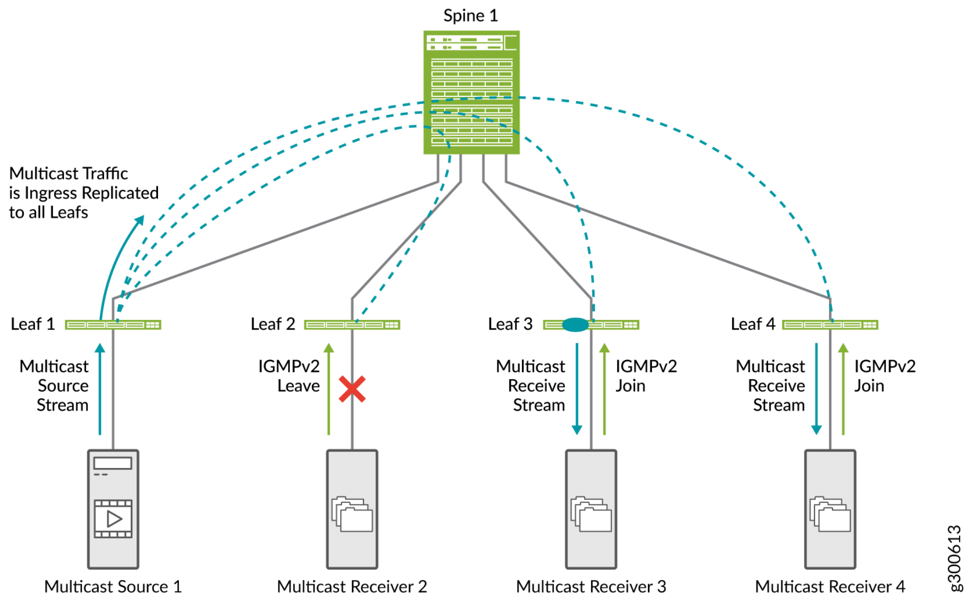
2. Multicast Receivers 3 and 4 send an IGMPv2 join request.

3. When leaf 1 receives ingress multicast traffic, it replicates it for all leaf devices, and forwards it to the spine.

4. The spine forwards the traffic to all leaf devices.

5. Leaf 2 receives the multicast traffic, but does not forward it to the receiver because the receiver sent an IGMP leave message.

**Figure 19: Multicast with IGMP Snooping**



In EVPN-VXLAN networks only IGMP version 2 is supported.

For more information about IGMP snooping, see *Overview of Multicast Forwarding with IGMP Snooping or MLD Snooping in an EVPN-VXLAN Environment*.

## Selective Multicast Forwarding

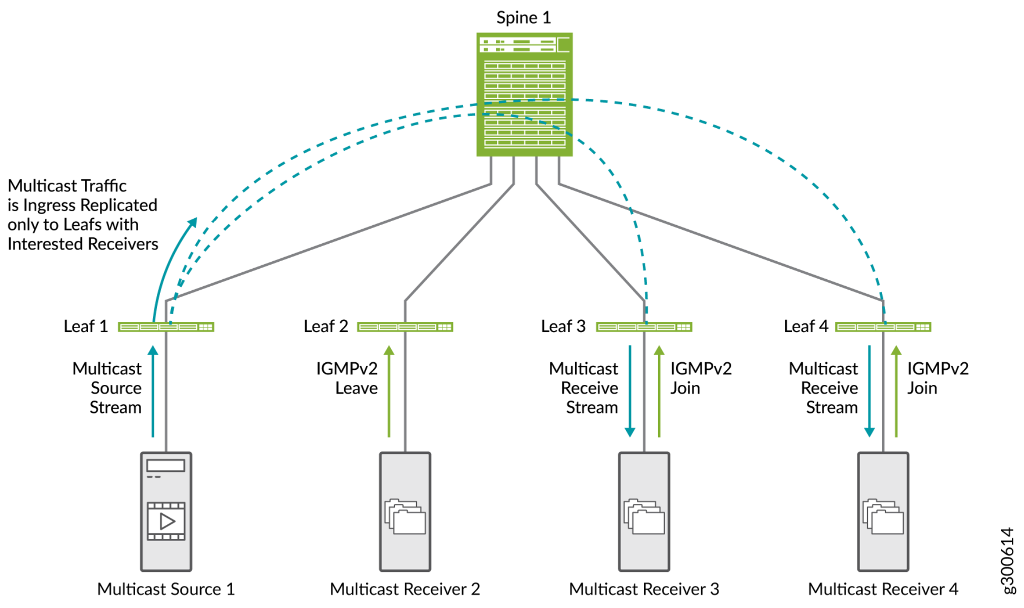
Selective multicast Ethernet (SMET) forwarding provides greater end-to-end network efficiency and reduces traffic in the EVPN network. It conserves bandwidth usage in the core of the fabric and reduces the load on egress devices that do not have listeners.

Devices with IGMP snooping enabled use selective multicast forwarding to forward multicast traffic in an efficient way. With IGMP snooping enabled a leaf device sends multicast traffic only to the access interface with an interested receiver. With SMET added, the leaf device selectively sends multicast traffic to only the leaf devices in the core that have expressed an interest in that multicast group.

Figure 20 on page 38 shows the SMET traffic flow along with IGMP snooping.

1. Multicast Receiver 2 sends an IGMPv2 leave request.
2. Multicast Receivers 3 and 4 send an IGMPv2 join request.
3. When leaf 1 receives ingress multicast traffic, it replicates the traffic only to leaf devices with interested receivers (leaf devices 3 and 4), and forwards it to the spine.
4. The spine forwards the traffic to leaf devices 3 and 4.

**Figure 20: Selective Multicast Forwarding with IGMP Snooping**



You do not need to enable SMET; it is enabled by default when IGMP snooping is configured on the device.

For more information about SMET, see *Overview of Selective Multicast Forwarding*.

## Assisted Replication of Multicast Traffic

The assisted replication (AR) feature offloads EVPN-VXLAN fabric leaf devices from ingress replication tasks. The ingress leaf does not replicate multicast traffic. It sends one copy of the multicast traffic to a spine that is configured as an AR replicator device. The AR replicator device distributes and controls multicast traffic. In addition to reducing the replication load on the ingress leaf devices, this method conserves bandwidth in the fabric between the leaf and the spine.

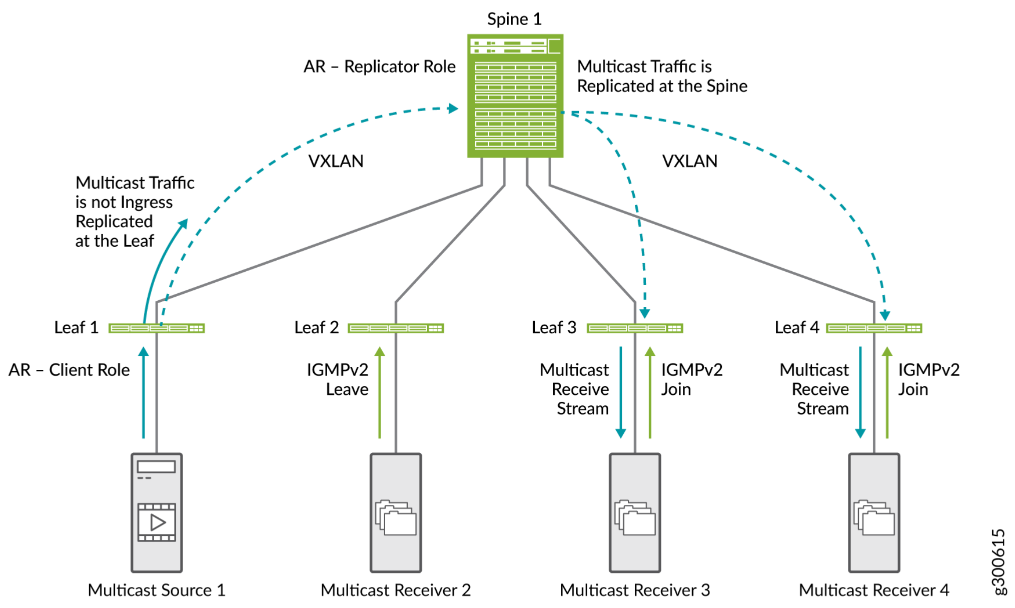
Figure 21 on page 39 shows how AR works along with IGMP snooping and SMET.

1. Leaf 1, which is set up as the AR leaf device, receives multicast traffic and sends one copy to the spine that is set up as the AR replicator device.

2. The spine replicates the multicast traffic. It replicates traffic for leaf devices that are provisioned with the VLAN VNI in which the multicast traffic originated from Leaf 1.

   Because we have IGMP snooping and SMET configured in the network, the spine sends the multicast traffic only to leaf devices with interested receivers.

**Figure 21: Multicast with AR, IGMP Snooping, and SMET**



In this document, we show multicast optimizations on a small scale. In a full-scale network with many spines and leafs, the benefits of the optimizations are much more apparent.

## Optimized Intersubnet Multicast for ERB Overlay Networks

When you have multicast sources and receivers both inside and outside an ERB overlay fabric, you can configure optimized intersubnet multicast (OISM) to enable efficient multicast traffic flow at scale.

OISM uses a local routing model for multicast traffic, which avoids traffic hairpinning and minimizes the traffic load within the EVPN core. OISM forwards multicast traffic only on the multicast source VLAN.

For intersubnet receivers, the leaf devices use IRB interfaces to locally route the traffic received on the source VLAN to other receiver VLANs on the same device. To further optimize multicast traffic flow in the EVPN-VXLAN fabric, OISM uses IGMP snooping and SMET to forward traffic in the fabric only to leaf devices with interested receivers.

OISM also enables the fabric to effectively route traffic from external multicast sources to internal receivers, and from internal multicast sources to external receivers. OISM uses a supplemental bridge domain (SBD) within the fabric to forward multicast traffic received on the border leaf devices from outside sources. The SBD design preserves the local routing model for externally-sourced traffic.

You can use OISM with AR to reduce the replication load on lower-capacity OISM leaf devices. (See "Assisted Replication of Multicast Traffic" on page 38.)

See Figure 22 on page 41 for a simple fabric with OISM and AR.

**Figure 22: OISM with AR**

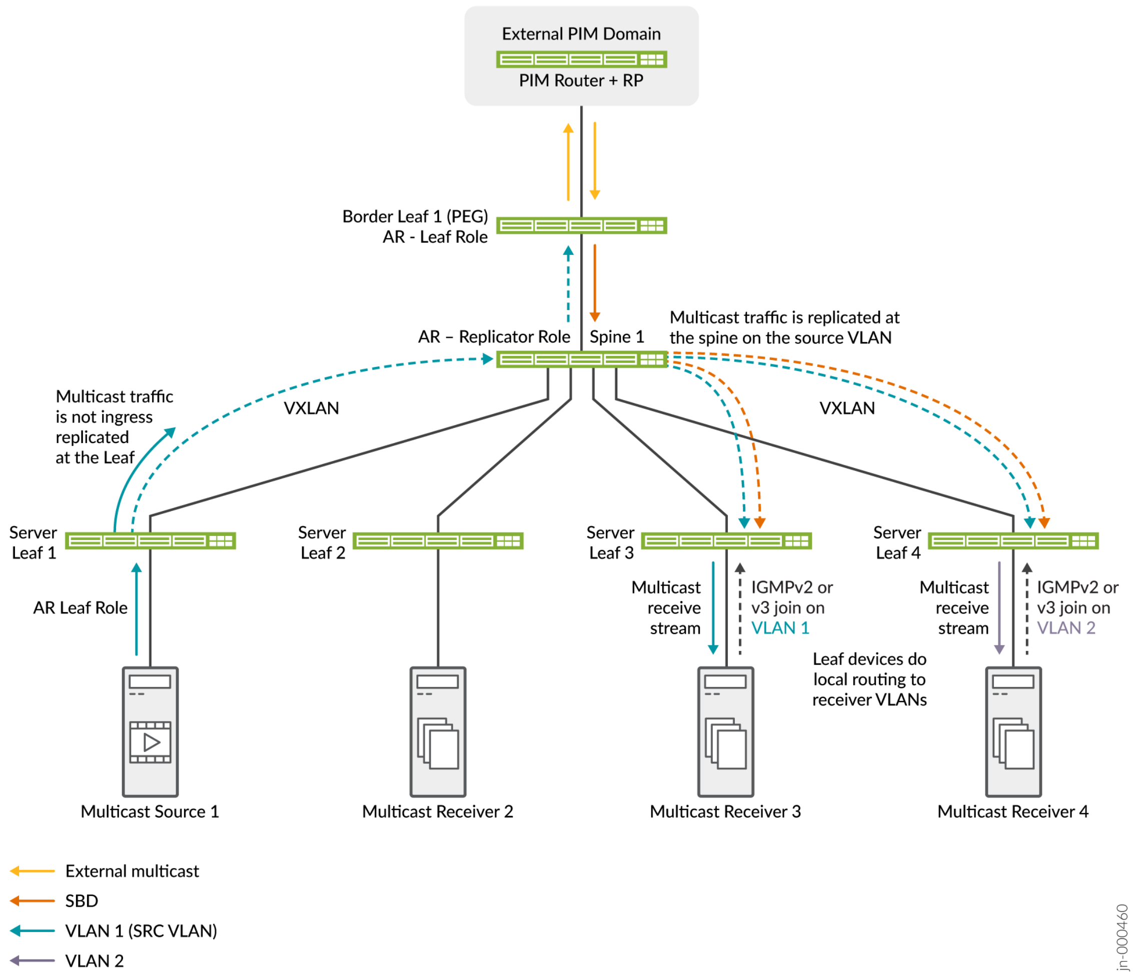


Figure 22 on page 41 shows OISM server leaf and border leaf devices, Spine 1 in the AR replicator role, and Server Leaf 1 as a multicast source in the AR leaf role. An external source and receivers might also exist in the external PIM domain. OISM and AR work together in this scenario as follows:

1. Multicast receivers behind Server Leaf 3 on VLAN 1 and behind Server Leaf 4 on VLAN 2 send IGMP Joins showing interested in the multicast group. External receivers might also join the multicast group.

2. The multicast source behind Server Leaf 1 sends multicast traffic for the group into the fabric on VLAN 1. Server Leaf 1 sends only one copy of the traffic to the AR replicator on Spine 1.

3. Also, external source traffic for the multicast group arrives at Border Leaf 1. Border Leaf 1 forwards the traffic on the SBD to Spine 1, the AR replicator.

4. The AR replicator sends copies from the internal source on the source VLAN and from the external source on the SBD to the OISM leaf devices with interested receivers.

5. The server leaf devices forward the traffic to the receivers on the source VLAN, and locally route the traffic to the receivers on the other VLANs.

## Ingress Virtual Machine Traffic Optimization for EVPN

When virtual machines and hosts are moved within a data center or from one data center to another, network traffic can become inefficient if the traffic is not routed to the optimal gateway. This can happen when a host is relocated. The ARP table does not always get flushed and data flow to the host is sent to the configured gateway even when there is a more optimal gateway. The traffic is “tromboned” and routed unnecessarily to the configured gateway.

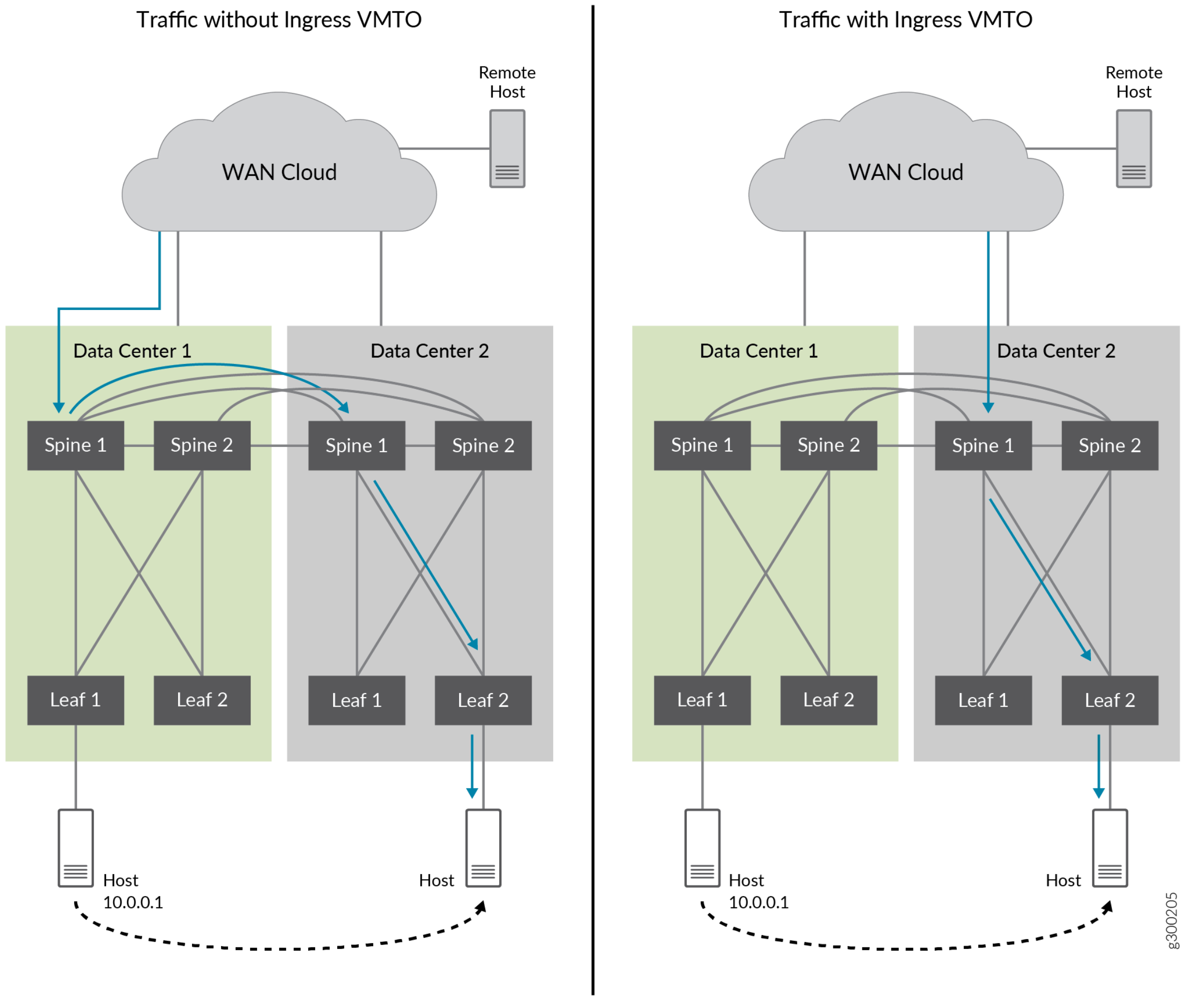
Ingress Virtual Machine Traffic Optimization (VMTO) provides greater network efficiency and optimizes ingress traffic and can eliminate the trombone effect between VLANs. When you enable ingress VMTO, routes are stored in a Layer 3 virtual routing and forwarding (VRF) table and the device routes inbound traffic directly back to host that was relocated.

Figure 23 on page 43 shows tromboned traffic without ingress VMTO and optimized traffic with ingress VMTO enabled.

- Without ingress VMTO, Spine 1 and 2 from DC1 and DC2 all advertise the remote IP host route 10.0.0.1 when the origin route is from DC2. The ingress traffic can be directed to either Spine 1 and 2 in DC1. It is then routed to Spine 1 and 2 in DC2 where route 10.0.0.1 was moved. This causes the tromboning effect.

- With ingress VMTO, we can achieve optimal forwarding path by configuring a policy for IP host route (10.0.01) to only be advertised by Spine 1 and 2 from DC2, and not from DC1 when the IP host is moved to DC2.

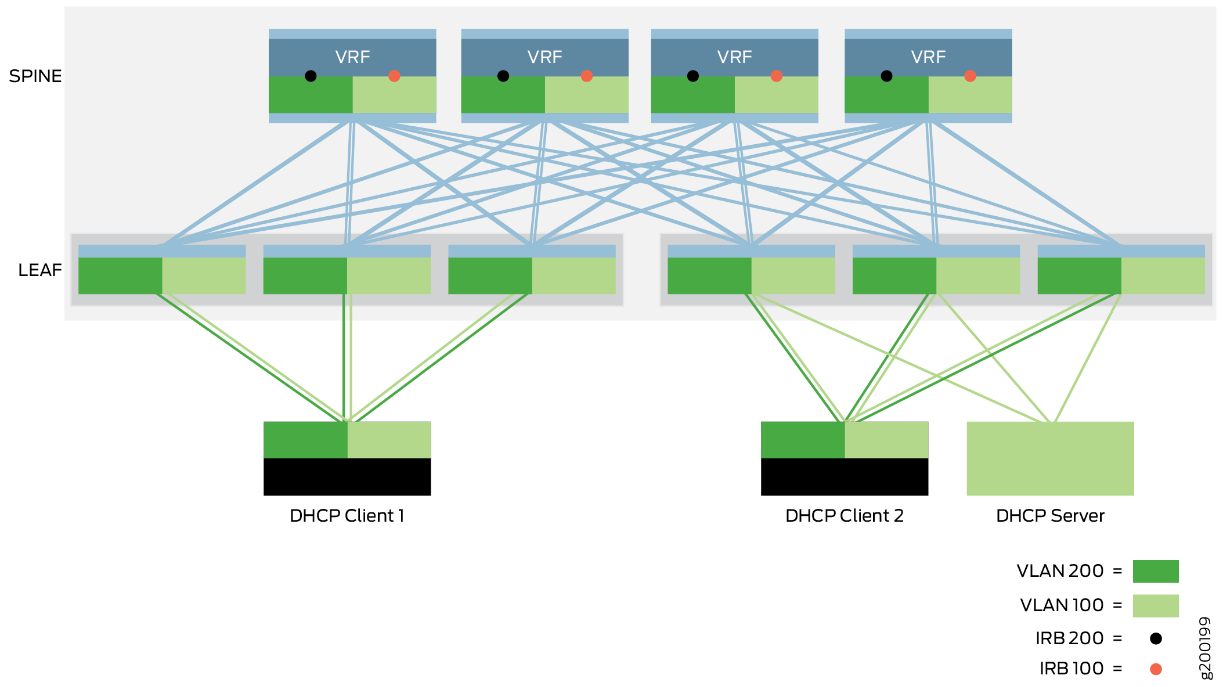
**Figure 23: Traffic with and without Ingress VMTO**



For information about configuring VMTO, see "Configuring VMTO" on page 686.

## DHCP Relay

Figure 24: DHCP Relay in a CRB Overlay



The Dynamic Host Configuration Protocol (DHCP) relay building block allows the network to pass DHCP messages between a DHCP client and a DHCP server. The DHCP relay implementation in this building block moves DHCP packets through a CRB overlay where the gateway is located at the spine layer.

The DHCP server and the DHCP clients connect into the network using access interfaces on leaf devices. The DHCP server and clients can communicate with each other over the existing network without further configuration when the DHCP client and server are in the same VLAN. When a DHCP client and server are in different VLANs, DHCP traffic between the client and server is forwarded between the VLANs via the IRB interfaces on spine devices. You must configure the IRB interfaces on the spine devices to support DHCP relay between VLANs.

For information about implementing the DHCP relay, see "DHCP Relay Design and Implementation" on page 687.

## Reducing ARP Traffic with ARP Synchronization and Suppression (Proxy ARP)

The goal of ARP synchronization is to synchronize ARP tables across all the VRFs that serve an overlay subnet to reduce the amount of traffic and optimize processing for both network devices and end systems. When an IP gateway for a subnet learns about an ARP binding, it shares it with other gateways so they do not need to discover the same ARP binding independently.

With ARP suppression, when a leaf device receives an ARP request, it checks its own ARP table that is synchronized with the other VTEP devices and responds to the request locally rather than flooding the ARP request.

Proxy ARP and ARP suppression are enabled by default on all QFX Series switches that can act as leaf devices in an ERB overlay. For a list of these switches, see "Data Center EVPN-VXLAN Fabric Reference Designs—Supported Hardware Summary" on page 51.

IRB interfaces on the leaf device deliver ARP requests and NDP requests from both local and remote leaf devices. When a leaf device receives an ARP request or NDP request from another leaf device, the receiving device searches its MAC+IP address bindings database for the requested IP address.

- If the device finds the MAC+IP address binding in its database, it responds to the request.
- If the device does not find the MAC+IP address binding, it floods the ARP request to all Ethernet links in the VLAN and the associated VTEPs.

Because all participating leaf devices add the ARP entries and synchronize their routing and bridging tables, local leaf devices respond directly to requests from locally connected hosts and remove the need for remote devices to respond to these ARP requests.

For information about implementing the ARP synchronization, Proxy ARP, and ARP suppression, see Enabling Proxy ARP and ARP Suppression for the Edge-Routed Bridging Overlay.

## Layer 2 Port Security Features on Ethernet-Connected End Systems

CRB and ERB overlays support the security features on Layer 2 Ethernet-connected end systems that we describe in the next sections.

For more information about these features, see *MAC Filtering, Storm Control, and Port Mirroring Support in an EVPN-VXLAN Environment*.

For information about configuring these features, see "Configuring Layer 2 Port Security Features on Ethernet-Connected End Systems" on page 696.

### Preventing BUM Traffic Storms With Storm Control

Storm control can prevent excessive traffic from degrading the network. It lessens the impact of BUM traffic storms by monitoring traffic levels on EVPN-VXLAN interfaces, and dropping BUM traffic when a specified traffic level is exceeded.

In an EVPN-VXLAN environment, storm control monitors:

- Layer 2 BUM traffic that originates in a VXLAN and is forwarded to interfaces within the same VXLAN.
- Layer 3 multicast traffic that is received by an IRB interface in a VXLAN and is forwarded to interfaces in another VXLAN.

### Using MAC Filtering to Enhance Port Security

MAC filtering enhances port security by limiting the number of MAC addresses that can be learned within a VLAN and therefore limit the traffic in a VXLAN. Limiting the number of MAC addresses protects the switch from flooding the Ethernet switching table. Flooding of the Ethernet switching table occurs when the number of new MAC addresses that are learned causes the table to overflow, and previously learned MAC addresses are flushed from the table. The switch relearns the MAC addresses, which can impact performance and introduce security vulnerabilities.

In this blueprint, MAC filtering limits the number of accepted packets that are sent to ingress-facing access interfaces based on MAC addresses. For more information about how MAC filtering works, see the MAC limiting information in Understanding MAC Limiting and MAC Move Limiting.

### Analyzing Traffic Using Port Mirroring

With analyzer-based port mirroring, you can analyze traffic down to the packet level in an EVPN-VXLAN environment. You can use this feature to enforce policies related to network usage and file sharing and to identify problem sources by locating abnormal or heavy bandwidth usage by particular stations or applications.

Port mirroring copies packets entering or exiting a port or entering a VLAN and sends the copies to a local interface for local monitoring or to a VLAN for remote monitoring. Use port mirroring to send traffic to applications that analyze traffic for purposes such as monitoring compliance, enforcing policies, detecting intrusions, monitoring and predicting traffic patterns, correlating events, and so on.

RELATED DOCUMENTATION

Infrastructure as a Service: EVPN and VXLAN Solution Guide

Juniper Networks EVPN Implementation for Next-Generation Data Center Architectures

2

CHAPTER

# Data Center Fabric Reference Design—Tested Implementation

# Data Center Fabric Reference Design Overview and Validated Topology

IN THIS SECTION

This section provides a high-level overview of the Data Center Fabric reference design topology and summarizes the topologies that were tested and validated by the Juniper Networks Test Team.

**NOTE**: Contact your local account manager for details on the tested scale for all supported features.

## Reference Design Overview

The Data Center Fabric reference design tested by Juniper Networks is based on an IP Fabric underlay in a Clos topology that uses the following devices:

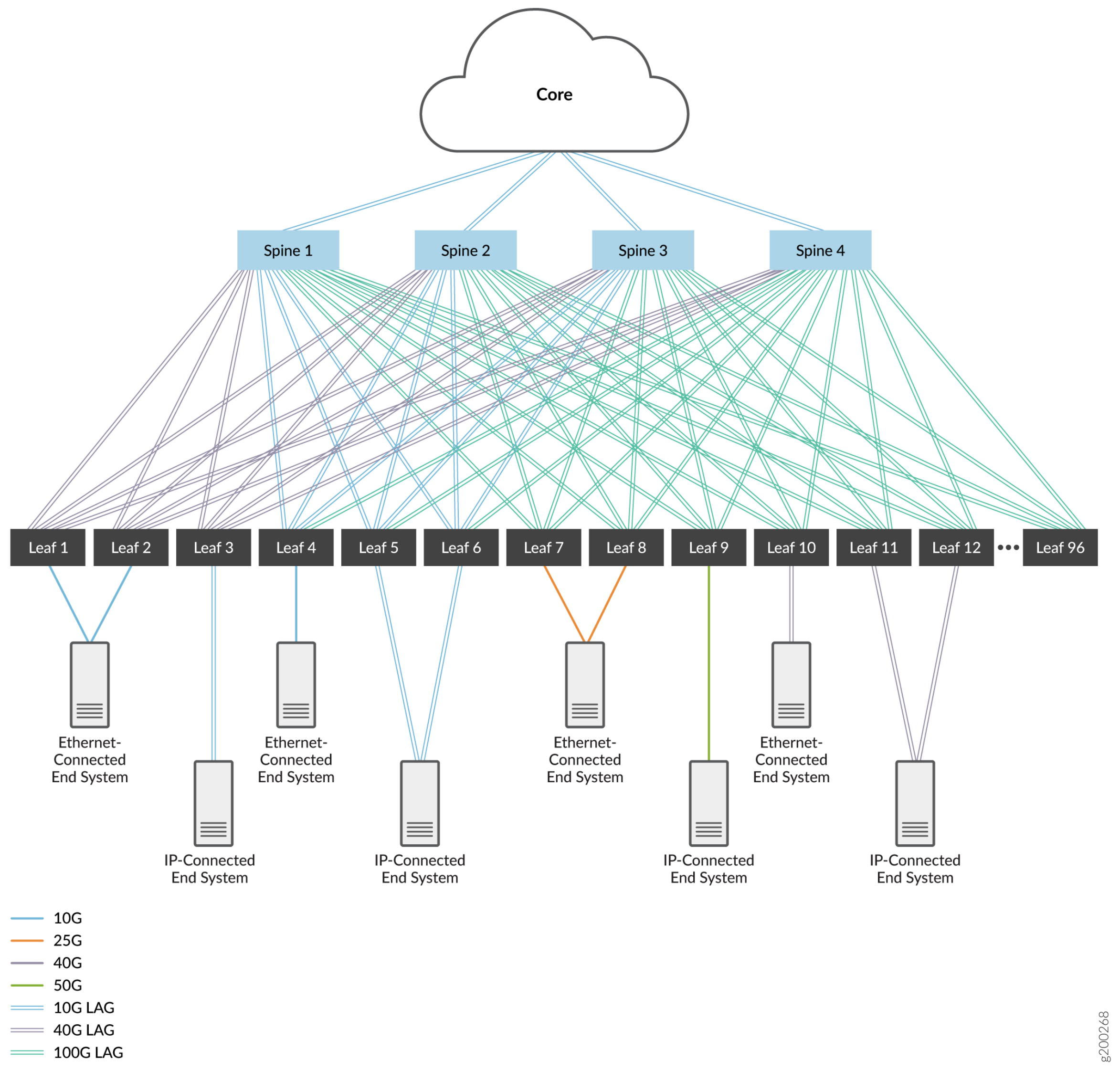
- Spine devices: up to four.
- Leaf devices: the number of leaf devices supported by Juniper Networks varies depending on the Junos OS or Junos OS Evolved software release and overlay type (centrally-routed bridging overlay or edge-routed bridging overlay).

  The number of tested leaf nodes reflected throughout this guide is 96, which was number tested in the initial reference design.

Each leaf device is interconnected to each spine device using either an aggregated Ethernet interface that includes two high-speed Ethernet interfaces (10-Gbps, 40-Gbps, or 100-Gbps) as LAG members or a single high-speed Ethernet interface.

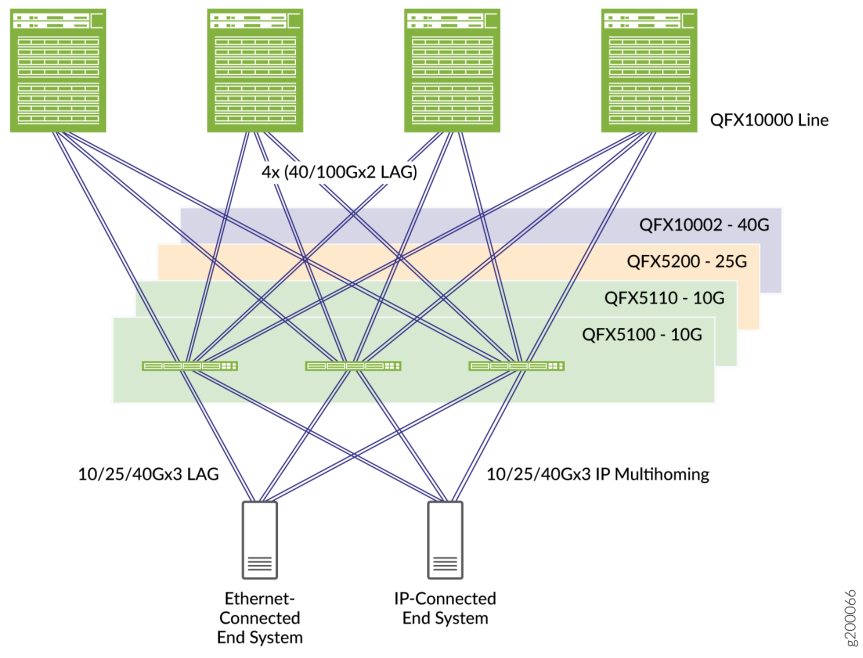
Figure 25 on page 50 provides an illustration of the topology used in this reference design:

**Figure 25: Data Center Fabric Reference Design - Topology**



End systems such as servers connect to the data center network through leaf device interfaces. Each end system was multihomed to three leaf devices using a 3-member aggregated Ethernet interface as shown in Figure 26 on page 51.

**Figure 26: Data Center Fabric Reference Design - Multihoming**



The objective for multihoming end systems to 3 different leaf devices is to verify that multihoming of an end system to more than 2 leaf devices is fully supported.

## Data Center EVPN-VXLAN Fabric Reference Designs—Supported Hardware Summary

Table 2 on page 52 provides a summary of the hardware that you can use to create the reference designs described in this guide. The table is organized by use cases, roles that devices play in each use case, and the hardware supported for each role.

**NOTE:**

- For each use case, we support the hardware listed in Table 2 on page 52 only with the associated Junos OS or Junos OS Evolved software releases.
- To learn about any existing issues and limitations for a hardware device in this reference design, see the release notes for the Junos OS or Junos OS Evolved software release with which the device was tested.

**Table 2: Data Center Fabric Reference Design Supported Hardware Summary**

| Device Roles | Hardware[4] | Junos OS or Junos OS Evolved (EVO) Software Releases[1] |
|---|---|---|
| **Centrally Routed Bridging Overlay** | | |
| Spine | QFX10002-36Q/72Q<br>QFX10008<br>QFX10016<br>MX Series | 17.3R3-S1 |
| | QFX5120-32C<br>QFX10002-60C | 19.1R2 |
| | PTX10001<br>PTX10004<br>PTX10008<br>QFX5130-32CD<br>QFX5700 | 23.2R2-EVO |
| Server leaf | QFX5100<br>QFX5110<br>QFX5200<br>QFX10002-36Q/72Q | 17.3R3-S1 |
| | QFX5210-64C | 18.1R3-S3 |
| | QFX5120-48Y | 18.4R2 |
| | QFX5120-32C<br>QFX10002-60C | 19.1R2 |

**Table 2: Data Center Fabric Reference Design Supported Hardware Summary** ***(Continued)***

| Device Roles | Hardware[4] | Junos OS or Junos OS Evolved (EVO) Software Releases[1] |
|---|---|---|
| | QFX5120-48T | 20.2R2 |
| | QFX5120-48YM[4] | 20.4R3 |
| Border spine[5] (data center gateway, and data center interconnect (DCI) using IPVPN) | QFX10002-36Q/72Q<br>QFX10008<br>QFX10016 | 17.3R3-S1 |
| | QFX5110 | 18.1R3-S3 |
| | QFX5120-48Y | 18.4R2 |
| | MX204<br>MX240, MX480, and MX960 with MPC7E<br>MX10003<br>MX10008 | 18.4R2-S2 |
| | QFX5120-32C<br>QFX10002-60C[2] | 19.1R2 |
| | QFX10002-60C[3] | 20.2R2 |
| | QFX5120-48YM | 20.4R3 |
| | QFX5130-32CD<br>QFX5700 | 23.2R2-EVO |

**Table 2: Data Center Fabric Reference Design Supported Hardware Summary *(Continued)***

| Device Roles | Hardware[4] | Junos OS or Junos OS Evolved (EVO) Software Releases[1] |
|---|---|---|
| Border spine[5] (data center gateway, and data center interconnect (DCI) using EVPN Type 5 routes) | QFX10002-36Q/72Q<br><br>QFX10008<br><br>QFX10016 | 17.3R3-S1 |
| | QFX5110 | 18.1R3-S3 |
| | QFX5120-48Y | 18.4R2 |
| | MX204<br><br>MX240, MX480, and MX960 with MPC7E<br><br>MX10003<br><br>MX10008 | 18.4R2-S2 |
| | QFX5120-32C<br><br>QFX10002-60C[2] | 19.1R2 |
| | QFX10002-60C[3] | 20.2R2 |
| | QFX5120-48YM | 20.4R3 |
| | PTX10001<br><br>PTX10004<br><br>PTX10008<br><br>QFX5130-32CD<br><br>QFX5700 | 23.2R2-EVO |

**Table 2: Data Center Fabric Reference Design Supported Hardware Summary** ***(Continued)***

| Device Roles | Hardware[4] | Junos OS or Junos OS Evolved (EVO) Software Releases[1] |
|---|---|---|
| Border spine[5] (service chaining) | QFX10002-36Q/72Q<br>QFX10008<br>QFX10016 | 17.3R3-S1 |
| | QFX5110 | 18.1R3-S3 |
| | MX204<br>MX240, MX480, and MX960 with MPC7E<br>MX10003<br>MX10008 | 18.4R2-S2 |
| | QFX10002-60C[2] | 19.1R2 |
| | QFX5120-32C<br>QFX5120-48T<br>QFX5120-48Y<br>QFX10002-60C[3] | 20.2R2 |
| | QFX5120-48YM | 20.4R3 |
| | PTX10001<br>PTX10004<br>PTX10008<br>QFX5130-32CD<br>QFX5700 | 23.2R2-EVO |

**Table 2: Data Center Fabric Reference Design Supported Hardware Summary *(Continued)***

| Device Roles | Hardware[4] | Junos OS or Junos OS Evolved (EVO) Software Releases[1] |
|---|---|---|
| Border leaf (data center gateway, and DCI using IPVPN) | QFX10002-36Q/72Q<br>QFX10008<br>QFX10016 | 17.3R3-S1 |
| | QFX5110 | 18.1R3-S3 |
| | QFX5120-48Y | 18.4R2 |
| | MX204<br>MX240, MX480, and MX960 with MPC7E<br>MX10003<br>MX10008 | 18.4R2-S2 |
| | QFX5120-32C<br>QFX10002-60C[2] | 19.1R2 |
| | QFX10002-60C[3] | 20.2R2 |
| | QFX5120-48YM | 20.4R3 |
| Border leaf (data center gateway, and DCI using EVPN Type 5 routes) | QFX10002-36Q/72Q<br>QFX10008<br>QFX10016 | 17.3R3-S1 |
| | QFX5110 | 18.1R3-S3 |
| | QFX5120-48Y | 18.4R2 |

**Table 2: Data Center Fabric Reference Design Supported Hardware Summary *(Continued)***

| Device Roles | Hardware[4] | Junos OS or Junos OS Evolved (EVO) Software Releases[1] |
|---|---|---|
| | MX204<br><br>MX240, MX480, and MX960 with MPC7E<br><br>MX10003<br><br>MX10008 | 18.4R2-S2 |
| | QFX5120-32C<br><br>QFX10002-60C[2] | 19.1R2 |
| | QFX10002-60C[3] | 20.2R2 |
| | QFX5120-48YM | 20.4R3 |
| | PTX10001<br><br>PTX10004<br><br>PTX10008 | 23.2R2-EVO |
| Border leaf (service chaining) | QFX10002-36Q/72Q<br><br>QFX10008<br><br>QFX10016 | 17.3R3-S1 |
| | QFX5110 | 18.1R3-S3 |

**Table 2: Data Center Fabric Reference Design Supported Hardware Summary** ***(Continued)***

| Device Roles | Hardware[4] | Junos OS or Junos OS Evolved (EVO) Software Releases[1] |
|---|---|---|
| | MX204<br>MX240, MX480, and MX960 with MPC7E<br>MX10003<br>MX10008 | 18.4R2-S2 |
| | QFX10002-60C[2] | 19.1R2 |
| | QFX5120-32C<br>QFX5120-48T<br>QFX5120-48Y<br>QFX10002-60C[3] | 20.2R2 |
| | QFX5120-48YM | 20.4R3 |
| | PTX10001<br>PTX10004<br>PTX10008 | 23.2R2-EVO |

**Table 2: Data Center Fabric Reference Design Supported Hardware Summary** ***(Continued)***

| Device Roles | Hardware[4] | Junos OS or Junos OS Evolved (EVO) Software Releases[1] |
|---|---|---|
| Lean super spine | QFX5110<br>QFX5120-32C<br>QFX5120-48Y<br>QFX5200<br>QFX5210-64C<br>QFX5220-32CD<br>QFX5220-128C<br>QFX10002-36Q/72Q/60C<br>QFX10008<br>QFX10016 | 20.2R2 |
| | QFX5120-48YM | 20.4R3 |
| | ACX7100-32C<br>ACX7100-48L<br>PTX10001<br>PTX10004<br>PTX10008<br>QFX5130-32CD<br>QFX5700 | 21.2R2-S1-EVO |
| | ACX7024 | 22.4R2-EVO |
| **Edge-Routed Bridging Overlay** | | |

**Table 2: Data Center Fabric Reference Design Supported Hardware Summary *(Continued)***

| Device Roles | Hardware[4] | Junos OS or Junos OS Evolved (EVO) Software Releases[1] |
|---|---|---|
| Lean spine | QFX5200<br>QFX10002-36Q/72Q<br>QFX10008<br>QFX10016 | 17.3R3-S1 |
| | QFX5110<br>QFX5210-64C | 18.1R3-S3 |
| | QFX5120-48Y | 18.4R2 |
| | QFX5120-32C<br>QFX10002-60C | 19.1R2 |
| | QFX5220-32CD<br>QFX5220-128C | 20.2R2 |
| | QFX5120-48YM | 20.4R3 |
| | ACX7100-32C<br>ACX7100-48L<br>PTX10001<br>PTX10004<br>PTX10008<br>QFX5130-32CD<br>QFX5700 | 21.2R2-S1-EVO |
| | ACX7024 | 22.4R2-EVO |

**Table 2: Data Center Fabric Reference Design Supported Hardware Summary *(Continued)***

| Device Roles | Hardware[4] | Junos OS or Junos OS Evolved (EVO) Software Releases[1] |
|---|---|---|
| Lean spine with IPv6 underlay | QFX5120-32C<br>QFX5120-48Y<br>QFX10002-36Q/72Q<br>QFX10002-60C<br>QFX10008<br>QFX10016 | 21.2R2-S1 |
| | QFX5120-48YM | 21.4R2 |
| Server leaf | QFX10002-36Q/72Q<br>QFX10008<br>QFX10016 | 17.3R3-S1 |
| | QFX10002-60C | 19.1R2 |
| | QFX5110 | 18.1R3-S3 |
| | QFX5120-48Y | 18.4R2 |
| | QFX5120-32C | 19.1R2 |
| | QFX5120-48T | 20.2R2 |
| | QFX5120-48YM | 20.4R3 |
| | QFX5130-32CD<br>QFX5700 | 21.2R2-S1-EVO |

**Table 2: Data Center Fabric Reference Design Supported Hardware Summary** ***(Continued)***

| Device Roles | Hardware[4] | Junos OS or Junos OS Evolved (EVO) Software Releases[1] |
|---|---|---|
| | ACX7100-32C<br>ACX7100-48L<br>PTX10001<br>PTX10004<br>PTX10008 | 21.4R2-EVO |
| | ACX7024 | 22.4R2-EVO |
| Server leaf with IPv6 underlay | QFX5120-32C<br>QFX5120-48T<br>QFX5120-48Y<br>QFX10002-36Q/72Q<br>QFX10002-60C<br>QFX10008<br>QFX10016 | 21.2R2-S1 |
| | QFX5120-48YM | 21.4R2 |
| Server leaf with optimized intersubnet multicast (OISM) | QFX5110<br>QFX5120-32C<br>QFX5120-48T<br>QFX5120-48Y<br>QFX10002-36Q/72Q<br>QFX10002-60C<br>QFX10008<br>QFX10016 | 21.4R2 |

**Table 2: Data Center Fabric Reference Design Supported Hardware Summary *(Continued)***

| Device Roles | Hardware[4] | Junos OS or Junos OS Evolved (EVO) Software Releases[1] |
|---|---|---|
| | QFX5130-32CD<br>QFX5700 | 22.2R2-EVO |
| | PTX10001<br>PTX10004<br>PTX10008 | 23.2R2-EVO |
| Server leaf with Enhanced OISM | QFX5120-32C<br>QFX5120-48T<br>QFX5120-48Y | 23.4R2 |
| | QFX5130-32CD<br>QFX5700 | 23.4R2-EVO |
| Border spine[5] (data center gateway and DCI using IPVPN) | QFX10002-36Q/72Q<br>QFX10008<br>QFX10016 | 17.3R3-S1 |
| | QFX5110 | 18.1R3-S3 |
| | MX204<br>MX240, MX480, and MX960 with MPC7E<br>MX10003<br>MX10008<br>QFX5120-48Y[2] | 18.4R2 |

**Table 2: Data Center Fabric Reference Design Supported Hardware Summary *(Continued)***

| Device Roles | Hardware[4] | Junos OS or Junos OS Evolved (EVO) Software Releases[1] |
|---|---|---|
| | QFX5120-32C<br><br>QFX10002-60C[2] | 19.1R2 |
| | QFX5120-48T[2]<br><br>QFX10002-60C[3] | 20.2R2 |
| | QFX5120-48YM | 20.4R3 |
| | QFX5130-32CD<br><br>QFX5700 | 21.2R2-S1-EVO |
| Border spine[5] (data center gateway and DCI using EVPN Type 5 routes) | QFX10002-36Q/72Q<br><br>QFX10008<br><br>QFX10016 | 17.3R3-S1 |
| | QFX5110 | 18.1R3-S3 |
| | MX204<br><br>MX240, MX480, and MX960 with MPC7E<br><br>MX10003<br><br>MX10008<br><br>QFX5120-48Y[2] | 18.4R2 |
| | QFX5120-32C<br><br>QFX10002-60C[2] | 19.1R2 |

**Table 2: Data Center Fabric Reference Design Supported Hardware Summary** ***(Continued)***

| Device Roles | Hardware[4] | Junos OS or Junos OS Evolved (EVO) Software Releases[1] |
|---|---|---|
| | QFX5120-48T[2]<br>QFX10002-60C[3] | 20.2R2 |
| | QFX5120-48YM | 20.4R3 |
| | QFX5130-32CD<br>QFX5700 | 21.2R2-S1-EVO |
| | ACX7100-32C<br>ACX7100-48L<br>PTX10001<br>PTX10004<br>PTX10008 | 21.4R2-EVO |
| | ACX7024 | 22.4R2-EVO |
| Border spine[5] (service chaining) | QFX10002-36Q/72Q<br>QFX10008<br>QFX10016 | 17.3R3-S1 |
| | QFX5110 | 18.1R3-S3 |
| | QFX5120-48Y[2] | 18.4R2 |

**Table 2: Data Center Fabric Reference Design Supported Hardware Summary *(Continued)***

| Device Roles | Hardware[4] | Junos OS or Junos OS Evolved (EVO) Software Releases[1] |
|---|---|---|
| | MX204<br><br>MX240, MX480, and MX960 with MPC7E<br><br>MX10003<br><br>MX10008 | 18.4R2-S2 |
| | QFX5120-32C<br><br>QFX10002-60C[2] | 19.1R2 |
| | QFX5120-48T[2]<br><br>QFX10002-60C[3] | 20.2R2 |
| | QFX5120-48YM | 20.4R3 |
| | QFX5130-32CD<br><br>QFX5700 | 21.2R2-S1-EVO |
| | ACX7100-32C<br><br>ACX7100-48L<br><br>PTX10001<br><br>PTX10004<br><br>PTX10008 | 21.4R2-EVO |
| | ACX7024 | 22.4R2-EVO |

**Table 2: Data Center Fabric Reference Design Supported Hardware Summary *(Continued)***

| Device Roles | Hardware[4] | Junos OS or Junos OS Evolved (EVO) Software Releases[1] |
|---|---|---|
| Border spine with IPv6 underlay[5] (including data center gateway, DCI using EVPN Type 5 routes and IPVPN, and service chaining) | QFX5120-32C<br>QFX5120-48T<br>QFX5120-48Y<br>QFX10002-36Q/72Q<br>QFX10002-60C<br>QFX10008<br>QFX10016 | 21.2R2-S1 |
| | QFX5120-48YM | 21.4R2 |
| Border spine with DCI gateway—EVPN-VXLAN to EVPN-MPLS stitching | ACX7024<br>ACX7100-32C<br>ACX7100-48L<br>PTX10001<br>PTX10008 | 23.4R2-EVO |
| Border leaf (data center gateway, and DCI using IPVPN) | QFX10002-36Q/72Q<br>QFX10008<br>QFX10016 | 17.3R3-S1 |
| | QFX5110 | 18.1R3-S3 |
| | QFX5120-48Y[2] | 18.4R2 |

**Table 2: Data Center Fabric Reference Design Supported Hardware Summary *(Continued)***

| Device Roles | Hardware[4] | Junos OS or Junos OS Evolved (EVO) Software Releases[1] |
|---|---|---|
| | MX204<br><br>MX240, MX480, and MX960 with MPC7E<br><br>MX10003<br><br>MX10008 | 18.4R2-S2 |
| | QFX5120-32C | 19.1R2 |
| | QFX5120-48T[2]<br><br>QFX10002-60C[3] | 20.2R2 |
| | QFX5120-48YM | 20.4R3 |
| | QFX5130-32CD<br><br>QFX5700 | 21.2R2-S1-EVO |
| Border leaf (data center gateway, and DCI using EVPN Type 5 routes) | QFX10002-36Q/72Q<br><br>QFX10008<br><br>QFX10016 | 17.3R3-S1 |
| | QFX5110 | 18.1R3-S3 |
| | QFX5120-48Y[2] | 18.4R2 |

**Table 2: Data Center Fabric Reference Design Supported Hardware Summary** ***(Continued)***

| Device Roles | Hardware[4] | Junos OS or Junos OS Evolved (EVO) Software Releases[1] |
|---|---|---|
| | MX204<br><br>MX240, MX480, and MX960 with MPC7E<br><br>MX10003<br><br>MX10008 | 18.4R2-S2 |
| | QFX5120-32C | 19.1R2 |
| | QFX5120-48T[2]<br><br>QFX10002-60C[3] | 20.2R2 |
| | QFX5120-48YM | 20.4R3 |
| | QFX5130-32CD<br><br>QFX5700 | 21.2R2-S1-EVO |
| | ACX7100-32C<br><br>ACX7100-48L<br><br>PTX10001<br><br>PTX10004<br><br>PTX10008 | 21.4R2-EVO |
| | ACX7024 | 22.4R2-EVO |
| Border leaf (service chaining) | QFX10002-36Q/72Q<br><br>QFX10008<br><br>QFX10016 | 17.3R3-S1 |

Table 2: Data Center Fabric Reference Design Supported Hardware Summary *(Continued)*

| Device Roles | Hardware[4] | Junos OS or Junos OS Evolved (EVO) Software Releases[1] |
|---|---|---|
| | QFX5110 | 18.1R3-S3 |
| | MX204<br><br>MX240, MX480, and MX960 with MPC7E<br><br>MX10003<br><br>MX10008 | 18.4R2-S2 |
| | QFX5120-32C<br><br>QFX5120-48T<br><br>QFX5120-48Y<br><br>QFX10002-60C[3] | 20.2R2 |
| | QFX5120-48YM | 20.4R3 |
| | QFX5130-32CD<br><br>QFX5700 | 21.2R2-S1-EVO |
| | ACX7100-32C<br><br>ACX7100-48L<br><br>PTX10001<br><br>PTX10004<br><br>PTX10008 | 21.4R2-EVO |
| | ACX7024 | 22.4R2-EVO |

**Table 2: Data Center Fabric Reference Design Supported Hardware Summary** ***(Continued)***

| Device Roles | Hardware[4] | Junos OS or Junos OS Evolved (EVO) Software Releases[1] |
|---|---|---|
| Border leaf with IPv6 underlay (including data center gateway, DCI using EVPN Type 5 routes and IPVPN, and service chaining) | QFX5120-32C<br>QFX5120-48T<br>QFX5120-48Y<br>QFX10002-36Q/72Q<br>QFX10002-60C<br>QFX10008<br>QFX10016 | 21.2R2-S1 |
| | QFX5120-48YM | 21.4R2 |
| Border leaf with DCI gateway and asymmetric IRB—EVPN-VXLAN to EVPN-VXLAN stitching for Layer 2 | QFX10002-36Q/72Q<br>QFX10002-60C<br>QFX10008<br>QFX10016 | 20.4R3-S1 |
| | QFX5130-32CD<br>QFX5700 | 22.2R2-EVO |
| | QFX5120-32C<br>QFX5120-48T<br>QFX5120-48Y<br>QFX5120-48YM | 23.2R2 |

**Table 2: Data Center Fabric Reference Design Supported Hardware Summary *(Continued)***

| Device Roles | Hardware[4] | Junos OS or Junos OS Evolved (EVO) Software Releases[1] |
|---|---|---|
| Border leaf with DCI gateway and symmetric IRB—EVPN-VXLAN to EVPN-VXLAN stitching for Layer 2 | QFX5120-32C<br>QFX5120-48T<br>QFX5120-48Y<br>QFX5120-48YM<br>QFX10002-36Q/72Q<br>QFX10008<br>QFX10016 | 23.2R2 |
| | ACX7100-32C<br>ACX7100-48L<br>QFX5130-32CD<br>QFX5700 | 23.2R2-EVO |

**Table 2: Data Center Fabric Reference Design Supported Hardware Summary *(Continued)***

| Device Roles | Hardware[4] | Junos OS or Junos OS Evolved (EVO) Software Releases[1] |
|---|---|---|
| Border leaf with DCI gateway—EVPN-VXLAN to EVPN-VXLAN stitching for EVPN Type 5 routes | MX204<br><br>MX240,MX480, and MX960 with MPC7E<br><br>MX10003<br><br>MX10008<br><br>QFX5120-32C<br><br>QFX5120-48T<br><br>QFX5120-48Y<br><br>QFX5120-48YM<br><br>QFX10002-36Q/72Q<br><br>QFX10002-60C<br><br>QFX10008<br><br>QFX10016 | 22.4R2 |
| | ACX7024<br><br>ACX7100-32C<br><br>ACX7100-48L<br><br>QFX5130-32CD<br><br>QFX5700<br><br>PTX10001<br><br>PTX10004<br><br>PTX10008 | 22.4R2-EVO |
| Border leaf with DCI gateway—EVPN-VXLAN to EVPN-MPLS stitching | MX Series | 21.4R2 |

**Table 2: Data Center Fabric Reference Design Supported Hardware Summary *(Continued)***

| Device Roles | Hardware[4] | Junos OS or Junos OS Evolved (EVO) Software Releases[1] |
|---|---|---|
| | ACX7024<br>ACX7100-32C<br>ACX7100-48L<br>PTX10001<br>PTX10008 | 23.4R2-EVO |
| Border leaf with optimized intersubnet multicast (OISM) | QFX10002-36Q/72Q<br>QFX10002-60C<br>QFX10008<br>QFX10016 | 21.4R2 |
| | QFX5120-32C<br>QFX5120-48T<br>QFX5120-48Y | 22.2R2 |
| | QFX5130-32CD<br>QFX5700 | 22.2R2-EVO |
| | PTX10001<br>PTX10004<br>PTX10008 | 23.2R2-EVO |
| Border leaf with Enhanced OISM | QFX5120-32C<br>QFX5120-48T<br>QFX5120-48Y | 23.4R2 |

**Table 2: Data Center Fabric Reference Design Supported Hardware Summary** ***(Continued)***

| Device Roles | Hardware[4] | Junos OS or Junos OS Evolved (EVO) Software Releases[1] |
|---|---|---|
| | QFX5130-32CD<br>QFX5700 | 23.4R2-EVO |
| Lean super spine | QFX5110<br>QFX5120-32C<br>QFX5120-48Y<br>QFX5200<br>QFX5210-64C<br>QFX5220-32CD<br>QFX5220-128C<br>QFX10002-36Q/72Q/60C<br>QFX10008<br>QFX10016 | 20.2R2 |
| | QFX5120-48YM | 20.4R3 |
| | ACX7100-32C<br>ACX7100-48L<br>PTX10001<br>PTX10004<br>PTX10008<br>QFX5130-32CD<br>QFX5700 | 21.2R2-S1-EVO |
| | ACX7024 | 22.4R2-EVO |

**Table 2: Data Center Fabric Reference Design Supported Hardware Summary** ***(Continued)***

| Device Roles | Hardware[4] | Junos OS or Junos OS Evolved (EVO) Software Releases[1] |
|---|---|---|
| Lean Super Spine with IPv6 underlay | QFX5120-32C<br>QFX5120-48Y<br>QFX10002-36Q/72Q<br>QFX10002-60C<br>QFX10008<br>QFX10016 | 21.2R2-S1 |
| | QFX5120-48YM | 21.4R2 |
| **Collapsed spine** | | |
| Leaf with Virtual Chassis | QFX5100<br>QFX5110<br>QFX5120 | 20.2R2 |

**Table 2: Data Center Fabric Reference Design Supported Hardware Summary *(Continued)***

| Device Roles | Hardware[4] | Junos OS or Junos OS Evolved (EVO) Software Releases[1] |
|---|---|---|
| Collapsed spine | MX204<br><br>MX240, MX480, and MX960 with MPC7E<br><br>MX10003<br><br>MX10008<br><br>QFX5110<br><br>QFX5120-32C<br><br>QFX5120-48T<br><br>QFX5120-48Y<br><br>QFX10002-36Q/72Q/60C<br><br>QFX10008<br><br>QFX10016 | 20.2R2 |
| | QFX5120-48YM | 20.4R3 |
| | QFX5130-32CD<br><br>QFX5700 | 21.2R2-S1-EVO |
| | ACX7100-32C<br><br>ACX7100-48L<br><br>PTX10001<br><br>PTX10004<br><br>PTX10008 | 21.4R2-EVO |
| | ACX7024 | 22.4R2-EVO |

[1] This column includes the initial Junos OS or Junos OS Evolved release train with which we introduce support for the hardware in the reference design. For each initial Junos OS or Junos OS Evolved release

train, we also support the hardware with later releases in the same release train. Although we list the initial Junos OS or Junos OS Evolved release train here, we recommend that you use the most recent hardened releases that are listed on the following page: Data Center Architectures Hardened Release Information for EVPN/VXLAN.

2 While functioning in this role, this hardware does not support centrally routed multicast.

3 While functioning in this role, the QFX10002-60C switch supports multicast traffic. However, this switch supports multicast at a lower scale than the QFX10002-36Q/72Q switches.

4 For multicast support, consult the Pathfinder Feature Explorer application for multicast features and the releases that support the feature on each platform.

5 For devices in the border spine role, set a larger value for Bidirectional Forwarding Detection (BFD) timers in the overlay—4 seconds or higher, with a multiplier of 3. These settings help avoid potential flapping of the spine-to-leaf connection due to BFD control timer expiration.

This table does not include backbone devices that connect the data center to a WAN cloud. Backbone devices provide physical connectivity between data centers and are required for DCI. See "Data Center Interconnect Design and Implementation Using Type 5 Routes " on page 331.

## Interfaces Summary

This section summarizes the interface connections between spine and leaf devices that were validated in this reference design.

It contains the following sections:

### Interfaces Overview

In the validated reference design, spine and leaf devices are interconnected using either an aggregated Ethernet interface that includes two high-speed Ethernet interfaces or a single high-speed Ethernet interface.

The reference design was validated with the following combinations of spine and leaf device interconnections:

- See Table 2 on page 52 for the interconnected platforms tested in spine and leaf device roles as of the indicated Junos OS and Junos OS Evolved hardening releases.

  We use 10-Gbps, 25-Gbps, 40-Gbps, or 100-Gbps interfaces on the supported platforms to interconnect spine and leaf devices. Starting in Junos OS and Junos OS Evolved Release 23.2R2, we also use 400-Gbps interfaces to interconnect leaf and spine devices that support this speed.

- We validated combinations of aggregated Ethernet interfaces containing two 10-Gbps, 25-Gbps, 40-Gbps, 100-Gbps, and 400-Gbps member interfaces between the supported platforms.
- We validated channelized 10-Gbps, 25-Gbps, 40-Gbps, or 100-Gbps interfaces used to interconnect spine and leaf devices as single links or as member links in a 2-member aggregated Ethernet bundle.

### Spine Device Interface Summary

As previously stated, the validated design includes up to 4 spine devices and leaf devices that are interconnected by one or two high-speed Ethernet interfaces.

QFX10008 and QFX10016 switches were used as they can achieve the port density necessary for this reference design. See QFX10008 Hardware Overview or QFX10016 Hardware Overview for information on supported line cards and the number of high-speed Ethernet interfaces supported on these switches.

QFX10002-36Q/72Q, QFX10002-60C, and QFX5120-32C switches, however, do not have the port density to support this reference design at the larger scales but can be deployed as spine devices in smaller scale environments. See QFX10002 Hardware Overview and QFX5120 System Overview for information about the number of high-speed Ethernet interfaces supported on QFX10002-36Q/72Q, QFX10002-60C, and QFX5120-32C switches, respectively.

All channelized spine device interface options are tested and supported in the validated reference design.

### Leaf Device Interface Summary

Each leaf device in the reference design connects to the four spine devices and has the port density to support this reference design.

The number and types of high-speed Ethernet interfaces used as uplink interfaces to spine devices vary by leaf device switch model.

To see which high-speed interfaces are available with each leaf device switch model, see the following documents:

- *QFX5200 Switch Description*
- *QFX5100 Device Hardware Overview*
- *QFX5110 Hardware Overview*
- *QFX5120 System Overview*
- *QFX5210 System Overview*
- *QFX5220 System Overview*

- *QFX10002 Switch Description*
- *QFX10008 System Overview*
- *QFX10016 System Overview*
- MX Series Routers
- *QFX5130 System Overview*
- *QFX5700 System Overview*
- *PTX10001-36MR System Overview*
- *PTX10004 System Overview*
- *PTX10008 System Overview*
- *ACX7024 and ACX7024X System Overview*
- *ACX7100-32C System Overview*
- *ACX7100-48L System Overview*

RELATED DOCUMENTATION

Data Center Fabric Blueprint Architecture Components | 9

## IP Fabric Underlay Network Design and Implementation

**IN THIS SECTION**

For an overview of the supported IP fabric underlay models and components used in these designs, see the IP Fabric Underlay Network section in "Data Center Fabric Blueprint Architecture Components" on page 9.

This section explains how to configure spine and leaf devices in 3-stage and 5-stage IPv4 fabric underlays. For information about how to configure the additional tier of super spine devices in a 5-stage IP fabric underlay, see "Five-Stage IP Fabric Design and Implementation" on page 255. For the steps to configure an IPv6 Fabric design in reference architectures that support that configuration, see "IPv6 Fabric Underlay and Overlay Network Design and Implementation with EBGP" on page 103 instead.

The IP underlay network building block is arranged in a Clos-based fabric topology. The underlay network uses EBGP as the routing protocol in place of a traditional IGP like OSPF. You can use other routing protocols in the underlay protocol in your data center; the usage of those routing protocols is beyond the scope of this document.

Aggregated Ethernet interfaces with MicroBFD are also used in this building block. MicroBFD improves fault detection in an aggregated Ethernet interface by running BFD on individual links of the aggregated Ethernet interface.

Figure 27 on page 82 and Figure 28 on page 83 provide high-level illustrations of a 3-stage and 5-stage IP fabric underlay networks, respectively.

**Figure 27: Three-Stage IP Fabric Underlay Network**

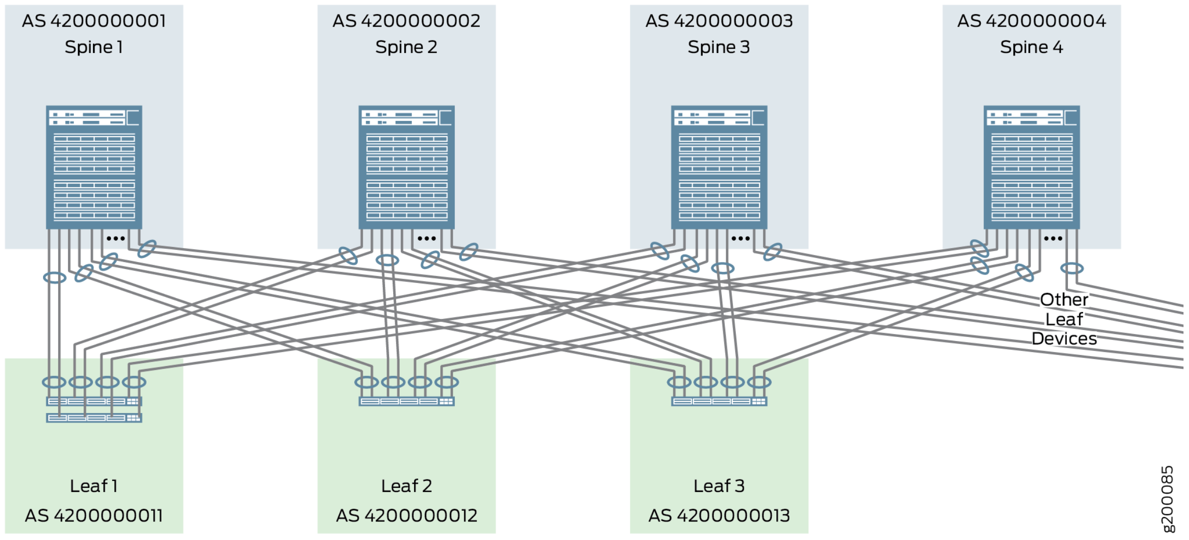
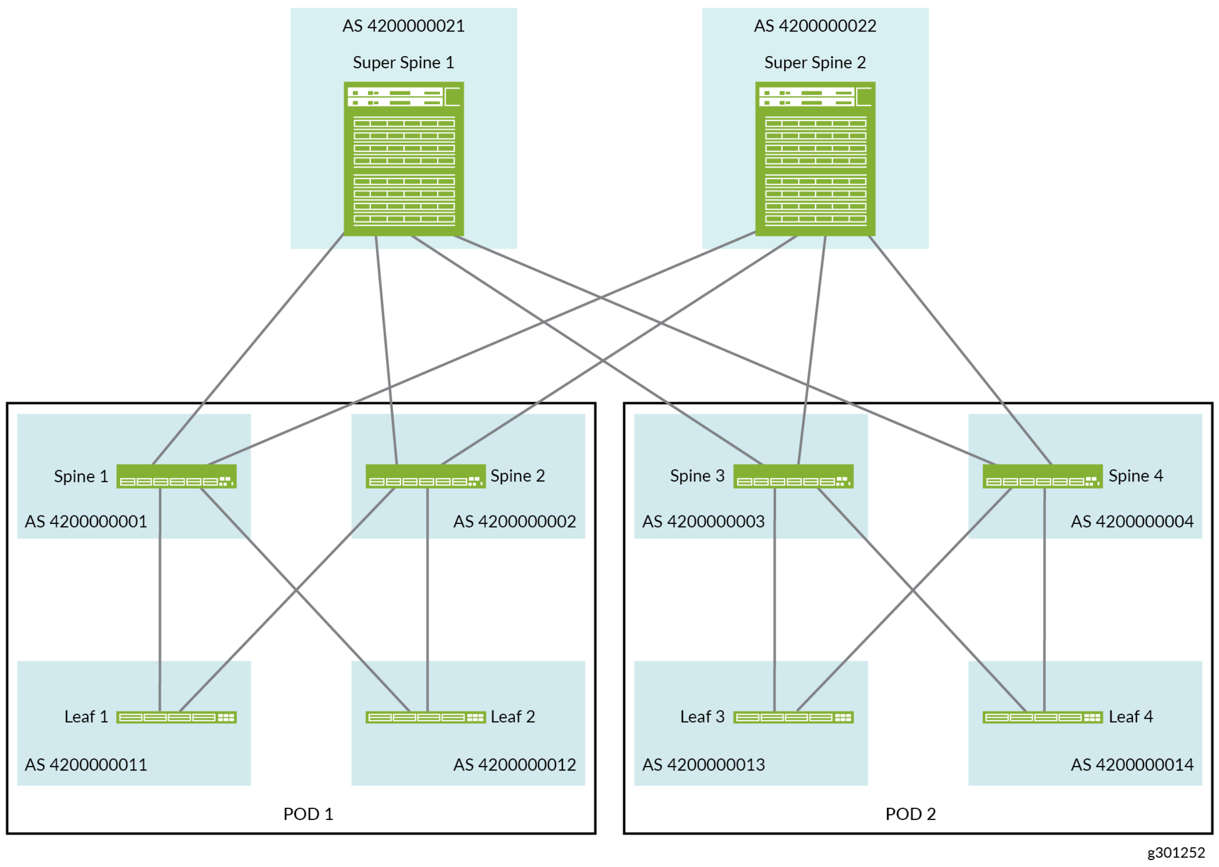


**Figure 28: Five-Stage IP Fabric Underlay Network**



## Configuring the Aggregated Ethernet Interfaces Connecting Spine Devices to Leaf Devices

In this design each spine device is interconnected to each leaf device using a single link or a two-member aggregated Ethernet interface. The decision to use a single link or an aggregated Ethernet interface largely depends on the needs of your network; see "Data Center Fabric Reference Design Overview and Validated Topology" on page 49 for more information on interface requirements.

The majority of IP Fabric topologies do not use aggregated Ethernet interfaces to interconnect spine and leaf devices. You can skip this section if you are connecting your spine and leaf devices using single links.

Use the following instructions to configure the interfaces that interconnect spine and leaf devices as aggregated Ethernet interfaces with two member links. An IPv4 address is assigned to each aggregated Ethernet interface. LACP with a fast periodic interval is also enabled.

Figure 29 on page 84 shows the spine device interfaces that are configured in this procedure:

**Figure 29: Spine 1 Interfaces**

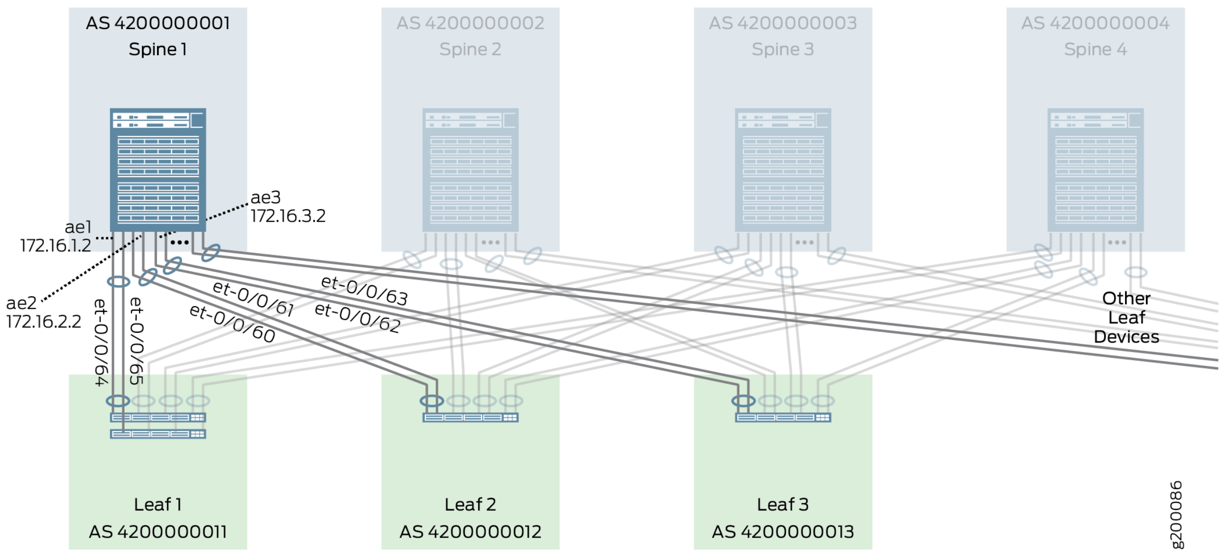
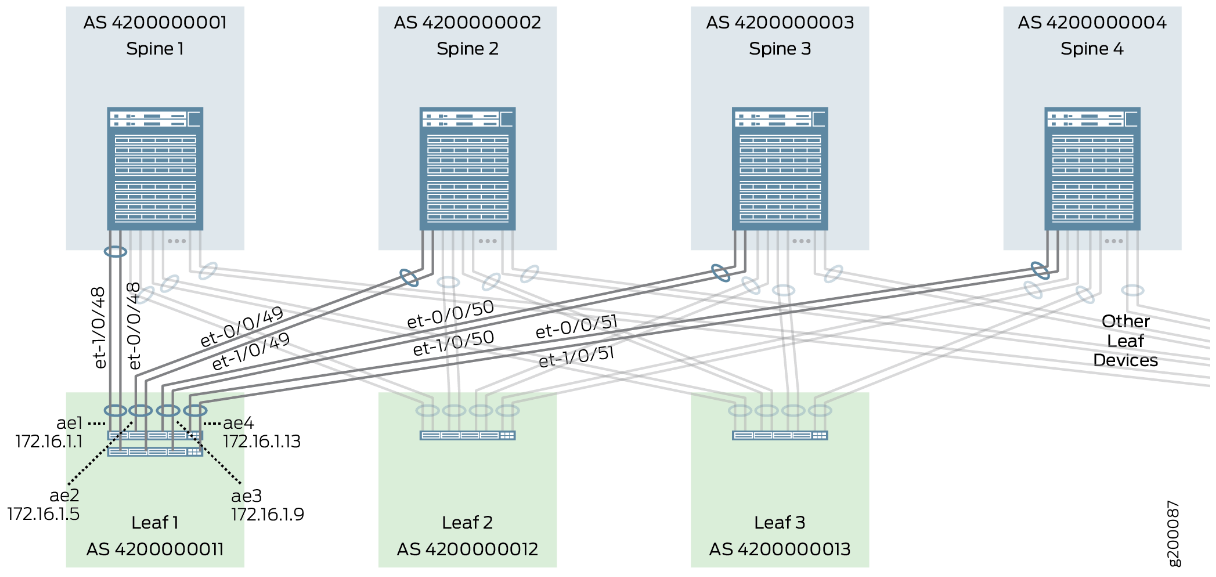


Figure 30 on page 84 shows the leaf device interfaces that are configured in this procedure:

**Figure 30: Leaf 1 Interfaces**



To configure aggregated Ethernet interfaces with fast LACP:

1. Set the maximum number of aggregated Ethernet interfaces permitted on the device.

   We recommend setting this number to the exact number of aggregated Ethernet interfaces on your device, including aggregated Ethernet interfaces that are not used for spine to leaf device connections.

   In this example, the aggregated Ethernet device count value is set at 10 for a leaf device and 100 for a spine device.

   *Leaf Device*:

   ```
   set chassis aggregated-devices ethernet device-count 10
   ```

   *Spine Device*:

   ```
   set chassis aggregated-devices ethernet device-count 100
   ```

2. Create and name the aggregated Ethernet interfaces, and optionally assign a description to each interface.

   This step shows how to create three aggregated Ethernet interfaces on spine 1 and four aggregated ethernet interfaces on leaf 1.

   Repeat this procedure for every aggregated Ethernet interface connecting a spine device to a leaf device.

   *Spine 1*:

   ```
   set interfaces ae1 description "LEAF1"
   set interfaces ae2 description "LEAF2"
   set interfaces ae3 description "LEAF3"
   ```

   *Leaf 1*:

   ```
   set interfaces ae1 description "SPINE1"
   set interfaces ae2 description "SPINE2"
   set interfaces ae3 description "SPINE3"
   set interfaces ae4 description "SPINE4"
   ```

3. Assign interfaces to each aggregated Ethernet interface on your device.

*Spine 1*:

```
set interfaces et-0/0/64 ether-options 802.3ad ae1
set interfaces et-0/0/65 ether-options 802.3ad ae1
set interfaces et-0/0/60 ether-options 802.3ad ae2
set interfaces et-0/0/61 ether-options 802.3ad ae2
set interfaces et-0/0/62 ether-options 802.3ad ae3
set interfaces et-0/0/63 ether-options 802.3ad ae3
```

*Leaf 1*:

```
set interfaces et-0/0/48 ether-options 802.3ad ae1
set interfaces et-1/0/48 ether-options 802.3ad ae1
set interfaces et-0/0/49 ether-options 802.3ad ae2
set interfaces et-1/0/49 ether-options 802.3ad ae2
set interfaces et-0/0/50 ether-options 802.3ad ae3
set interfaces et-1/0/50 ether-options 802.3ad ae3
set interfaces et-0/0/51 ether-options 802.3ad ae4
set interfaces et-1/0/51 ether-options 802.3ad ae4
```

4. Assign an IP address to each aggregated Ethernet interface on the device.

   *Spine 1*:

```
set interfaces ae1 unit 0 family inet address 172.16.1.2/30
set interfaces ae2 unit 0 family inet address 172.16.2.2/30
set interfaces ae3 unit 0 family inet address 172.16.3.2/30
```

   *Leaf 1*:

```
set interfaces ae1 unit 0 family inet address 172.16.1.1/30
set interfaces ae2 unit 0 family inet address 172.16.1.5/30
set interfaces ae3 unit 0 family inet address 172.16.1.9/30
set interfaces ae4 unit 0 family inet address 172.16.1.13/30
```

5. Enable fast LACP on every aggregated Ethernet interface on the device.

   LACP is enabled using the fast periodic interval, which configures LACP to send a packet every second.

*Spine 1*:

```
set interfaces ae1 aggregated-ether-options lacp active
set interfaces ae1 aggregated-ether-options lacp periodic fast
set interfaces ae2 aggregated-ether-options lacp active
set interfaces ae2 aggregated-ether-options lacp periodic fast
set interfaces ae3 aggregated-ether-options lacp active
set interfaces ae3 aggregated-ether-options lacp periodic fast
```

*Leaf 1*:

```
set interfaces ae1 aggregated-ether-options lacp active
set interfaces ae1 aggregated-ether-options lacp periodic fast
...
...
set interfaces ae4 aggregated-ether-options lacp active
set interfaces ae4 aggregated-ether-options lacp periodic fast
```

6. After the configuration is committed, confirm that the aggregated Ethernet interfaces are enabled, that the physical links are up, and that packets are being transmitted if traffic has been sent.

   The output below provides this confirmation information for `ae1` on Spine 1.

```
user@spine-1> show interfaces ae1
Physical interface: ae1, Enabled, Physical link is Up
 (some output removed for brevity)

  Logical interface ae1.0 (Index 549) (SNMP ifIndex 541)
    Flags: Up SNMP-Traps 0x4004000 Encapsulation: ENET2
    Statistics        Packets        pps         Bytes          bps
    Bundle:
        Input :        609303          0      57831130            0
        Output:       7063505          0    4664278858            0

(some output removed for brevity)
```

7. Confirm the LACP receive state is `Current` and that the transmit state is `Fast` for each link in each aggregated Ethernet interface bundle.

The output below provides LACP status for interface ae1.

```
user@spine-1> show lacp interfaces ae1
Aggregated interface: ae1
...(some output removed for brevity)
    LACP protocol:        Receive State  Transmit State          Mux State
      et-0/0/0                  Current   Fast periodic Collecting distributing
      et-0/0/1                  Current   Fast periodic Collecting distributing
(additional output removed for brevity)
```

8. Repeat this procedure for every device in your topology.

   The guide presumes that spine and leaf devices are interconnected by two-member aggregated Ethernet interfaces in other sections. When you are configuring or monitoring a single link instead of an aggregated Ethernet link, substitute the physical interface name of the single link interface in place of the aggregated Ethernet interface name.

## Configuring an IP Address for an Individual Link

This section covers the procedure to add an IP address to a single link interface connecting a spine or leaf device. The process for adding an IP address to an aggregated Ethernet interface is covered in "Configuring the Aggregated Ethernet Interfaces Connecting Spine Devices to Leaf Devices" on page 83.

To add an IP address to a single link interface:

1. Assign an IP address to each interface.

   *Spine 1 Example*:

```
set interfaces et-0/0/64 unit 0 family inet address 172.16.1.2/30
set interfaces et-0/0/65 unit 0 family inet address 172.16.2.2/30
set interfaces et-0/0/66 unit 0 family inet address 172.16.3.2/30
```

   *Leaf 1 Example*:

```
set interfaces et-0/0/48 unit 0 family inet address 172.16.1.1/30
set interfaces et-0/0/49 unit 0 family inet address 172.16.1.5/30
set interfaces et-0/0/50 unit 0 family inet address 172.16.1.9/30
set interfaces et-0/0/51 unit 0 family inet address 172.16.1.13/30
```

2. After the interface configuration is committed, confirm that the interfaces are enabled, that the physical links are up, and that packets are being transmitted if traffic has been sent.

   The output below provides this confirmation information for `et-0/0/64` on Spine 1.

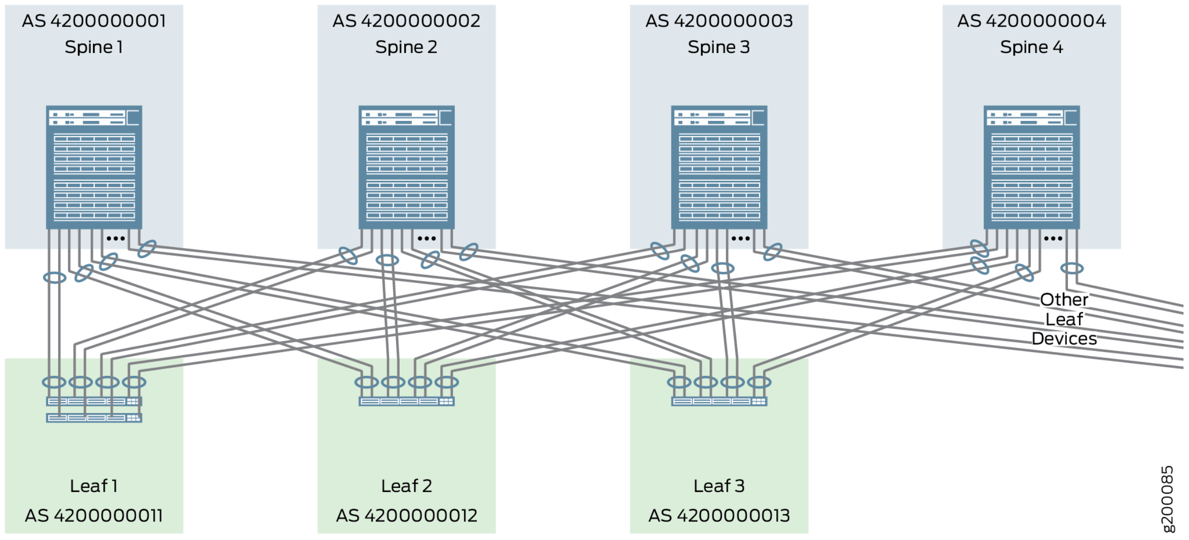
```
user@spine-1> show interfaces et-0/0/64
Physical interface: et-0/0/64, Enabled, Physical link is Up
 (some output removed for brevity)

  Logical interface ae1.0 (Index 549) (SNMP ifIndex 541)
    Flags: Up SNMP-Traps 0x4004000 Encapsulation: ENET2
    Statistics        Packets        pps         Bytes          bps
    Bundle:
        Input :        609303          0      57831130            0
        Output:       7063505          0    4664278858            0

(some output removed for brevity)
```

## Enabling EBGP as the Routing Protocol in the Underlay Network

In this design, EBGP is the routing protocol of the underlay network and each device in the IP fabric is assigned a unique 32-bit autonomous system number (ASN). The underlay routing configuration ensures that all devices in the underlay IP fabric are reliably reachable from one another. Reachability between VTEP across the underlay IP Fabric is also required to support overlay networking with VXLAN.

Figure 31 on page 90 shows the EBGP configuration of the underlay network.

**Figure 31: EBGP Underlay Network Overview**



To enable EBGP as the routing protocol for the underlay network on a device:

1. Create and name the BGP peer group. EBGP is enabled as part of this step.

   The underlay EBGP group is named `UNDERLAY` in this design.

   *Spine or Leaf Device*:

   ```
   set protocols bgp group UNDERLAY type external
   ```

2. Configure the ASN for each device in the underlay.

   Recall that in this design, every device is assigned a unique ASN in the underlay network. The ASN for EBGP in the underlay network is configured at the BGP peer group level using the `local-as` statement because the system ASN setting is used for MP-IBGP signaling in the overlay network.

   The examples below show how to configure the ASN for EBGP for Spine 1 and Leaf 1.

   *Spine 1*:

   ```
   set protocols bgp group UNDERLAY local-as 4200000001
   ```

   *Leaf 1*:

   ```
   set protocols bgp group UNDERLAY local-as 4200000011
   ```

3. Configure BGP peers by specifying the ASN of each BGP peer in the underlay network on each spine and leaf device.

   In this design, for a spine device, every leaf device is a BGP peer, and for a leaf device, every spine device is a BGP peer.

   The example below demonstrates how to configure the peer ASN in this design.

   *Spine 1*:

   ```
   set protocols bgp group UNDERLAY neighbor 172.16.1.1 peer-as 4200000011
   set protocols bgp group UNDERLAY neighbor 172.16.2.1 peer-as 4200000012
   set protocols bgp group UNDERLAY neighbor 172.16.3.1 peer-as 4200000013
   ```

   *Leaf 1*:

   ```
   set protocols bgp group UNDERLAY neighbor 172.16.1.2 peer-as 4200000001
   set protocols bgp group UNDERLAY neighbor 172.16.1.6 peer-as 4200000002
   set protocols bgp group UNDERLAY neighbor 172.16.1.10 peer-as 4200000003
   set protocols bgp group UNDERLAY neighbor 172.16.1.14 peer-as 4200000004
   ```

4. Set the BGP hold time. The BGP hold time is the length of time, in seconds, that a peer waits for a BGP message—typically a keepalive, update, or notification message—before closing a BGP connection.

   Shorter BGP hold time values guard against BGP sessions staying open for unnecessarily long times in scenarios where problems occur, such as keepalives not being sent. A longer BGP hold time ensures BGP sessions remain active even during problem periods.

   The BGP hold time is configured at 10 seconds for every device in this design.

   *Spine or Leaf Device*:

   ```
   set protocols bgp group UNDERLAY hold-time 10
   ```

5. Configure EBGP to signal the unicast address family for the underlay BGP peer group.

   *Spine or Leaf Device*:

   ```
   set protocols bgp group UNDERLAY family inet unicast
   ```

6. Configure an export routing policy that advertises the IP address of the loopback interface to EBGP peering devices.

This export routing policy is used to make the IP address of the loopback interface reachable from all devices in the IP Fabric. Loopback IP address reachability is required to allow leaf and spine device peering using MP-IBGP in the overlay network. IBGP peering in the overlay network must be established to allow devices in the fabric to share EVPN routes. See "Configure IBGP for the Overlay" on page 96.

The route filter IP address in this step—192.168.1.10—is the loopback address of the leaf device.

*Leaf 1*:

```
set policy-options policy-statement BGP_LOOPBACK0 term TERM1 from protocol direct
set policy-options policy-statement BGP_LOOPBACK0 term TERM1 from route-filter
192.168.1.10/32 exact
set policy-options policy-statement BGP_LOOPBACK0 term TERM1 then accept
set protocols bgp group UNDERLAY export BGP_LOOPBACK0
```

7. After committing the configuration, enter the **show bgp summary** command on each device to confirm that the BGP state is established and that traffic paths are active.

   Issue the **show bgp summary** command on Spine 1 to verify EBGP status.

```
user@spine-1> show bgp summary
Groups: 1 Peers: 96 Down peers: 0
Table          Tot Paths  Act Paths Suppressed    History Damp State    Pending
inet.0           834        232          0          0          0          0

Peer                     AS      InPkt     OutPkt    OutQ   Flaps Last Up/Dwn State|#Active/
Received/Accepted/Damped...
172.16.1.1         4200000011      13530      13750       0       0     10:21:37 Establ inet.0:
204/209/209/0
172.16.2.1         4200000012      13532      13821       0       0     10:21:33 Establ inet.0:
27/209/209/0
(additional output removed for brevity)
```

8. Repeat this procedure for every spine and leaf device in your topology.

## Enabling Load Balancing

ECMP load balancing allows traffic to be sent to the same destination over multiple equal cost paths. Load balancing must be enabled on all spine and leaf devices to ensure that traffic is sent over all available paths provided by the IP Fabric.

Traffic is load balanced per Layer 4 flow on Junos devices. The ECMP algorithm load balances each traffic flow over one of the multiple paths, and all traffic for that flow is transmitted using the selected link.

To enable ECMP-based load balancing on a device:

1. Enable multipath with the multiple AS option in BGP on all devices in the IP Fabric.

   EBGP, by default, selects one best path for each prefix and installs that route in the forwarding table. When BGP multipath is enabled, all equal-cost paths to a given destination are installed into the forwarding table. The `multiple-as` option enables load balancing between EBGP neighbors in different autonomous systems.

   *All Spine and Leaf Devices*:

   ```
   set protocols bgp group UNDERLAY multipath multiple-as
   ```

2. Create a policy statement that enables per-packet load balancing.

   *All Spine and Leaf Devices*:

   ```
   set policy-options policy-statement PFE-ECMP then load-balance per-packet
   ```

3. Export the policy statement to the forwarding table.

   *All Spine and Leaf Devices*:

   ```
   set routing-options forwarding-table export PFE-ECMP
   ```

## Configuring Micro Bidirectional Forwarding Detection on Member Links in Aggregated Ethernet Interfaces

BFD is a simple bidirectional fault detection protocol that verifies bidirectional connectivity between directly-connected devices by periodically and rapidly sending a simple hello packet over the link or links that interconnect the devices. BFD can detect and communicate link faults in sub-second time frames to allow the control-plane software to quickly switch to an alternate path.

MicroBFD allows BFD to run on individual member links in an aggregated Ethernet interface.

In this design, microBFD is supported on connections between QFX10002-36Q/72Q, QFX10008, and QFX10016 switches.

To enable microBFD:

1. Set the minimum interval for BFD.

   The minimum interval is the amount of time in milliseconds between hello packets BFD sends to a directly connected neighbor. It is also the basis for the amount of time that BFD waits to receive a response to hello packets before directing traffic destined to that link to other member links in the aggregated Ethernet interface. The actual wait time factors in a multiplier that represents the number of hello packet intervals that go by without a response before BFD declares the peer to be down. You can customize the multiplier value, or by default BFD waits 3 times the minimum interval before redirecting traffic away from the unresponsive neighbor.

   This design sets the minimum interval to 100 milliseconds on all member links and uses the default multiplier value (3).

   If the minimum interval is set to 100, for example, BFD sends a hello packet every 100 milliseconds and starts directing traffic to other member links in an aggregated Ethernet interface if the hello packet does not receive a response in 300 milliseconds.

   Set the minimum interval to a lower value if you want to ensure traffic is quickly redirected after a link failure. Set the minimum interval to a higher value if you want to minimize the chance that BFD accidentally misidentifies an active link as a down link.

   The output below shows how to configure the minimum interval to 100 milliseconds on aggregated Ethernet interface 1.

   *Spine and Leaf Device*:

   ```
   set interfaces ae1 aggregated-ether-options bfd-liveness-detection minimum-interval 100
   ```

2. Configure the IP address used by MicroBFD over the aggregated Ethernet bundle. For bridged AE interfaces, this address should be the IP address used by MicroBFD over the aggregated Ethernet bundle.

   The output below show how to assign the loopback IP address to ae1 on the leaf device.

   *Spine and Leaf Device*:

   ```
   set interfaces ae1 aggregated-ether-options bfd-liveness-detection local-address 192.168.1.10
   ```

3. Specify the loopback IP address of the neighbor device on the other end of the aggregated Ethernet interface.

   The output below is from aggregated Ethernet interface 1 on leaf device 1, which is used to connect to spine device 1. Spine device 1 uses 192.168.0.1 as it’s loopback IP address.

*Spine and Leaf Device*:

```
set interfaces ae1 aggregated-ether-options bfd-liveness-detection neighbor 192.168.0.1
```

4. Specify the minimum number of links that have to remain up before all links in the aggregated Ethernet interface stop sending and receiving traffic.

   Setting the minimum links to 1 ensures the aggregated Ethernet bundle always transmits traffic unless all links in the aggregated Ethernet interface are unable to forward or receive traffic. Consider setting the minimum links value to a higher number if you feel performance could be significantly degraded in your network if a number of member links in your aggregated Ethernet interface stopped forwarding traffic.

   All devices in the design set the minimum links value to 1.

   *Spine and Leaf Device*:

   ```
   set interfaces ae1 aggregated-ether-options minimum-links 1
   ```

   Repeat the above steps for each aggregated Ethernet interface in your topology to enable microBFD.

5. To verify BFD sessions after enabling microBFD, enter the **show bfd session** command:

   *Leaf 10 Example*:

   ```
   user@leaf10> show bfd session
                                                  Detect   Transmit
   Address                  State     Interface      Time     Interval  Multiplier
   192.168.0.1              Up        et-0/0/11      0.300     0.100        3
   192.168.0.1              Up        et-0/0/29      0.300     0.100        3
   ```

## IP Fabric Underlay Network — Release History

Table 3 on page 96 provides a history of all of the features in this section and their support within this reference design.

**Table 3: IP Fabric Underlay Network Release History**

| Release | Description |
|---|---|
| 19.1R2 | QFX10002-60C and QFX5120-32C switches running Junos OS Release 19.1R2 and later releases in the same release train also support all features documented in this section except the following:<br>• MicroBFD, which is supported on QFX10002-36Q/72Q, QFX10008, and QFX10016 switches only. |
| 18.4R2 | QFX5120-48Y switches running Junos OS Release 18.4R2 and later releases in the same release train support all features documented in this section except MicroBFD. |
| 18.1R3-S3 | QFX5110 switches running Junos OS Release 18.1R3-S3 and later releases in the same release train support all features documented in this section except MicroBFD. |
| 17.3R3-S1 | All devices in the reference design that support Junos OS Release 17.3R3-S1 and later releases in the same release train also support all features documented in this section. The following is an exception:<br>• MicroBFD, which is supported on QFX10002-36Q/72Q, QFX10008, and QFX10016 switches only. |

## RELATED DOCUMENTATION

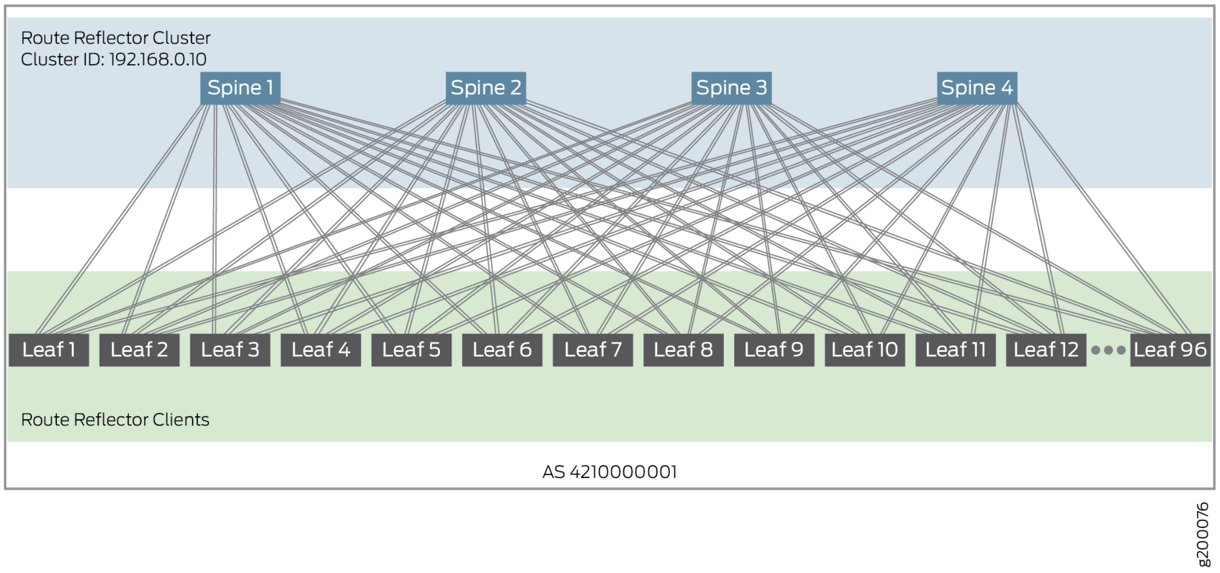
Software as a Service

*Understanding Independent Micro BFD Sessions for LAG*

# Configure IBGP for the Overlay

For a control-plane driven overlay, there must be a signalling path between the VXLAN virtual tunnel endpoint (VTEP) devices. In this reference design with an IPv4 Fabric underlay, all overlay types use IBGP with Multiprotocol BGP (MP-IBGP) to maintain the signalling path between the VTEPs within an autonomous system. The spine devices act as a route reflector cluster, and the leaf devices are route reflector clients, as shown in Figure 32 on page 97.

**Figure 32: IBGP Route Reflector Cluster**



To configure an EVPN-VXLAN data center fabric architecture with an IPv6 Fabric, see "IPv6 Fabric Underlay and Overlay Network Design and Implementation with EBGP" on page 103 instead of this procedure. In an IPv6 Fabric configuration, we use EBGP and IPv6 for underlay connectivity, as well as EBGP and IPv6 for peering and EVPN signalling in the overlay. With an IPv6 Fabric, the VTEPs encapsulate the VXLAN packets with an IPv6 outer header and tunnel the packets using IPv6. You can use either an IPv4 Fabric or an IPv6 Fabric in your data center architecture. You can't mix IPv4 Fabric and IPv6 Fabric elements in the same architecture.

To configure IBGP for the overlay peering in an IPv4 Fabric, perform the following:

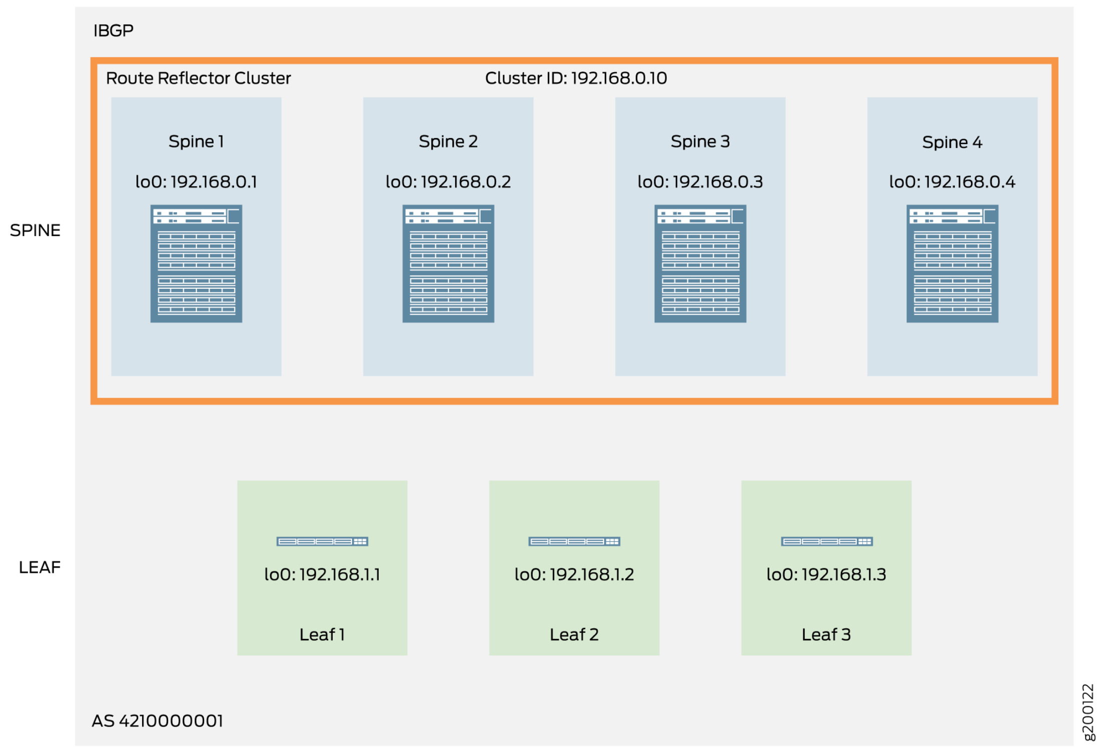
1. Configure an AS number for overlay IBGP. All leaf and spine devices participating in the overlay use the same AS number. In this example, the AS number is private AS 4210000001.

   *Spine and Leaf Devices*:

   ```
   set routing-options autonomous-system 4210000001
   ```

2. Configure IBGP using EVPN signaling on each spine device to peer with every leaf device (Leaf 1 through Leaf 96). Also, form the route reflector cluster (cluster ID 192.168.0.10) and configure equal cost multipath (ECMP) for BGP. The configuration included here belongs to Spine 1, as shown in Figure 33 on page 98.

**Figure 33: IBGP – Spine Device**



**TIP**: By default, BGP selects only one best path when there are multiple, equal-cost BGP paths to a destination. When you enable BGP multipath by including the `multipath` statement at the `[edit protocols bgp group *group-name*]` hierarchy level, the device installs all of the equal-cost BGP paths into the forwarding table. This feature helps load balance the traffic across multiple paths.

*Spine 1*:

```
set protocols bgp group OVERLAY type internal
set protocols bgp group OVERLAY local-address 192.168.0.1
set protocols bgp group OVERLAY family evpn signaling
set protocols bgp group OVERLAY cluster 192.168.0.10
set protocols bgp group OVERLAY multipath
set protocols bgp group OVERLAY neighbor 192.168.1.1
...
set protocols bgp group OVERLAY neighbor 192.168.1.96
```

3. Configure IBGP on the spine devices to peer with all the other spine devices acting as route reflectors. This step completes the full mesh peering topology required to form a route reflector cluster.

   *Spine 1*:

   ```
   set protocols bgp group  OVERLAY_RR_MESH type internal
   set protocols bgp group  OVERLAY_RR_MESH local-address 192.168.0.1
   set protocols bgp group  OVERLAY_RR_MESH family evpn signaling
   set protocols bgp group  OVERLAY_RR_MESH neighbor 192.168.0.2
   set protocols bgp group  OVERLAY_RR_MESH neighbor 192.168.0.3
   set protocols bgp group  OVERLAY_RR_MESH neighbor 192.168.0.4
   ```

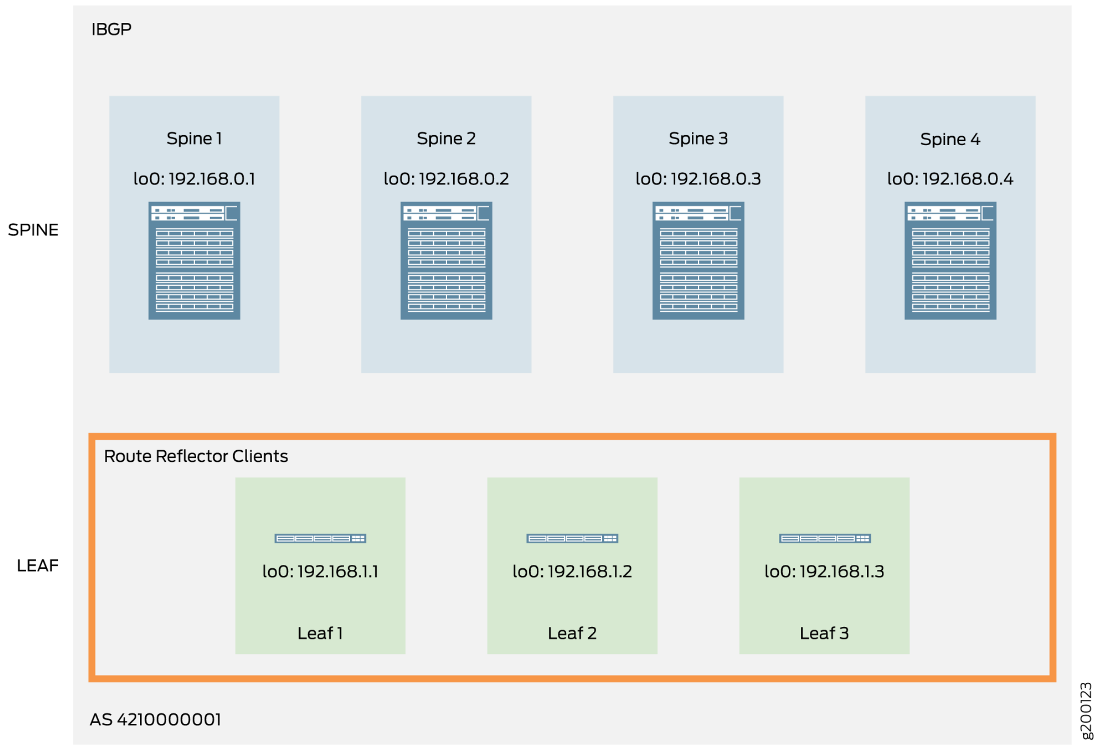
4. Configure BFD on all BGP groups on the spine devices to enable rapid detection of failures and reconvergence.

   *Spine 1*:

   ```
   set protocols bgp group OVERLAY bfd-liveness-detection minimum-interval 350
   set protocols bgp group OVERLAY bfd-liveness-detection multiplier 3
   set protocols bgp group OVERLAY bfd-liveness-detection session-mode automatic
   set protocols bgp group  OVERLAY_RR_MESH bfd-liveness-detection minimum-interval 350
   set protocols bgp group  OVERLAY_RR_MESH bfd-liveness-detection multiplier 3
   set protocols bgp group  OVERLAY_RR_MESH bfd-liveness-detection session-mode automatic
   ```

5. Configure IBGP with EVPN signaling from each leaf device (route reflector client) to each spine device (route reflector cluster). The configuration included here belongs to Leaf 1, as shown in Figure 34 on page 100.

**Figure 34: IBGP – Leaf Device**



*Leaf 1*:

```
set protocols bgp group OVERLAY type internal
set protocols bgp group OVERLAY local-address 192.168.1.1
set protocols bgp group OVERLAY family evpn signaling
set protocols bgp group OVERLAY neighbor 192.168.0.1
set protocols bgp group OVERLAY neighbor 192.168.0.2
set protocols bgp group OVERLAY neighbor 192.168.0.3
set protocols bgp group OVERLAY neighbor 192.168.0.4
```

6. Configure BFD on the leaf devices to enable rapid detection of failures and reconvergence.

**NOTE**: QFX5100 switches only support BFD liveness detection minimum intervals of 1 second or longer. The configuration here has a minimum interval of 350 ms, which is supported on devices other than QFX5100 switches.

*Leaf 1*:

```
set protocols bgp group OVERLAY bfd-liveness-detection minimum-interval 350
set protocols bgp group OVERLAY bfd-liveness-detection multiplier 3
set protocols bgp group OVERLAY bfd-liveness-detection session-mode automatic
```

7. Verify that IBGP is functional on the spine devices.

```
user@spine-1> show bgp summary
Groups: 5 Peers: 221 Down peers: 0
Table          Tot Paths  Act Paths Suppressed    History Damp State    Pending
inet.0
                    9711        182          0          0          0          0
inet6.0
                       0          0          0          0          0          0
bgp.evpn.0
                   31520      31520          0          0          0          0
Peer                     AS      InPkt     OutPkt    OutQ   Flaps Last Up/Dwn State|#Active/
Received/Accepted/Damped...
192.168.0.2       421000001      28724      31106       0       0    22:40:41 Establ
  bgp.evpn.0: 8227/8227/8227/0
 default-switch.evpn.0: 54/54/54/0...

192.168.1.96      421000001       4831      73047       0       0    22:43:41 Establ
  bgp.evpn.0: 1549/1549/1549/0
  default-switch.evpn.0: 11/11/11/0
  __default_evpn__.evpn.0: 1471/1471/1471/0
---(more)---
```

8. Verify that BFD is operational on the spine devices.

```
user@spine-1> show bfd session
                                                  Detect   Transmit
Address                  State     Interface      Time     Interval  Multiplier
192.168.0.2              Up                       1.050     0.350        3
192.168.0.3              Up                       1.050     0.350        3
192.168.0.4              Up                       1.050     0.350        3
192.168.1.1              Up                       1.050     0.350        3
...
192.168.1.96             Up                       1.050     0.350        3
```

9. Verify that IBGP is operational on the leaf devices.

```
user@leaf-1> show bgp summary
Groups: 2 Peers: 8 Down peers: 0
Table          Tot Paths  Act Paths Suppressed    History Damp State    Pending
inet.0
                     834        233          0          0          0          0
bgp.evpn.0
                    3193        833          0          0          0          0
Peer                     AS      InPkt     OutPkt    OutQ   Flaps Last Up/Dwn State|#Active/
Received/Accepted/Damped...
## IBGP Overlay
    192.168.0.1      4210000001       9371        596       0       2     4:17:03 Establ
  bgp.evpn.0: 706/829/829/0
  default-switch.evpn.0: 701/824/824/0
  __default_evpn__.evpn.0: 5/5/5/0
192.168.0.2      4210000001      10175        579       0       2     4:16:35 Establ
  bgp.evpn.0: 43/834/834/0
  default-switch.evpn.0: 43/829/829/0
  __default_evpn__.evpn.0: 0/5/5/0
192.168.0.3      4210000001      10463        621       0       2     4:34:55 Establ
  bgp.evpn.0: 43/834/834/0
  default-switch.evpn.0: 43/829/829/0
  __default_evpn__.evpn.0: 0/5/5/0
192.168.0.4      4210000001       8250        463       0       1     3:12:47 Establ
  bgp.evpn.0: 41/696/696/0
  default-switch.evpn.0: 41/691/691/0
  __default_evpn__.evpn.0: 0/5/5/0
```

10. Verify that BFD is operational on the leaf devices.

```
user@leaf-10> show bfd session
                                                  Detect   Transmit
Address                  State     Interface      Time     Interval  Multiplier
192.168.0.1              Up                       1.050     0.350        3
192.168.0.2              Up                       1.050     0.350        3
192.168.0.3              Up                       1.050     0.350        3
192.168.0.4              Up                       1.050     0.350        3
```

RELATED DOCUMENTATION

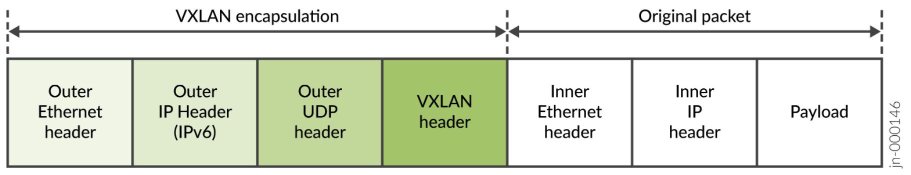
Understanding BGP Route Reflectors

# IPv6 Fabric Underlay and Overlay Network Design and Implementation with EBGP

IN THIS SECTION

Most use cases in this guide are based on an IP Fabric that uses IPv4 and EBGP for underlay connectivity with IBGP overlay peering. On supporting platforms, starting in Junos OS Release 21.2R2-S1 and 21.4R1, you can alternatively use an IPv6 Fabric infrastructure. With an IPv6 Fabric, the VXLAN virtual tunnel endpoints (VTEPs) encapsulate the VXLAN header with an IPv6 outer header and tunnel the packets using IPv6. The workload packets with the payload can use either IPv4 or IPv6. See Figure 35 on page 103.

**Figure 35: IPv6 Fabric VXLAN Packet Encapsulation**



An IPv6 Fabric uses IPv6 addressing, IPv6 and EBGP for underlay connectivity, and IPv6 and EBGP for overlay peering. You can't mix IPv4 and IPv6 underlays and overlay peering in the same fabric.

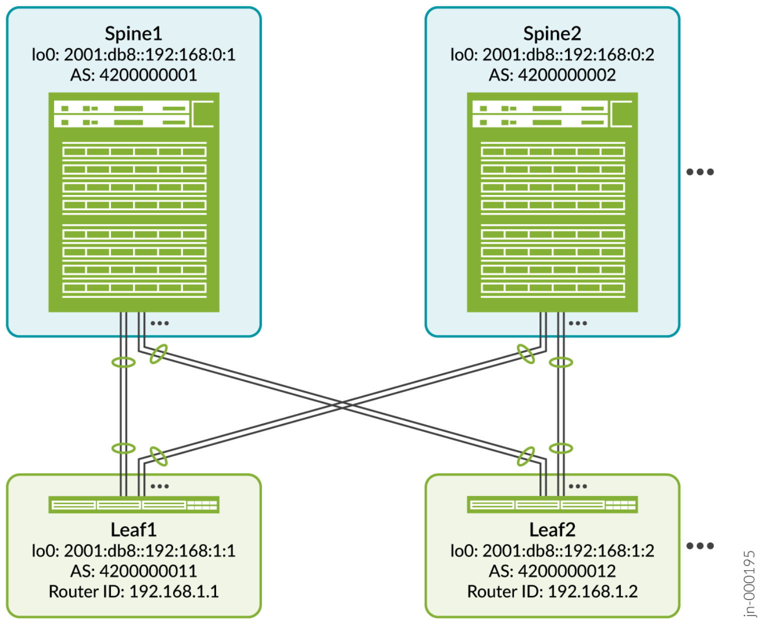
This section describes how to configure the IPv6 Fabric design. In this environment, you can take advantage of the expanded addressing capabilities and efficient packet processing that the IPv6 protocol offers.

We have qualified this IPv6 Fabric in our reference architectures with:

- The following routing and bridging overlay designs:
  - Bridged overlay
  - Edge-routed bridging (ERB) overlay
- EVPN instances configured using MAC-VRF routing instances only.

Figure 36 on page 104 shows a high-level representative view of the spine and leaf devices in an IPv6 Fabric.

**Figure 36: Basic Spine and Leaf Fabric with an IPv6 Fabric**



The topology can be the same or similar to the supported topologies with an IPv4 Fabric.

The main differences in how you configure an IPv6 Fabric instead of an IPv4 Fabric include:

- You configure IPv6 interfaces to interconnect the devices.
- You assign an IPv6 address to the loopback interface on the devices serving as VTEPs.
- In the EVPN routing instance, you set the VTEP source interface as the device’s loopback IPv6 address, rather than an IPv4 address.

- You configure the underlay EBGP peering between the IPv6 interface addresses interconnecting the devices. You configure the overlay EBGP peering between the device IPv6 loopback addresses.

See "Data Center EVPN-VXLAN Fabric Reference Designs—Supported Hardware Summary" on page 51 for the initial hardened release in which a platform supports an IPv6 fabric design, based on the type of overlay architecture and the role the device serves in the fabric. Look for the table rows that state the device role "with IPv6 underlay".

See EVPN-VXLAN with an IPv6 Underlay in the EVPN User Guide for an overview of EVPN-VXLAN feature support and limitations with an IPv6 Fabric.

For an overview of the supported IP fabric underlay and overlay models and components used in our reference architecture designs, see "Data Center Fabric Blueprint Architecture Components" on page 9.

## Configure Interfaces and EBGP as the Routing Protocol in the IPv6 Fabric Underlay

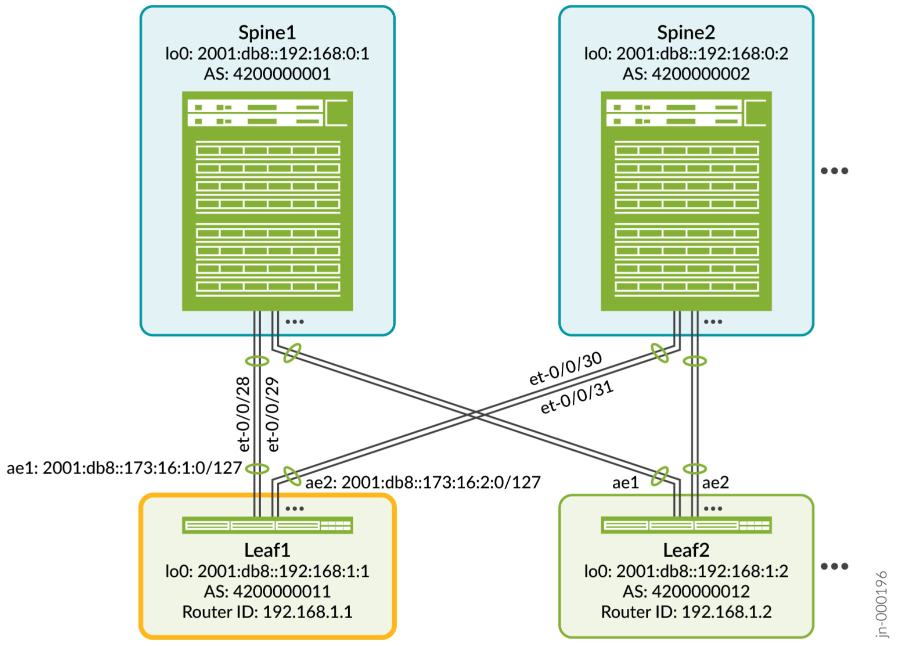
In this design (similar to the IPv4 Fabric in "IP Fabric Underlay Network Design and Implementation" on page 80), you interconnect the spine and leaf devices using aggregated Ethernet interfaces with two member links. (You can alternatively use a single link, or more than two member links in an aggregated Ethernet bundle, for each spine and leaf connection.)

This procedure shows how to configure the interfaces on the leaf side toward the spines, and enable EBGP with IPv6 as the underlay routing protocol on the leaf device.

**NOTE**: Although this procedure doesn't show the spine side configuration, you configure the interconnecting interfaces and the EBGP underlay on the spine side in the same way as you do on the leaf device.

Figure 37 on page 106 shows the interfaces on leaf device Leaf1 that you configure in this procedure.

**Figure 37: Leaf1 Interfaces and IPv6 Addressing with EBGP for Spine and Leaf Connectivity**



To configure aggregated Ethernet interfaces and EBGP in the underlay with IPv6 on Leaf1:

1. Set the maximum number of aggregated Ethernet interfaces permitted on the device.

   We recommend setting this number to the exact number of aggregated Ethernet interfaces on your device, including aggregated Ethernet interfaces that are not for the spine to leaf device connections here. In this scaled-down example, we set the count to 10. If you have more spine and leaf devices or employ aggregated Ethernet interfaces for other purposes, set the appropriate number for your network.

```
set chassis aggregated-devices ethernet device-count 10
```

2. Create the aggregated Ethernet interfaces toward the spine devices, optionally assigning a description to each interface.

```
set interfaces ae1 description “To Spine1”
set interfaces ae2 description “To Spine2”
```

3. Assign interfaces to each aggregated Ethernet interface.

In this case, we show creating aggregated Ethernet interfaces with two member links each. In this step, you also specify the minimum number of links (one) that have to remain up before all links in the aggregated Ethernet interface stop sending and receiving traffic.

```
set interfaces ae1 aggregated-ether-options minimum-links 1
set interfaces et-0/0/28 ether-options 802.3ad ae1
set interfaces et-0/0/29 ether-options 802.3ad ae1
set interfaces ae2 aggregated-ether-options minimum-links 1
set interfaces et-0/0/30 ether-options 802.3ad ae2
set interfaces et-0/0/31 ether-options 802.3ad ae2
```

4. Assign an IPv6 address to each aggregated Ethernet interface.

   In this step, you also specify the interface MTU size. You set two MTU values for each aggregated Ethernet interface, one for the physical interface and one for the IPv6 logical interface. We configure a higher MTU on the physical interface to account for VXLAN encapsulation.

```
set interfaces ae1 mtu 9192
set interfaces ae1 unit 0 family inet6 mtu 9000
set interfaces ae1 unit 0 family inet6 address 2001:db8::173:16:1:0/127
set interfaces ae2 mtu 9192
set interfaces ae2 unit 0 family inet6 mtu 9000
set interfaces ae2 unit 0 family inet6 address 2001:db8::173:16:2:0/127
```

5. Enable fast LACP on the aggregated Ethernet interfaces.

   You enable LACP with the `fast` periodic interval, which configures LACP to send a packet every second.

```
set interfaces ae1 aggregated-ether-options lacp active
set interfaces ae1 aggregated-ether-options lacp periodic fast
set interfaces ae2 aggregated-ether-options lacp active
set interfaces ae2 aggregated-ether-options lacp periodic fast
```

6. Configure an IPv6 loopback address and a router ID on the device.

   Although the underlay uses the IPv6 address family, for BGP handshaking to work in the overlay, you must configure the router ID as an IPv4 address.

   The router ID is often the device's IPv4 loopback address, but is not required to match that address. For simplicity in this example, we don't assign an IPv4 loopback address, but to easily associate device IPv6 addresses and router IDs, we assign IPv4 router IDs with similar address components.

In Figure 37 on page 106, the device IPv6 loopback address for Leaf1 is 2001:db8::192:168:1:1 and the IPv4 router ID is 192.168.1.1.

```
set interfaces lo0 unit 0 family inet6 address 2001:db8::192:168:1:1/128 primary
set routing-options router-id 192.168.1.1
```

7. Enable EBGP (`type external`) with IPv6 as the underlay network routing protocol.

   With EBGP, each device in the underlay fabric has a unique local 32-bit autonomous system (AS) number (ASN). Figure 37 on page 106 shows the ASN values for each device in this configuration example. The EBGP ASN for Leaf1 is 4200000011. Leaf1 connects to Spine1 (ASN 4200000001) and Spine2 (ASN 4200000002) using the IPv6 addresses of the aggregated Ethernet interfaces toward each spine device. The underlay routing configuration ensures that the devices can reliably reach each other.

   The only difference in this configuration from the IPv4 Fabric configuration in "IP Fabric Underlay Network Design and Implementation" on page 80 is that you use IPv6 addressing instead of IPv4 addressing.

   In this step, you also enable BGP multipath with the multiple AS option. By default, EBGP selects one best path for each prefix and installs that route in the forwarding table. When you enable BGP multipath, the device installs all equal-cost paths to a given destination into the forwarding table. The `multiple-as` option enables load balancing between EBGP neighbors in different autonomous systems.

```
set protocols bgp group underlay-ipv6-bgp type external
set protocols bgp group underlay-ipv6-bgp local-as 4200000011
set protocols bgp group underlay-ipv6-bgp multipath multiple-as
set protocols bgp group underlay-ipv6-bgp neighbor 2001:db8::173:16:1:1 peer-as 4200000001
set protocols bgp group underlay-ipv6-bgp neighbor 2001:db8::173:16:2:1 peer-as 4200000002
```

8. Configure an export routing policy that advertises the IPv6 address of the loopback interface to the EBGP peer devices in the underlay.

   In this example, because we configure only an IPv6 address on the loopback interface, this simple policy correctly retrieves and advertises that address in the EBGP underlay.

```
set policy-options policy-statement underlay-ipv6-clos-export term loopback from interface
lo0.0
set policy-options policy-statement underlay-ipv6-clos-export term loopback then accept
set protocols bgp group underlay-ipv6-bgp export underlay-ipv6-clos-export
```

9. (QFX Series Broadcom-based switches running Junos OS releases in the 21.2 release train only) Enable the Broadcom VXLAN flexible flow feature on the device if needed.

   QFX Series Broadcom-based switches require the flexible flow feature to support IPv6 VXLAN tunneling. You don't need this step starting in Junos OS Release 21.4R1, where the default configuration enables this option for you on platforms that require this feature. When you set this option and commit the configuration, you must then reboot the device for the change to take effect.

   ```
   set forwarding-options vxlan-flexflow
   ```

10. (QFX5130 and QFX5700 switches only) On any QFX5130 or QFX5700 switches in the fabric that you configure with EVPN-VXLAN, make sure you set the `host-profile` unified forwarding profile option to support EVPN with VXLAN encapsulation (see *Layer 2 Forwarding Tables* for details):

    ```
    set system packet-forwarding-options forwarding-profile host-profile
    ```

## Configure EBGP for IPv6 Overlay Peering

Use this procedure with an IPv6 Fabric EBGP underlay to configure the IPv6 overlay peering. Both the underlay and overlay must use IPv6, so you can’t use the overlay configuration in "Configure IBGP for the Overlay" on page 96 (which describes configuring overlay peering in an IPv4 Fabric).

Because the overlay peering in the IPv6 Fabric also uses EBGP as the routing protocol, the overlay peering configuration is very similar to the underlay configuration in "Configure Interfaces and EBGP as the Routing Protocol in the IPv6 Fabric Underlay" on page 105. The main difference is that in the underlay, we specify the IPv6 addresses of the Layer 3 interfaces that connect to the EBGP neighbors (in this example, the aggregated Ethernet interface addresses). In contrast, in the overlay, we use the device IPv6 loopback addresses to specify the EBGP neighbors. Refer again to Figure 37 on page 106 for the device addresses and ASN values in this example.

Another difference in the overlay configuration here is that we configure EVPN signaling.

To configure EBGP overlay peering with IPv6 on Leaf1 to Spine1 and Spine2:

1. Enable IPv6 EBGP peering with EVPN signalling between the leaf and spine devices. Specify the device’s IPv6 loopback address as the `local-address` in the overlay BGP group configuration.

   In this step, similar to the underlay EBGP configuration, you also:

   - Specify the device’s local ASN.

- Enable BGP multipath with the multiple AS option to install all equal-cost paths to a destination into the forwarding table, and enable load balancing between EBGP neighbors with different ASNs.

In the overlay BGP group, when you configure the neighbor devices, you specify the IPv6 loopback addresses of the peer neighbor devices. (The underlay BGP group configuration uses the interconnecting aggregated Ethernet interface IPv6 address for neighbor peering.)

```
set protocols bgp group overlay-ipv6-ebgp type external
set protocols bgp group overlay-ipv6-ebgp local-address 2001:db8::192:168:1:1
set protocols bgp group overlay-ipv6-ebgp local-as 4200000011
set protocols bgp group overlay-ipv6-ebgp multipath multiple-as
set protocols bgp group overlay-ipv6-ebgp neighbor 2001:db8::192:168:0:1 peer-as 4200000001
set protocols bgp group overlay-ipv6-ebgp neighbor 2001:db8::192:168:0:2 peer-as 4200000002
```

2. Set the `multihop` option to enable EBGP peering in the overlay using device loopback addresses.

   When we use EBGP in the overlay, the EBGP peering happens between the device IPv6 loopback addresses. However, EBGP was designed to establish peering between directly-connected IP or IPv6 interface addresses. As a result, EBGP peering between device loopback addresses requires an extra hop for an EBGP control packet to reach its destination. The `multihop` option enables the device to establish the EBGP sessions in the overlay under these conditions.

   Also include the `no-nexthop-change` option with the multihop statement so that intermediate EBGP overlay peers don’t change the BGP next-hop attribute from the originating value across multiple hops.

```
set protocols bgp group overlay-ipv6-ebgp multihop no-nexthop-change
```

3. (Recommended in the overlay) Enable Bidirectional Forwarding Detection (BFD) in the overlay to help detect BGP neighbor failures.

```
set protocols bgp group overlay-ipv6-ebgp bfd-liveness-detection minimum-interval 1000
set protocols bgp group overlay-ipv6-ebgp bfd-liveness-detection multiplier 3
set protocols bgp group overlay-ipv6-ebgp bfd-liveness-detection session-mode automatic
```

## Verify EBGP IPv6 Underlay and Overlay Device Connectivity

After you've committed the underlay and overlay configurations in "Configure Interfaces and EBGP as the Routing Protocol in the IPv6 Fabric Underlay" on page 105 and "Configure EBGP for IPv6 Overlay Peering" on page 109, issue the following commands:

1. Enter the **show bgp summary** command on Leaf1 to confirm EBGP connectivity toward the spine devices.

   The example output shows the following:

   - The established underlay connectivity with Spine1 and Spine 2—refer to the `Peer` column showing aggregated Ethernet interface addresses 2001:db8::173:16:1:1 and 2001:db8::173:16:2:1, respectively.

   - The established overlay peering toward Spine1 and Spine2—refer to the `Peer` column showing device loopback addresses 2001:db8::192:168:0:1 and 2001:db8::192:168:0:2, respectively.

```
user@leaf1> show bgp summary

Threading mode: BGP I/O
Default eBGP mode: advertise - accept, receive - accept
Groups: 2 Peers: 4 Down peers: 0
Table          Tot Paths  Act Paths Suppressed    History Damp State    Pending
bgp.evpn.0
                   41024      20512          0          0          0          0
inet6.0
                      18         16          0          0          0          0
Peer                         AS      InPkt     OutPkt    OutQ   Flaps Last Up/Dwn   State|
#Active/Received/Accepted/Damped...
2001:db8::192:168:0:1  4200000001     858480     621472       0       1 3w5d 22:44:29 Establ
  bgp.evpn.0: 100/204/204/0
  MAC-VRF-1.evpn.0: 50/100/100/0
  default-switch.evpn.0: 0/0/0/0
  __default_evpn__.evpn.0: 0/4/4/0
2001:db8::192:168:0:2  4200000002     917586     551727       0       1 3w5d 22:44:29 Establ
  bgp.evpn.0: 104/204/204/0
  MAC-VRF-1: 50/100/100/0
  default-switch.evpn.0: 0/0/0/0
  __default_evpn__.evpn.0: 4/4/4/0
2001:db8::173:16:1:1   4200000001      84449      85373       0       1 3w5d 22:44:30
  inet6.0: 8/9/9/0
```

```
2001:db8::173:16:2:1    4200000002      84451       85396       0       1 3w5d 22:44:28
  inet6.0: 8/9/9/0
```

2. Enter the **show bfd session** command on Leaf1 to verify the BFD session is up between Leaf1 and the two spine devices (loopback IPv6 addresses 2001:db8::192:168:0:1 and 2001:db8::192:168:0:2).

```
user@leaf1> show bfd session
                                                  Detect   Transmit
Address                  State     Interface      Time     Interval  Multiplier
2001:db8::192:168:0:1    Up                       3.000     1.000        3
2001:db8::192:168:0:2    Up                       3.000     1.000        3
```

## RELATED DOCUMENTATION

IP Fabric Underlay Network Design and Implementation | 80

Configure IBGP for the Overlay | 96

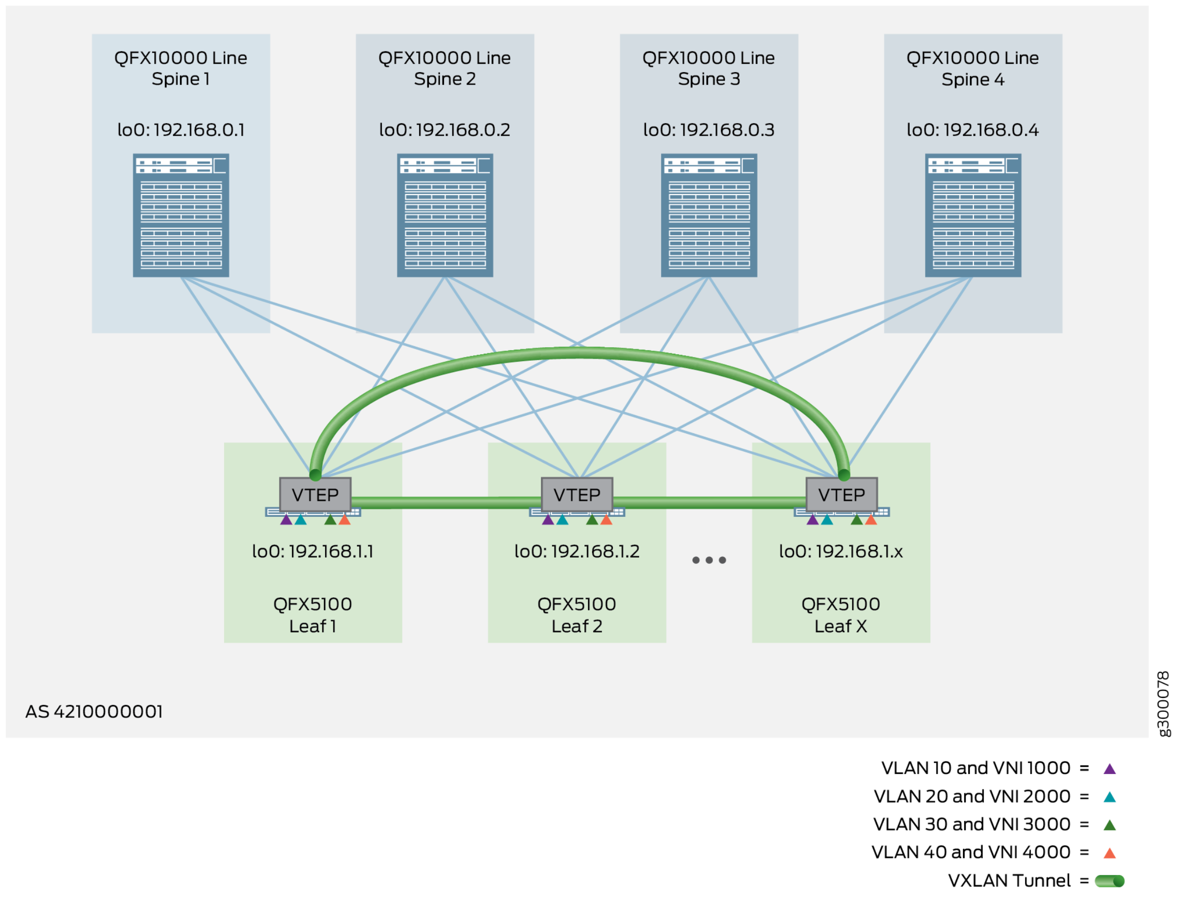
# Bridged Overlay Design and Implementation

**IN THIS SECTION**

A *bridged overlay* provides Ethernet bridging between leaf devices in an EVPN network, as shown in Figure 38 on page 113. This overlay type simply extends VLANs between the leaf devices across VXLAN tunnels. Bridged overlays provide an entry level overlay style for data center networks that require Ethernet connectivity but do not need routing services between the VLANs.

In this example, loopback interfaces on the leaf devices act as VXLAN tunnel endpoints (VTEPs). The tunnels enable the leaf devices to send VLAN traffic to other leaf devices and Ethernet-connected end systems in the data center. The spine devices only provide basic EBGP underlay and IBGP overlay connectivity for these leaf-to-leaf VXLAN tunnels.

**Figure 38: Bridged Overlay**



**NOTE**: If inter-VLAN routing is required for a bridged overlay, you can use an MX Series router or SRX Series security device that is external to the EVPN/VXLAN fabric. Otherwise, you can select one of the other overlay types that incorporate routing (such as an "edge-routed bridging overlay" on page 206, a "centrally-routed bridging overlay" on page 142, or a "routed overlay" on page 237) discussed in this Cloud Data Center Architecture Guide.

The following sections provide the detailed steps of how to configure a bridged overlay:

## Configuring a Bridged Overlay

IN THIS SECTION

Bridged overlays are supported on all platforms included in this reference design. To configure a bridged overlay, you configure VNIs, VLANs, and VTEPs on the leaf devices, and BGP on the spine devices. We support either an IPv4 Fabric or an IPv6 Fabric (with supported platforms) as the fabric infrastructure with bridged overlay architectures.

When you implement this style of overlay on a spine device, the focus is on providing overlay transport services between the leaf devices. Consequently, you configure an IP Fabric underlay and IBGP overlay peering with IPv4, or an IPv6 Fabric underlay with EBGP IPv6 overlay peering. There are no VTEPs or IRB interfaces needed, because the spine device does not provide any routing functionality or EVPN/VXLAN capabilities in a bridged overlay.

On the leaf devices, you can configure a bridged overlay using the default switch instance or using MAC-VRF instances.

**NOTE**: We support EVPN-VXLAN on devices running Junos OS Evolved only with MAC-VRF instance configurations.

In addition, we support the IPv6 Fabric infrastructure design only with MAC-VRF instance configurations.

Some configuration steps that affect the Layer 2 configuration differ with MAC-VRF instances. Likewise, one or two steps might differ for IPv6 Fabric configurations compared to IPv4 Fabric configurations. The leaf device configuration includes the following steps:

- Enable EVPN with VXLAN encapsulation to connect to other leaf devices, and configure the loopback interface as a VTEP source interface. If you are using MAC-VRF instances instead of the default switching instance, configure a MAC-VRF instance with these parameters in the MAC-VRF instance. If your fabric uses an IPv6 Fabric, you configure the VTEP source interface as an IPv6 interface.

- Establish route targets and route distinguishers. If you are using MAC-VRF instances instead of the default switching instance, configure a MAC-VRF instance with these parameters in the MAC-VRF instance.
- Configure Ethernet Segment Identifier (ESI) settings.
- Map VLANs to VNIs.

Again, you do not include IRB interfaces or routing on the leaf devices for this overlay method.

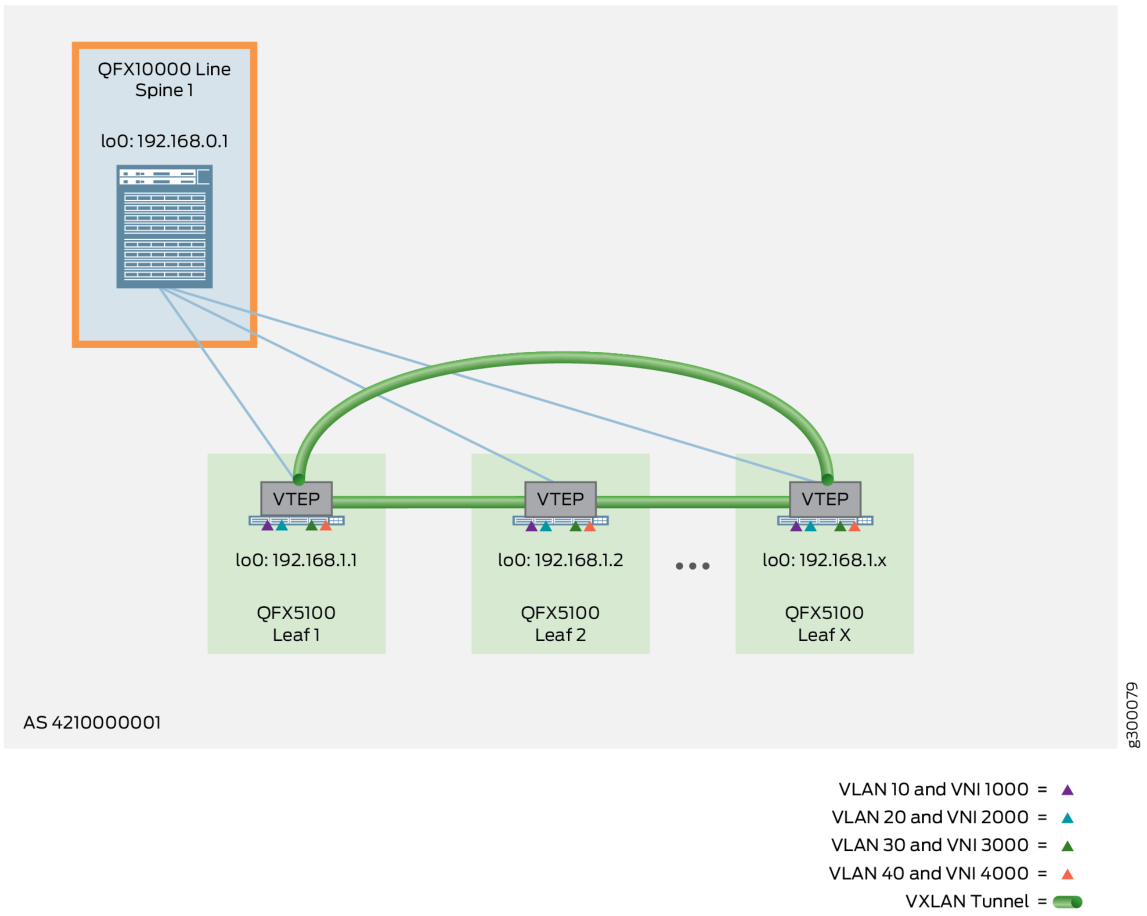
The following sections provide the detailed steps of how to configure and verify the bridged overlay:

## Configuring a Bridged Overlay on the Spine Device

To configure a bridged overlay on a spine device, perform the following steps:

**NOTE**: The following example shows the configuration for Spine 1, as shown in Figure 39 on page 116.

**Figure 39: Bridged Overlay – Spine Device**



1. Ensure the IP fabric underlay is in place. To configure an IP fabric on a spine device, see "IP Fabric Underlay Network Design and Implementation" on page 80.

   If you are using an IPv6 Fabric, see "IPv6 Fabric Underlay and Overlay Network Design and Implementation with EBGP" on page 103 instead. Those instructions include how to configure the IPv6 underlay connectivity with EBGP and IPv6 overlay peering.

2. Confirm that your IBGP overlay is up and running. To configure an IBGP overlay on your spine devices, see "Configure IBGP for the Overlay" on page 96.

   If you are using an IPv6 Fabric, you don't need this step. Step 1 also covers how to configure the EBGP IPv6 overlay peering that corresponds to the IPv6 underlay connectivity configuration.

3. (QFX5130 and QFX5700 switches only) On any QFX5130 or QFX5700 switches in the fabric that you configure with EVPN-VXLAN, set the `host-profile` unified forwarding profile option to support EVPN with VXLAN encapsulation (see *Layer 2 Forwarding Tables* for details):

```
set system packet-forwarding-options forwarding-profile host-profile
```

## Verifying a Bridged Overlay on the Spine Device

Issue the following commands to verify that the overlay is working properly on your spine devices:

1. Verify that the spine device has reachability to the leaf devices. This output shows the possible routes to Leaf 1.

   (With an IPv6 Fabric, enter this command with the IPv6 address of the spine device instead of an IPv4 address.)

```
user@spine-1> show route 192.168.1.1

inet.0: 446 destinations, 19761 routes (446 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

192.168.1.1/32     *[BGP/170] 00:06:29, localpref 100
                      AS path: 4200000010 I, validation-state: unverified
                    > to 172.16.1.1 via ae1.0

                     ...

                     [BGP/170] 00:06:18, localpref 100
                      AS path: 4200000106 4200000004 4200000010 I, validation-state: 
unverified
                    > to 172.16.96.1 via ae96.0
```

2. Verify that IBGP is functional on the spine devices acting as a route reflector cluster. You should see peer relationships with all spine device loopback interfaces (192.168.0.1 through 192.168.0.4) and all leaf device loopback interfaces (192.168.1.1 through 192.168.1.96).

   Use the same command if you have an IPv6 Fabric with EBGP IPv6 overlay peering. In the output, look for the IPv6 addresses of the peer device interconnecting interfaces to verify underlay EBGP connectivity. Look for peer device loopback addresses to verify overlay EBGP peering. Ensure that the state is `Establ` (established).

```
user@spine-1> show bgp summary
Groups: 5 Peers: 221 Down peers: 0
Table          Tot Paths  Act Paths Suppressed    History Damp State    Pending
inet.0
                    9711        182          0          0          0          0
...
Peer                     AS      InPkt     OutPkt    OutQ   Flaps Last Up/Dwn State|#Active/
Received/Accepted/Damped...
192.168.0.2       421000001      28724      31106       0       0    22:40:41 Establ
```

```
192.168.0.3        421000001       27424       32106        0        0     22:43:41 Establ
192.168.0.4        421000001       29457       30494        0        0     22:39:04 Establ
192.168.1.1        421000001        3814       75108        0        0     22:43:54 Establ
...
192.168.1.96       421000001        4831       73047        0        0     22:43:41 Establ
```

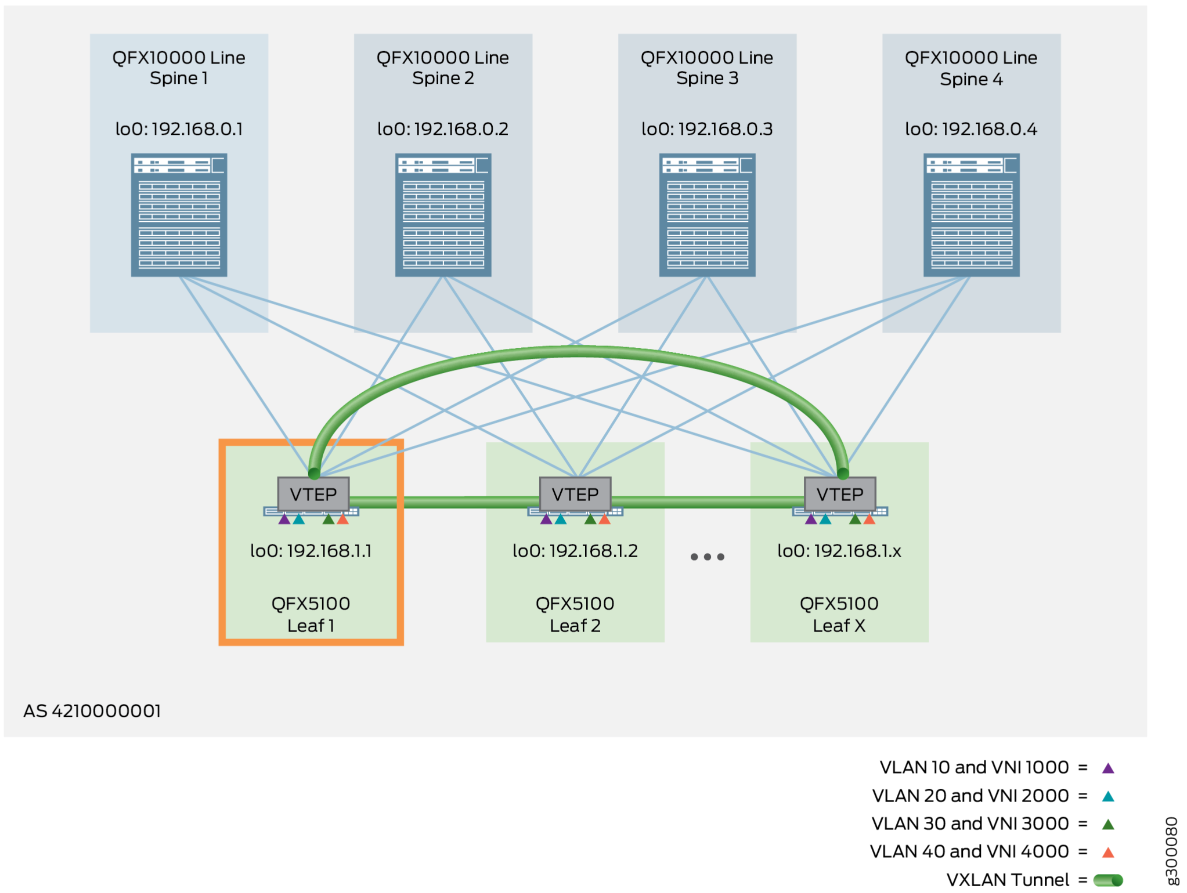
## Configuring a Bridged Overlay on the Leaf Device

To configure a bridged overlay on a leaf device, perform the following:

**NOTE:**

- The following example shows the configuration for Leaf 1, as shown in Figure 40 on page 118.

**Figure 40: Bridged Overlay – Leaf Device**



1. Configure the IP Fabric underlay and overlay:

   For an IP Fabric underlay using IPv4:

   - Ensure the IP fabric underlay is in place. To configure an IP fabric on a leaf device, see "IP Fabric Underlay Network Design and Implementation" on page 80.

   - Confirm that your IBGP overlay peering is up and running. To configure IPv4 IBGP overlay peering on your leaf device, see "Configure IBGP for the Overlay" on page 96.

   For an IPv6 Fabric underlay with EBGP IPv6 overlay peering:

   - Ensure the IPv6 underlay is in place and the EBGP overlay peering is up and running. To configure an IPv6 Fabric, see "IPv6 Fabric Underlay and Overlay Network Design and Implementation with EBGP" on page 103.

2. Configure the EVPN protocol with VXLAN encapsulation, and specify the VTEP source interface (in this case, the loopback interface of the leaf device).

   If your configuration uses the default instance, you configure EVPN-VXLAN at the global level. You also specify the VTEP source interface at the `[edit switch-options]` hierarchy level:

   *Leaf 1 (Default Instance)*:

   ```
   set protocols evpn encapsulation vxlan
   set protocols evpn extended-vni-list all
   set switch-options vtep-source-interface lo0.0
   ```

   If your configuration uses MAC-VRF instances, define a routing instance of type `mac-vrf`. Then configure EVPN-VXLAN and the VTEP source interface at that MAC-VRF routing instance hierarchy level. You also must configure a service type for the MAC-VRF instance. We configure the `vlan-aware` service type so you can associate multiple VLANs with the MAC-VRF instance. This setting is consistent with the alternative configuration that uses the default instance.

   *Leaf 1 (MAC-VRF Instance)*:

   ```
   set routing-instances MAC-VRF-1 instance-type mac-vrf
   set routing-instances MAC-VRF-1 service-type vlan-aware
   set routing-instances MAC-VRF-1 protocols evpn encapsulation vxlan
   set routing-instances MAC-VRF-1 protocols evpn extended-vni-list all
   set routing-instances MAC-VRF-1 vtep-source-interface lo0.0
   ```

   If you have an IPv6 Fabric infrastructure (supported only with MAC-VRF instances), in this step you include the `inet6` option when you configure the VTEP source interface to use the device loopback address. This option enables IPv6 VXLAN tunneling in the fabric. This is the only difference in the

MAC-VRF instance configuration with an IPv6 Fabric as compared to the MAC-VRF instance configuration with an IPv4 Fabric.

*Leaf 1 (MAC-VRF Instance with an IPv6 Fabric)*:

```
set routing-instances MAC-VRF-1 instance-type mac-vrf
set routing-instances MAC-VRF-1 service-type vlan-aware
set routing-instances MAC-VRF-1 protocols evpn encapsulation vxlan
set routing-instances MAC-VRF-1 protocols evpn extended-vni-list all
set routing-instances MAC-VRF-1 vtep-source-interface lo0.0 inet6
```

3. Define an EVPN route target and route distinguisher, and use the `auto` option to derive route targets automatically. Setting these parameters specifies how the routes are imported and exported. The import and export of routes from a bridging table is the basis for dynamic overlays. In this case, members of the global BGP community with a route target of target:64512:1111 participate in the exchange of EVPN/VXLAN information.

   If your configuration uses the default instance, you use statements in the `[edit switch-options]` hierarchy, as follows:

   *Leaf 1 (Default Instance)*:

```
set switch-options route-distinguisher 192.168.1.1:1
set switch-options vrf-target target:64512:1111
set switch-options vrf-target auto
```

   The main difference with a MAC-VRF configuration is that you configure these statements in the MAC-VRF instance at the `[edit routing-instances` *`mac-vrf-instance-name`*`]` hierarchy level, as follows:

   *Leaf 1 (MAC-VRF Instance)*:

```
set routing-instances MAC-VRF-1 route-distinguisher 192.168.1.1:1
set routing-instances MAC-VRF-1 vrf-target target:64512:1111
set routing-instances MAC-VRF-1 vrf-target auto
```

   **NOTE**: A specific route target processes EVPN Type 1 routes, while an automatic route target processes Type 2 routes. This reference design requires both route targets.

4. (MAC-VRF instances only) Enable shared tunnels on devices in the QFX5000 line running Junos OS.

   A device can have problems with VTEP scaling when the configuration uses multiple MAC-VRF instances. As a result, to avoid this problem, we require that you enable the shared tunnels feature on the QFX5000 line of switches running Junos OS with a MAC-VRF instance configuration. When you configure the shared tunnels option, the device minimizes the number of next-hop entries to reach remote VTEPs. This statement is optional on the QFX10000 line of switches running Junos OS because those devices can handle higher VTEP scaling than QFX5000 switches. You also don't need to configure this option on devices running Junos OS Evolved, where shared tunnels are enabled by default.

Include the following statement to globally enable shared VXLAN tunnels on the device:

```
set forwarding-options evpn-vxlan shared-tunnels
```

**NOTE**: This setting requires you to reboot the device.

5. (Required on PTX10000 Series routers only) Enable tunnel termination globally (in other words, on all interfaces) on the device:

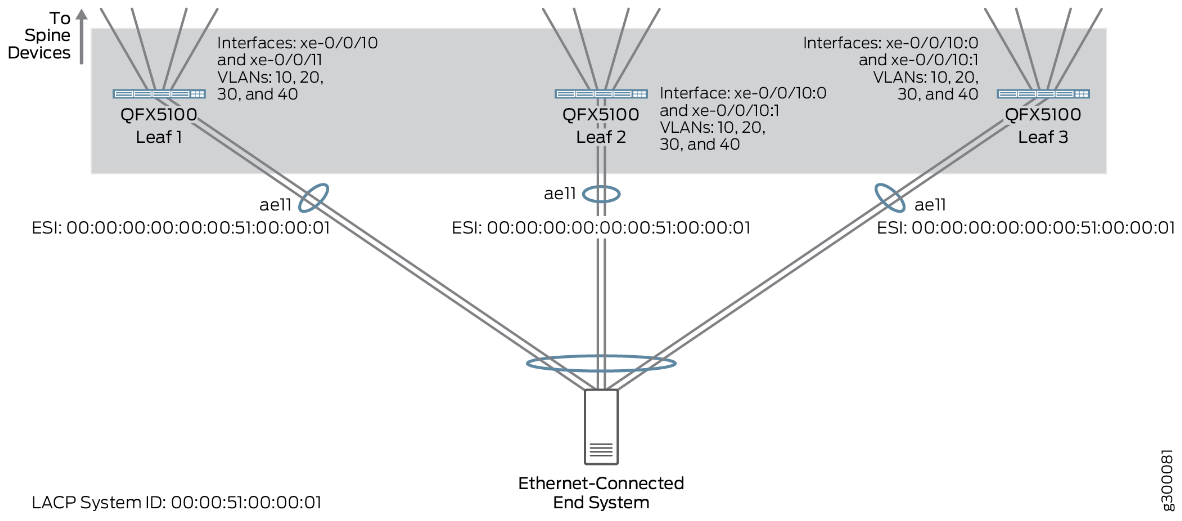
```
set forwarding-options tunnel-termination
```

6. Configure ESI settings. Because the end systems in this reference design are multihomed to three leaf devices per device type cluster (such as QFX5100), you must configure the same ESI identifier and LACP system identifier on all three leaf devices for each unique end system. Unlike other topologies where you would configure a different LACP system identifier per leaf device and have VXLAN select a single designated forwarder, use the same LACP system identifier to allow the 3 leaf devices to appear as a single LAG to a multihomed end system. In addition, use the same aggregated Ethernet interface number for all ports included in the ESI.

   The configuration for Leaf 1 is shown below, but you must replicate this configuration on both Leaf 2 and Leaf 3 per the topology shown in Figure 41 on page 122.

   **TIP**: When you create an ESI number, always set the high order octet to 00 to indicate the ESI is manually created. The other 9 octets can be any hexadecimal value from 00 to FF.

**Figure 41: ESI Topology for Leaf 1, Leaf 2, and Leaf 3**


To Spine Devices
Interfaces: xe-0/0/10 and xe-0/0/11 VLANs: 10, 20, 30, and 40
QFX5100 Leaf 1
QFX5100 Leaf 2
Interface: xe-0/0/10:0 and xe-0/0/10:1 VLANs: 10, 20, 30, and 40
Interfaces: xe-0/0/10:0 and xe-0/0/10:1 VLANs: 10, 20, 30, and 40
QFX5100 Leaf 3
ae11
ESI: 00:00:00:00:00:00:51:00:00:01
ae11
ESI: 00:00:00:00:00:00:51:00:00:01
ae11
ESI: 00:00:00:00:00:00:51:00:00:01
LACP System ID: 00:00:51:00:00:01
Ethernet-Connected End System
g300081

*Leaf 1*:

```
set interfaces ae11 esi 00:00:00:00:00:00:51:00:00:01
set interfaces ae11 esi all-active
set interfaces ae11 aggregated-ether-options lacp active
set interfaces ae11 aggregated-ether-options lacp system-id 00:00:51:00:00:01
set interfaces ae11 unit 0 family ethernet-switching interface-mode trunk
set interfaces ae11 unit 0 family ethernet-switching vlan members [ 10 20 30 40 ]
set interfaces xe-0/0/10 ether-options 802.3ad ae11
set interfaces xe-0/0/11 ether-options 802.3ad ae11
```

If your configuration uses MAC-VRF instances, you must also add the configured aggregated Ethernet interface to the MAC-VRF instance:

```
set routing-instances MAC-VRF-1 interface ae11.0
```

7. Configure VLANs and map them to VNIs. This step enables the VLANs to participate in VNIs across the EVPN/VXLAN domain.

   This step shows the VLAN to VNI mapping either in the default instance or in a MAC-VRF instance configuration.

*Leaf 1 (Default Instance)*:

```
set vlans VNI_1000 vlan-id 10
set vlans VNI_1000 vxlan vni 1000
set vlans VNI_2000 vlan-id 20
set vlans VNI_2000 vxlan vni 2000
set vlans VNI_3000 vlan-id 30
set vlans VNI_3000 vxlan vni 3000
set vlans VNI_4000 vlan-id 40
set vlans VNI_4000 vxlan vni 4000
```

*Leaf 1 (MAC-VRF Instance)*:

The only difference with a MAC-VRF instance configuration is that you configure these statements in the MAC-VRF instance at the `[edit routing-instances` *`mac-vrf-instance-name`*`]` hierarchy level, as follows:

```
set routing-instances MAC-VRF-1 vlans VNI_1000 vlan-id 10
set routing-instances MAC-VRF-1 vlans VNI_1000 vxlan vni 1000
set routing-instances MAC-VRF-1 vlans VNI_2000 vlan-id 20
set routing-instances MAC-VRF-1 vlans VNI_2000 vxlan vni 2000
set routing-instances MAC-VRF-1 vlans VNI_3000 vlan-id 30
set routing-instances MAC-VRF-1 vlans VNI_3000 vxlan vni 3000
set routing-instances MAC-VRF-1 vlans VNI_4000 vlan-id 40
set routing-instances MAC-VRF-1 vlans VNI_4000 vxlan vni 4000
```

## Verifying the Bridged Overlay on the Leaf Device

Run the following commands to verify that the overlay is working properly on your leaf devices.

The commands here show output for a default instance configuration. With a MAC-VRF instance configuration, you can alternatively use:

- `show mac-vrf forwarding` commands that are aliases for the `show ethernet-switching` commands in this section.
- The `show mac-vrf routing instance` command, which is an alias for the `show evpn instance` command in this section.

See MAC-VRF Routing Instance Type Overview for tables of `show mac-vrf forwarding` and `show ethernet-switching` command mappings, and `show mac-vrf routing` command aliases for `show evpn` commands.

The output with a MAC-VRF instance configuration displays similar information for MAC-VRF routing instances as this section shows for the default instance. One main difference you might see is in the output with MAC-VRF instances on devices where you enable the shared tunnels feature. With shared tunnels enabled, you see VTEP interfaces in the following format:

```
vtep-index.shared-tunnel-unit
```

where:

- *index* is the index associated with the MAC-VRF routing instance.
- *shared-tunnel-unit* is the unit number associated with the shared tunnel remote VTEP logical interface.

For example, if a device has a MAC-VRF instance with index 26 and the instance connects to two remote VTEPs, the shared tunnel VTEP logical interfaces might look like this:

```
vtep-26.32823
vtep-26.32824
```

If your configuration uses an IPv6 Fabric, you provide IPv6 address parameters where applicable. Output from the commands that display IP addresses reflect the IPv6 device and interface addresses from the underlying fabric. See "IPv6 Fabric Underlay and Overlay Network Design and Implementation with EBGP" on page 103 for the fabric parameters reflected in command outputs in this section with an IPv6 Fabric.

1. Verify the interfaces are operational. Interfaces xe-0/0/10 and xe-0/0/11 are dual homed to the Ethernet-connected end system through interface ae11, while interfaces et-0/0/48 through et-0/0/51 are uplinks to the four spine devices.

```
user@leaf-1> show interfaces terse | match ae.*
xe-0/0/10.0             up    up   aenet    --> ae11.0
et-0/0/48.0             up    up   aenet    --> ae1.0
et-0/0/49.0             up    up   aenet    --> ae2.0
et-0/0/50.0             up    up   aenet    --> ae3.0
et-0/0/51.0             up    up   aenet    --> ae4.0
xe-0/0/11.0             up    up   aenet    --> ae11.0
ae1                     up    up  ## To Spine 1
ae1.0                   up    up   inet     172.16.1.1/30
ae2                     up    up  ## To Spine 2
ae2.0                   up    up   inet     172.16.1.5/30
ae3                     up    up  ## To Spine 3
ae3.0                   up    up   inet     172.16.1.9/30
ae4                     up    up  ## To Spine 4
```

```
ae4.0                  up    up   inet     172.16.1.13/30
ae11                   up    up  ## To End System
ae11.0                 up    up   eth-switch
user@leaf-1> show lacp interfaces
Aggregated interface: ae1
    LACP state:       Role   Exp   Def  Dist  Col  Syn  Aggr  Timeout  Activity
      et-0/0/48      Actor    No    No   Yes  Yes  Yes   Yes     Fast    Active
      et-0/0/48    Partner    No    No   Yes  Yes  Yes   Yes     Fast    Active
    LACP protocol:        Receive State  Transmit State          Mux State
      et-0/0/48                 Current   Fast periodic Collecting distributing



...

Aggregated interface: ae11
    LACP state:       Role   Exp   Def  Dist  Col  Syn  Aggr  Timeout  Activity
      xe-0/0/10      Actor    No    No   Yes  Yes  Yes   Yes     Fast    Active
      xe-0/0/10    Partner    No    No   Yes  Yes  Yes   Yes     Fast    Active
      xe-0/0/11      Actor    No    No   Yes  Yes  Yes   Yes     Fast    Active
      xe-0/0/11    Partner    No    No   Yes  Yes  Yes   Yes     Fast    Active
    LACP protocol:        Receive State  Transmit State          Mux State
      xe-0/0/10                 Current   Fast periodic Collecting distributing
      xe-0/0/11                 Current   Fast periodic Collecting distributing


user@leaf-1> show ethernet-switching interface ae11
Routing Instance Name : default-switch
Logical Interface flags (DL - disable learning, AD - packet action drop,
                         LH - MAC limit hit, DN - interface down,
                         MMAS - Mac-move action shutdown,  AS - Autostate-exclude enabled,
                         SCTL - shutdown by Storm-control, MI - MAC+IP limit hit)

Logical         Vlan                  TAG   MAC    MAC+IP STP         Logical
Tagging
interface       members                     limit  limit  state       interface flags
ae11.0                                      65535  0
tagged
                VNI_1000              10    65535  0      Forwarding
tagged
                VNI_2000              20    65535  0      Forwarding
tagged
                VNI_3000              30    65535  0      Forwarding
tagged
```

```
                VNI_4000              40    65535  0      Forwarding
tagged
```

2. Verify that the leaf devices have reachability to their peer leaf devices.

   For example, on Leaf 1 with an IPv6 Fabric, view the possible routes to remote Leaf 2 using the `show route` *`address`* command with device IPv6 address 2001:db8::192:168:1:2 for Leaf 2.

```
user@leaf-1> show route 2001:db8::192:168:1:2



inet6.0: 18 destinations, 25 routes (18 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

2001:db8::192:168:1:2/128
                   *[BGP/170] 18:53:41, localpref 100
                      AS path: 4200000001 4200000012 I, validation-state: unverified
                    >  to 2001:db8::173:16:1:1 via ae1.0
                    [BGP/170] 18:53:41, localpref 100
                      AS path: 4200000002 4200000012 I, validation-state: unverified
                    >  to 2001:db8::173:16:2:1 via ae2.0

:vxlan.inet6.0: 11 destinations, 11 routes (11 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

2001:db8::192:168:1:2/128
                   *[Static/1] 22:23:04, metric2 0
                       to 2001:db8::173:16:2:1 via ae2.0
                    >  to 2001:db8::173:16:1:1 via ae1.0
```

3. Verify on Leaf 1 and Leaf 3 that the Ethernet switching table has installed both the local MAC addresses and the remote MAC addresses learned through the overlay.

   **NOTE**: To identify end systems learned remotely from the EVPN overlay, look for the MAC address, ESI logical interface, and ESI number. For example, Leaf 1 learns about an end system with the MAC address of `02:0c:10:03:02:02` through `esi.1885`. This end system has an ESI number of `00:00:00:00:00:00:51:10:00:01`. Consequently, this matches the ESI number configured for Leaf 4, 5, and 6 (QFX5110 switches), so we know that this end system is multihomed to these three leaf devices.

```
user@leaf-1> show ethernet-switching table vlan-id 30

MAC flags (S - static MAC, D - dynamic MAC, L - locally learned, P - Persistent static
           SE - statistics enabled, NM - non configured MAC, R - remote PE MAC, O - ovsdb
MAC)



Ethernet switching table : 10 entries, 10 learned
Routing instance : default-switch
   Vlan                MAC                 MAC      Logical              Active
   name                address             flags    interface            source
## Learned locally
   VNI_3000            02:0c:10:03:02:01   DL       ae11.0
## Learned from the QFX5110 switches - Leaf 4 to 6
   VNI_3000            02:0c:10:03:02:02   DR       esi.1885
00:00:00:00:00:00:51:10:00:01
## Learned from the QFX5200 switches - Leaf 7 to 9
   VNI_3000            02:0c:10:03:02:03   DR       esi.1887
00:00:00:00:00:00:52:00:00:01
## Learned from the QFX10002 switches - Leaf 10 to 12
   VNI_3000            02:0c:10:03:02:04   DR       esi.1892
00:00:00:00:00:01:00:00:00:01
## End System MAC address, connected locally to the leaf
device
02:0c:10:03:02:01
## MAC address learned over the overlay, these end systems
are also multihomed
02:0c:10:03:02:02,03,04
```

4. Verify the remote EVPN routes from a specific VNI and MAC address.

**NOTE**: The format of the EVPN routes is *EVPN-route-type:route-distinguisher:vni:mac-address*.

For example, with an IPv4 Fabric, view the remote EVPN routes from VNI 1000 and MAC address 02:0c:10:01:02:02. In this case, the EVPN routes come from Leaf 4 (route distinguisher 192.168.1.4) by way of Spine 1 (192.168.0.1).

```
user@leaf-1> show route table bgp.evpn.0 evpn-ethernet-tag-id 1000 evpn-mac-address
02:0c:10:01:02:02

bgp.evpn.0: 910 destinations, 3497 routes (904 active, 0 holddown, 24 hidden)
+ = Active Route, - = Last Active, * = Both

2:192.168.1.4:1::1000::02:0c:10:01:02:02/304 MAC/IP
                   *[BGP/170] 00:11:37, localpref 100, from 192.168.0.1
                      AS path: I, validation-state: unverified
                    > to 172.16.1.10 via ae3.0
                    [BGP/170] 00:11:37, localpref 100, from 192.168.0.2
                      AS path: I, validation-state: unverified
                    > to 172.16.1.10 via ae3.0
                    [BGP/170] 00:11:37, localpref 100, from 192.168.0.3
                      AS path: I, validation-state: unverified
                    > to 172.16.1.10 via ae3.0
                    [BGP/170] 00:11:37, localpref 100, from 192.168.0.4
                      AS path: I, validation-state: unverified
                    > to 172.16.1.10 via ae3.0
user@leaf-1> show route table bgp.evpn.0 evpn-ethernet-tag-id 1000 evpn-mac-address
02:0c:10:01:02:02 detail

bgp.evpn.0: 925 destinations, 3557 routes (919 active, 0 holddown, 24 hidden)
2:192.168.1.4:1::1000::02:0c:10:01:02:02/304 MAC/IP (4 entries, 0 announced)
        *BGP    Preference: 170/-101
                Route Distinguisher: 192.168.1.4:1
                Next hop type: Indirect, Next hop index: 0
                Address: 0xb3a2170
                Next-hop reference count: 160
                Source: 192.168.0.1
                Protocol next hop: 192.168.1.4
                Indirect next hop: 0x2 no-forward INH Session ID: 0x0
                State: <Active Int Ext>
                Local AS: 4210000001 Peer AS: 4210000001
                Age: 13:42      Metric2: 0
                Validation State: unverified
                Task: BGP_4210000001.192.168.0.1
                AS path: I (Originator)
                Cluster list:  192.168.0.10
                Originator ID: 192.168.1.4
                Communities: target:62273:268445456 encapsulation:vxlan(0x8)
                Import Accepted
                Route Label: 1000
                ESI: 00:00:00:00:00:00:51:10:00:01
                Localpref: 100
                Router ID: 192.168.0.1
                Secondary Tables: default-switch.evpn.0
...
## This output has been abbreviated.
```

Or with an IPv6 Fabric, for example, view the remote EVPN routes from VNI 1000 and MAC address c8:fe:6a:e4:2e:00. In this case, the EVPN routes come from Leaf 2 (route distinguisher 192.168.1.2) by way of Spine 1 (2001:db8::192:168:0:1).

```
user@leaf-1> show route table bgp.evpn.0 evpn-ethernet-tag-id 1000 evpn-mac-address
c8:fe:6a:e4:2e:00

bgp.evpn.0: 43916 destinations, 76851 routes (43916 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

2:192.168.1.2:1::1000::c8:fe:6a:e4:2e:00/304 MAC/IP
                   *[BGP/170] 19:46:55, localpref 100, from 2001:db8::192:168:0:2
                      AS path: 4200000001 4200000012 I, validation-state: unverified
                    >  to 2001:db8::173:16:1:1 via ae1.0
                    [BGP/170] 23:16:47, localpref 100, from 2001:db8::192:168:0:1
                      AS path: 4200000002 4200000012 I, validation-state: unverified
                    >  to 2001:db8::173:16:2:1 via ae2.0
2:192.168.1.2:1::1000::c8:fe:6a:e4:2e:00::77.5.241.242/304 MAC/IP
                   *[BGP/170] 19:46:55, localpref 100, from 2001:db8::192:168:0:2
                      AS path: 4200000001 4200000012 I, validation-state: unverified
                    >  to 2001:db8::173:16:1:1 via ae1.0
                    [BGP/170] 23:16:47, localpref 100, from 2001:db8::192:168:0:1
                      AS path: 4200000002 4200000012 I, validation-state: unverified
                    >  to 2001:db8::173:16:2:1 via ae2.0
2:192.168.1.2:1::1000::c8:fe:6a:e4:2e:00::2001:db8::77:0:5f1:242/304 MAC/IP
                   *[BGP/170] 19:46:55, localpref 100, from 2001:db8::192:168:0:2
                      AS path: 4200000001 4200000012 I, validation-state: unverified
                    >  to 2001:db8::173:16:1:1 via ae1.0
                    [BGP/170] 23:16:47, localpref 100, from 2001:db8::192:168:0:1
                      AS path: 4200000002 4200000012 I, validation-state: unverified
                    >  to 2001:db8::173:16:2:1 via ae2.0
2:192.168.1.2:1::1000::c8:fe:6a:e4:2e:00::fe80::cafe:6a05:f1e4:2e00/304 MAC/IP
                   *[BGP/170] 19:46:55, localpref 100, from 2001:db8::192:168:0:2
                      AS path: 4200000001 4200000012 I, validation-state: unverified
                    >  to 2001:db8::173:16:1:1 via ae1.0
                    [BGP/170] 23:16:47, localpref 100, from 2001:db8::192:168:0:1
                      AS path: 4200000002 4200000012 I, validation-state: unverified
                    >  to 2001:db8::173:16:2:1 via ae2.0

user@leaf-1> show route table bgp.evpn.0 evpn-ethernet-tag-id 1000 evpn-mac-address
c8:fe:6a:e4:2e:00 detail

bgp.evpn.0: 43916 destinations, 76851 routes (43916 active, 0 holddown, 0 hidden)
2:192.168.1.2:1::1000::c8:fe:6a:e4:2e:00/304 MAC/IP (2 entries, 0 announced)
        *BGP    Preference: 170/-101
                Route Distinguisher: 192.168.0.2:1
                Next hop type: Indirect, Next hop index: 0
                Address: 0x8c0701c
                Next-hop reference count: 41840
                Source: 2001:db8::192:168:0:1
                Protocol next hop: 2001:db8::192:168:1:2
                Indirect next hop: 0x2 no-forward INH Session ID: 0
                State: <Active Ext>
                Peer AS: 4200000001
                Age: 23:16:11   Metric2: 0
                Validation State: unverified
                Task: BGP_4200000001_42000000.2001:db8::192:168:0:1
                AS path: 4200000001 4200000012 I
                Communities: target:2:1000 encapsulation:vxlan(0x8) mac-mobility:0x1:sticky
(sequence 0)
                Import Accepted
                Route Label: 1000
                ESI: 00:00:00:00:00:00:00:00:00:00
                Localpref: 100
                Router ID: 192.168.0.1
                Secondary Tables: MAC-VRF-1.evpn.0
                Thread: junos-main
                Indirect next hops: 1
                        Protocol next hop: 2001:db8::192:168:1:2
                        Indirect next hop: 0x2 no-forward INH Session ID: 0
                        Indirect path forwarding next hops: 2
                                Next hop type: Router
                                Next hop: 2001:db8::173:16:1:1 via ae1.0
```

```
                                  Session Id: 0
                                  Next hop: 2001:db8::173:16:2:1 via ae2.0
                                  Session Id: 0
                                  2001:db8::192:168:1:2/128 Originating RIB: inet6.0
                                    Node path count: 1
                                    Forwarding nexthops: 2
                                          Next hop type: Router
                                          Next hop: 2001:db8::173:16:1:1 via ae1.0
                                          Session Id: 0
                                          Next hop: 2001:db8::173:16:2:1 via ae2.0
                                          Session Id: 0
```

5. Verify the source and destination address of each VTEP interface and view their status. Use the `show ethernet-switching vxlan-tunnel-end-point source` and `show interfaces vtep` commands.

**NOTE**: A scaled-out reference design can have 96 leaf devices, which corresponds to 96 VTEP interfaces - one VTEP interface per leaf device. The output here is truncated for readability.

The following example shows these commands with an IPv4 Fabric:

```
user@leaf-1> show ethernet-switching vxlan-tunnel-end-point source
Logical System Name       Id  SVTEP-IP         IFL   L3-Idx
<default>                 0   192.168.1.1      lo0.0    0
    L2-RTT                   Bridge Domain              VNID     MC-Group-IP
    default-switch           VNI_1000+10                1000     0.0.0.0
    default-switch           VNI_2000+20                2000     0.0.0.0
    default-switch           VNI_3000+30                3000     0.0.0.0
    default-switch           VNI_4000+40                4000     0.0.0.0
user@leaf-1> show interfaces terse vtep
Interface                Admin Link Proto    Local                 Remote
vtep                     up    up
vtep.32768               up    up
vtep.32769               up    up   eth-switch
vtep.32770               up    up   eth-switch
vtep.32771               up    up   eth-switch
vtep.32772               up    up   eth-switch
...
vtep.32869               up    up   eth-switch
user@leaf-1> show interfaces vtep
Physical interface: vtep, Enabled, Physical link is Up
```

```
  Interface index: 646, SNMP ifIndex: 503
  Type: Software-Pseudo, Link-level type: VxLAN-Tunnel-Endpoint, MTU: Unlimited, Speed:
Unlimited
  Device flags   : Present Running
  Link type      : Full-Duplex
  Link flags     : None
  Last flapped   : Never
    Input packets : 0
    Output packets: 0

  Logical interface vtep.32768 (Index 554) (SNMP ifIndex 648)
    Flags: Up SNMP-Traps 0x4000 Encapsulation: ENET2
    VXLAN Endpoint Type: Source, VXLAN Endpoint Address: 192.168.1.1, L2 Routing Instance:
default-switch, L3 Routing Instance: default
    Input packets : 0
    Output packets: 0



...  Logical interface vtep.32814 (Index 613) (SNMP ifIndex 903)
    Flags: Up SNMP-Traps Encapsulation: ENET2
    VXLAN Endpoint Type: Remote, VXLAN Endpoint Address: 192.168.1.96, L2 Routing Instance:
default-switch, L3 Routing Instance: default
    Input packets : 0
    Output packets: 6364
    Protocol eth-switch, MTU: Unlimited
      Flags: Trunk-Mode
```

Or the following example shows these commands with an IPv6 Fabric:

```
user@leaf-1> show ethernet-switching vxlan-tunnel-end-point source
Logical System Name       Id  SVTEP-IP              IFL   L3-Idx    SVTEP-Mode    ELP-SVTEP-
IP
<default>                 0   2001:db8::192:168:1:1 lo0.0 0
    L2-RTT                Bridge Domain        VNID     Translation-VNID    MC-Group-IP
    MAC-VRF-1             VNI_1000+10          1000                        ::
    MAC-VRF-1             VNI_2000+20          2000                        ::
    MAC-VRF-1             VNI_3000+30          3000                        ::
    MAC-VRF-1             VNI_4000+40          4000                        ::

user@leaf-1> show interfaces vtep
Physical interface: vtep, Enabled, Physical link is Up
  Interface index: 650, SNMP ifIndex: 506
  Type: Software-Pseudo, Link-level type: VxLAN-Tunnel-Endpoint, MTU: Unlimited, Speed:
Unlimited
  Device flags   : Present Running
  Link type      : Full-Duplex
  Link flags     : None
  Last flapped   : Never
    Input packets : 0
    Output packets: 0

  Logical interface vtep.32768 (Index 2031) (SNMP ifIndex 1737)
    Flags: Up SNMP-Traps 0x4000 Encapsulation: ENET2
    VXLAN Endpoint Type: Source, VXLAN Endpoint Address: 2001:db8::192:168:1:1, L2 Routing
Instance: MAC-VRF-1, L3 Routing Instance: default
    Input packets : 0
    Output packets: 0

  Logical interface vtep.32775 (Index 2038) (SNMP ifIndex 1744)
    Flags: Up SNMP-Traps 0x4000 Encapsulation: ENET2
    VXLAN Endpoint Type: Source, VXLAN Endpoint Address: 2001:db8::192:168:1:1, L2 Routing
Instance: default-switch, L3 Routing Instance: default
    Input packets : 0
    Output packets: 0

  Logical interface vtep.32776 (Index 829) (SNMP ifIndex 1775)
    Flags: Up SNMP-Traps 0x4000 Encapsulation: ENET2
    VXLAN Endpoint Type: Source, VXLAN Endpoint Address: 2001:db8::192:168:1:1, L3 Routing
Instance: default
    Input packets : 0
    Output packets: 0

  Logical interface vtep.32777 (Index 830) (SNMP ifIndex 1776)
    Flags: Up SNMP-Traps 0x4000 Encapsulation: ENET2
    VXLAN Endpoint Type: Shared Remote, VXLAN Endpoint Address: 2001:db8::192:168:1:2, L3
Routing Instance: default
    Input packets : 3873955
    Output packets: 12766
    Protocol eth-switch, MTU: Unlimited
      Flags: Is-Primary, Trunk-Mode
```

6. Verify that each VNI maps to the associated VXLAN tunnel.

For example, with an IPv4 Fabric:

```
user@leaf-1> show ethernet-switching vxlan-tunnel-end-point remote
                 0   192.168.1.1      lo0.0    0
 RVTEP-IP         IFL-Idx   NH-Id
 192.168.1.2      586       1782
    VNID          MC-Group-IP
    2000          0.0.0.0
    4000          0.0.0.0
    1000          0.0.0.0
    3000          0.0.0.0

...

RVTEP-IP         IFL-Idx   NH-Id
192.168.1.96     613       1820
    VNID          MC-Group-IP
    1000          0.0.0.0
    2000          0.0.0.0
    3000          0.0.0.0
    4000          0.0.0.0
```

Or for example, with an IPv6 Fabric:

```
user@leaf-1> show ethernet-switching vxlan-tunnel-end-point remote
Logical System Name       Id  SVTEP-IP              IFL   L3-Idx    SVTEP-Mode    ELP-SVTEP-
IP
<default>                 0   2001:db8::192:168:1:1 lo0.0 0
 RVTEP-IP              IFL-Idx     Interface    NH-Id   RVTEP-Mode  ELP-IP        Flags
 2001:db8::192:168:1:2 830         vtep.32777   22929   RNVE
 RVTEP-IP              L2-RTT         IFL-Idx   Interface      NH-Id   RVTEP-Mode  ELP-
IP        Flags
 2001:db8::192:168:1:2 MAC-VRF-1      675430409 vtep-532.32777 22929   RNVE
    VNID          MC-Group-IP
    1000          ::
    2000          ::
    3000          ::
    4000          ::
```

7. Verify that MAC addresses are learned through the VXLAN tunnels.

For example, with an IPv4 Fabric:

```
user@leaf-1> show ethernet-switching 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, P -Pinned MAC)

Logical system   : <default>
Routing instance : default-switch
 Bridging domain : VNI_1000+10, VLAN : 10, VNID : 1000
   MAC                 MAC      Logical          Remote VTEP
   address             flags    interface        IP address
   02:0c:10:01:02:04   DR       esi.1764         192.168.1.11 192.168.1.12 192.168.1.10
   02:0c:10:01:02:02   DR       esi.1771         192.168.1.6 192.168.1.4
   02:0c:10:01:02:03   DR       esi.1774         192.168.1.7

...

   0e:ad:10:01:00:60   D        vtep.32784       192.168.1.84
   0e:ad:10:01:00:30   D        vtep.32785       192.168.1.36
   0e:ad:10:01:00:48   D        vtep.32786       192.168.1.60
   0e:ad:10:01:00:4e   D        vtep.32788       192.168.1.66
   0e:ad:10:01:00:4c   D        vtep.32789       192.168.1.64
   0e:ad:10:01:00:36   D        vtep.32790       192.168.1.42

...

---(more)---
```

Or for example, with an IPv6 Fabric:

```
user@leaf-1> show ethernet-switching 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, P -Pinned MAC)

Logical system   : <default>
Routing instance : MAC-VRF-1
 Bridging domain : VNI_1000+10, VLAN : 10, VNID : 1000
   MAC                 MAC      Logical          Remote VTEP
   address             flags    interface        IP address
   00:00:5e:00:00:04   DRP      vtep-532.32777   2001:db8::192:168:1:2
   00:00:00:00:09:80   S,NM     vtep-532.32777   2001:db8::192:168:1:2
   c8:fe:6a:e4:2e:00   DRP      vtep-532.32777   2001:db8::192:168:1:2
```

8. Verify multihoming information of the gateway and the aggregated Ethernet interfaces.

   For example, with an IPv4 Fabric:

```
user@leaf-1> show ethernet-switching vxlan-tunnel-end-point esi
## Local AE link - QFX5100 leaf devices
ESI                              RTT                        VLNBH INH     ESI-IFL   LOC-IFL
#RVTEPs
00:00:00:00:00:00:51:00:00:01 default-switch          1768  131078   esi.1768   ae11.0,
2
    RVTEP-IP             RVTEP-IFL      VENH     MASK-ID   FLAGS
    192.168.1.2          vtep.32780     1782     1         2
    192.168.1.3          vtep.32772     1767     0         2
## Remote AE Link for QFX5110 leaf devices
ESI                              RTT                        VLNBH INH     ESI-IFL   LOC-IFL
#RVTEPs
00:00:00:00:00:00:51:10:00:01 default-switch          1771  131081   esi.1771
3
    RVTEP-IP             RVTEP-IFL      VENH     MASK-ID   FLAGS
    192.168.1.6          vtep.32771     1766     2         2
    192.168.1.4          vtep.32770     1765     1         2
    192.168.1.5          vtep.32774     1770     0         2
## Remote AE Link for QFX5200 leaf devices
ESI                              RTT                        VLNBH INH     ESI-IFL   LOC-IFL
#RVTEPs
00:00:00:00:00:00:52:00:00:01 default-switch          1774  131084   esi.1774
3
    RVTEP-IP             RVTEP-IFL      VENH     MASK-ID   FLAGS
    192.168.1.9          vtep.32778     1776     2         2
    192.168.1.8          vtep.32777     1775     1         2
    192.168.1.7          vtep.32776     1773     0         2
## Remote AE Link for QFX10002 leaf devices
ESI                              RTT                        VLNBH INH     ESI-IFL   LOC-IFL
#RVTEPs
00:00:00:00:00:01:00:00:00:01 default-switch          1764  131074   esi.1764
3
    RVTEP-IP             RVTEP-IFL      VENH     MASK-ID   FLAGS
    192.168.1.11         vtep.32775     1772     2         2
    192.168.1.12        vtep.32773     1769     1         2
    192.168.1.10        vtep.32769     1759     0         2

...
```

Or for example, with an IPv6 Fabric:

```
user@leaf-1> show ethernet-switching vxlan-tunnel-end-point esi
ESI                   RTT                      VLNBH INH     ESI-IFL   LOC-IFL   #RVTEPs
00:00:00:ff:00:01:00:03:00:05 MAC-VRF-1                36499 525338  esi.36499 ae5.0,
1      Aliasing
    RVTEP-IP               RVTEP-IFL      VENH    MASK-ID   FLAGS        MAC-COUNT
    2001:db8::191:168:1:2 vtep-532.32777 22929   0         2            0
```

9. Verify that the VXLAN tunnel from one leaf to another leaf is load balanced with equal cost multipathing (ECMP) over the underlay.

```
user@leaf-1> show route forwarding-table table default-switch extensive | find vtep.32770

Destination:  vtep.32770
  Route type: interface
  Route reference: 0                   Route interface-index: 576
  Multicast RPF nh index: 0
  P2mpidx: 0
  Flags: sent to PFE
  Nexthop:
  Next-hop type: composite             Index: 1765     Reference: 12
  Next-hop type: indirect              Index: 131076   Reference: 3
  Next-hop type: unilist               Index: 131193   Reference: 238
  Nexthop: 172.16.1.2
  Next-hop type: unicast               Index: 1791     Reference: 10
  Next-hop interface: ae1.0         Weight: 0x0
  Nexthop: 172.16.1.6
  Next-hop type: unicast               Index: 1794     Reference: 10
  Next-hop interface: ae2.0         Weight: 0x0
  Nexthop: 172.16.1.10
  Next-hop type: unicast               Index: 1758     Reference: 10
  Next-hop interface: ae3.0         Weight: 0x0
  Nexthop: 172.16.1.14
  Next-hop type: unicast               Index: 1795     Reference: 10
  Next-hop interface: ae4.0         Weight: 0x0
```

10. Verify that remote MAC addresses are reachable through ECMP.

    For example, with an IPv4 Fabric:

```
user@leaf-1> show route forwarding-table table default-switch extensive destination
02:0c:10:01:02:03/48
Routing table: default-switch.evpn-vxlan [Index 4]
Bridging domain: VNI_1000.evpn-vxlan [Index 3]
VPLS:
Enabled protocols: Bridging, ACKed by all peers, EVPN VXLAN,

Destination:  02:0c:10:01:02:03/48
  Learn VLAN: 0                        Route type: user
  Route reference: 0                   Route interface-index: 582
  Multicast RPF nh index: 0
  P2mpidx: 0
  IFL generation: 169                  Epoch: 0
  Sequence Number: 0                   Learn Mask: 0x400000000000000003000000000000000000000
  L2 Flags: control_dyn
  Flags: sent to PFE
  Nexthop:
  Next-hop type: composite             Index: 1773     Reference: 12
  Next-hop type: indirect              Index: 131085   Reference: 3
  Next-hop type: unilist               Index: 131193   Reference: 238
  Nexthop: 172.16.1.2
  Next-hop type: unicast               Index: 1791     Reference: 10
  Next-hop interface: ae1.0            Weight: 0x0
  Nexthop: 172.16.1.6
  Next-hop type: unicast               Index: 1794     Reference: 10
  Next-hop interface: ae2.0            Weight: 0x0
  Nexthop: 172.16.1.10
  Next-hop type: unicast               Index: 1758     Reference: 10
  Next-hop interface: ae3.0            Weight: 0x0
  Nexthop: 172.16.1.14
  Next-hop type: unicast               Index: 1795     Reference: 10
  Next-hop interface: ae4.0            Weight: 0x0
```

**NOTE**: Though the MAC address is reachable over multiple VTEP interfaces, QFX5100, QFX5110, QFX5120-32C, and QFX5200 switches do not support ECMP across the overlay because of a merchant ASIC limitation. Only the QFX10000 line of switches contain a custom Juniper Networks ASIC that supports ECMP across both the overlay and the underlay.

```
user@leaf-1> show ethernet-switching table vlan-id 10 | match 02:0c:10:01:02:03
   VNI_1000            02:0c:10:01:02:03   DR       esi.1774
00:00:00:00:00:00:52:00:00:01
user@leaf-1> show route forwarding-table table default-switch extensive destination
02:0c:10:01:02:03/48
Routing table: default-switch.evpn-vxlan [Index 9]
Bridging domain: VNI_1000.evpn-vxlan [Index 3]
VPLS:
Enabled protocols: Bridging, ACKed by all peers, EVPN VXLAN,

Destination:  02:0c:10:01:02:03/48
  Learn VLAN: 0                        Route type: user
  Route reference: 0                   Route interface-index: 550
  Multicast RPF nh index: 0
  P2mpidx: 0
  IFL generation: 0                    Epoch: 0
  Sequence Number: 0                   Learn Mask: 0x400000000000000001000000000000000000
  L2 Flags: control_dyn, esi
  Flags: sent to PFE
  Next-hop type: indirect              Index: 2097173  Reference: 5
  Nexthop:
  Next-hop type: composite             Index: 1947     Reference: 2
  Nexthop:
  Next-hop type: composite             Index: 1948     Reference: 8
  Next-hop type: indirect              Index: 2097174  Reference: 3
  Next-hop type: unilist               Index: 2097280  Reference: 241
  Nexthop: 172.16.10.2
  Next-hop type: unicast               Index: 1950     Reference: 11
  Next-hop interface: ae1.0            Weight: 0x0
  Nexthop: 172.16.10.6
  Next-hop type: unicast               Index: 1956     Reference: 10
  Next-hop interface: ae2.0            Weight: 0x0
  Nexthop: 172.16.10.10
  Next-hop type: unicast               Index: 1861     Reference: 10
  Next-hop interface: ae3.0            Weight: 0x0
  Nexthop: 172.16.10.14
  Next-hop type: unicast               Index: 1960     Reference: 10
  Next-hop interface: ae4.0            Weight: 0x0
```

Or for example, with an IPv6 Fabric:

```
user@leaf-1> show route forwarding-table table MAC-VRF-1 extensive | find c8:fe:6a:e4:2e:00

Destination:  c8:fe:6a:e4:2e:00/48
  Learn VLAN: 0                         Route type: user
  Route reference: 0                    Route interface-index: 1641
  Multicast RPF nh index: 0
  P2mpidx: 0
  IFL generation: 0                     Epoch: 0
  Sequence Number: 0                    Learn Mask:
0x400000000000000000000000000000000000000
  L2 Flags: control_dyn
  Flags: sent to PFE
  Nexthop:
  Next-hop type: composite              Index: 22929    Reference: 8358
  Next-hop type: indirect               Index: 524287   Reference: 3
  Next-hop type: unilist                Index: 524286   Reference: 531
  Nexthop: 2001:db8::173:16:1:1
  Next-hop type: unicast                Index: 22928    Reference: 6
  Next-hop interface: ae1.0             Weight: 0x0
  Nexthop: 2001:db8::173:16:2:1
  Next-hop type: unicast                Index: 12290    Reference: 6
  Next-hop interface: ae2.0             Weight: 0x0
```

11. Verify which device is the Designated Forwarder (DF) for broadcast, unknown, and multicast (BUM) traffic coming from the VTEP tunnel.

    For example, with an IPv4 Fabric:

    **NOTE**: Because the DF IP address is listed as 192.168.1.2, Leaf 2 is the DF.

```
user@leaf-1> show evpn instance esi 00:00:00:00:00:00:51:00:00:01 designated-forwarder
Instance: default-switch
  Number of ethernet segments: 12
    ESI: 00:00:00:00:00:00:51:00:00:01
      Designated forwarder: 192.168.1.2
```

Or, for example, with an IPv4 Fabric:

**NOTE**: Because the DF IPv6 address is listed as 2001:db8::192:168:1:1, Leaf 1 is the DF.

```
user@leaf-1> show evpn instance esi 00:00:00:ff:00:01:00:03:00:05 designated-forwarder
Instance: MAC-VRF-1
  Number of ethernet segments: 2
    ESI: 00:00:00:ff:00:01:00:03:00:05
      Designated forwarder: 2001:db8::192:168:1:1
```

### SEE ALSO

*EVPN Overview*

*Understanding VXLANs*

*Understanding EVPN with VXLAN Data Plane Encapsulation*

*Configuring Aggregated Ethernet LACP (CLI Procedure)*

## Bridged Overlay — Release History

Table 4 on page 141 provides a history of all of the features in this section and their support within this reference design.

**Table 4: Bridged Overlay in the Cloud Data Center Reference Design– Release History**

| Release | Description |
|---|---|
| 19.1R2 | QFX10002-60C and QFX5120-32C switches running Junos OS Release 19.1R2 and later releases in the same release train support all features documented in this section. |
| 18.4R2 | QFX5120-48Y switches running Junos OS Release 18.4R2 and later releases in the same release train support all features documented in this section. |
| 18.1R3-S3 | QFX5110 switches running Junos OS Release 18.1R3-S3 and later releases in the same release train support all features documented in this section. |

**Table 4: Bridged Overlay in the Cloud Data Center Reference Design– Release History *(Continued)***

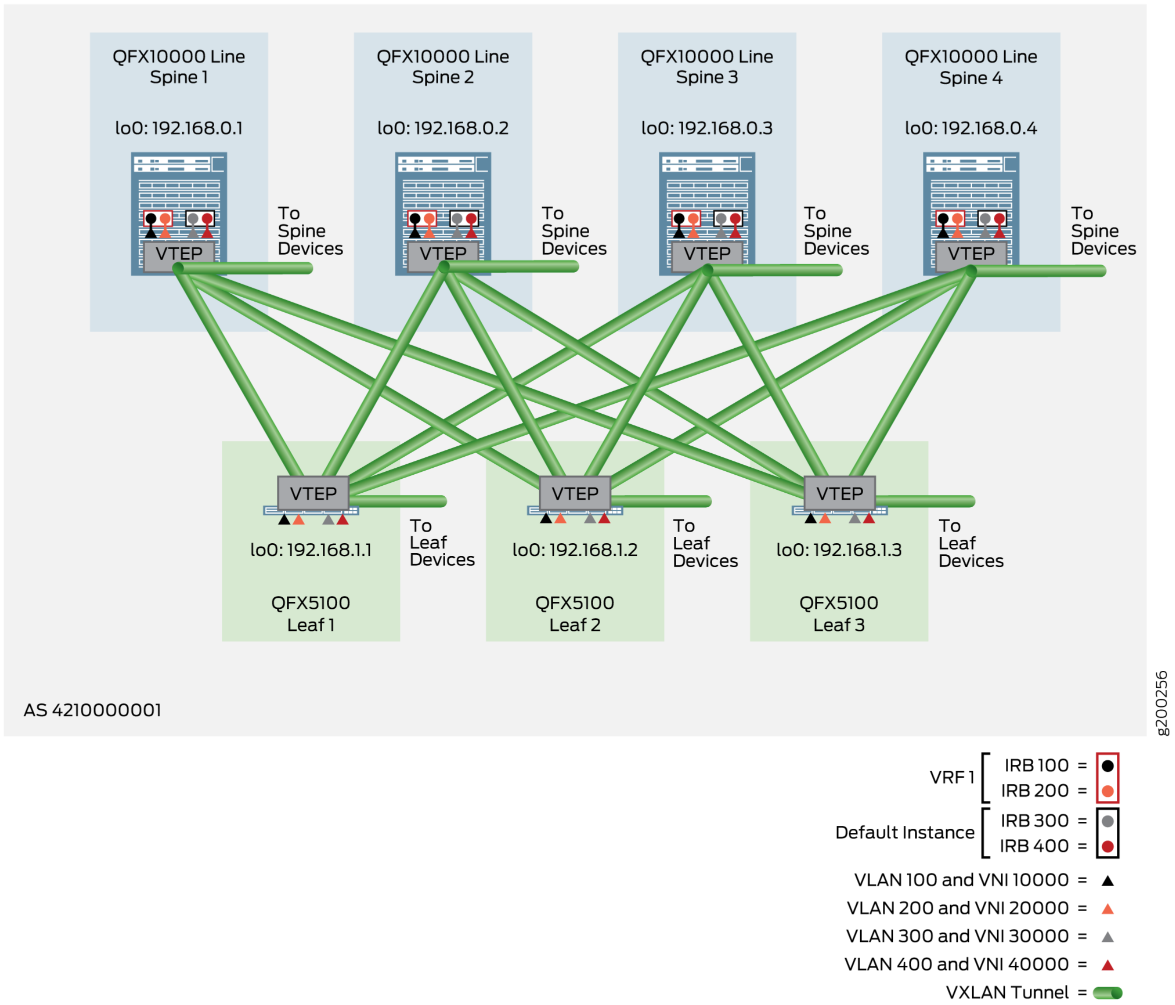
| Release | Description |
|---|---|
| 17.3R3-S2 | All devices in the reference design that support Junos OS Release 17.3R3-S2 and later releases in the same release train also support all features documented in this section. |

## Centrally-Routed Bridging Overlay Design and Implementation

**IN THIS SECTION**

A *centrally-routed bridging (CRB) overlay* performs routing at a central location in the EVPN network as shown in Figure 42 on page 143, In this example, IRB interfaces are configured in the overlay at each spine device to route traffic between the VLANs that originate at the leaf devices and end systems. For an overview of CRB overlays, see the Centrally-Routed Bridging Overlay section in "Data Center Fabric Blueprint Architecture Components" on page 9.

**Figure 42: CRB Overlay**



The following sections provide the detailed steps of how to implement a CRB overlay:

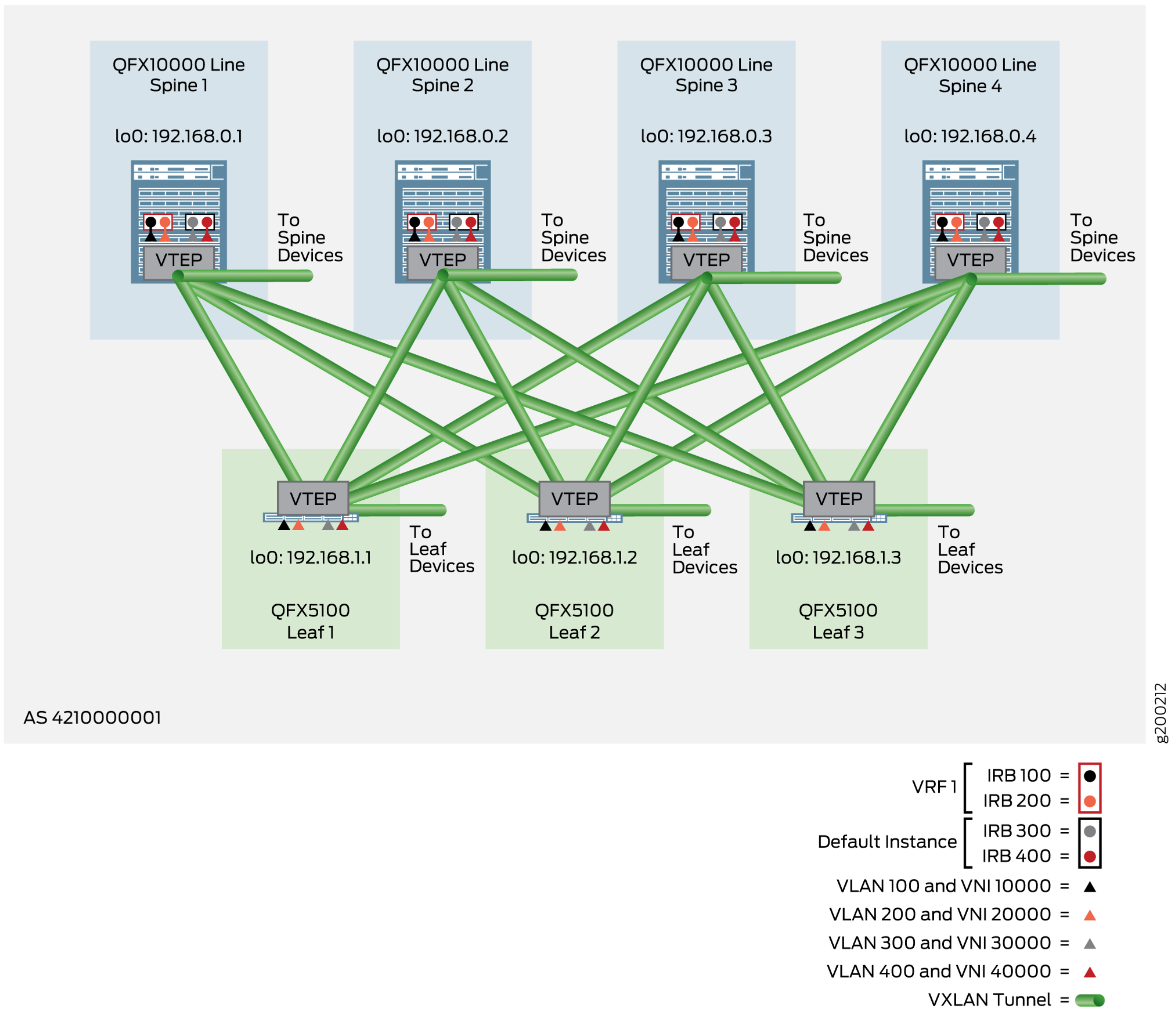
## Configuring a VLAN-Aware CRB Overlay in the Default Instance

**IN THIS SECTION**

The VLAN-aware CRB overlay is a basic overlay supported on all platforms included in this reference design. It uses the simplest VLAN-aware method to enable a single, default switching instance that supports up to 4094 VLANs.

As shown in Figure 43 on page 145, you configure VLANs at the leaf devices, and IRB interfaces for routing at the spine devices. Such configuration is placed in the default switching instance at the `[edit vlans]`, `[edit interfaces]`, `[edit protocols evpn]`, and `[edit switch-options]` hierarchy levels. Routing instances are not required for this overlay style, but can be implemented as an option depending on the needs of your network.

**Figure 43: VLAN-Aware CRB Overlay**



When you implement this style of overlay on a spine device, you:

- Configure IRB interfaces to route traffic between Ethernet virtual network instances.
- Set virtual gateway addresses.
- Add VXLAN features to optimize traffic paths.
- Configure EVPN with VXLAN encapsulation in the default switching instance or in a routing instance.
- Set the loopback interface as the VTEP source interface.
- Configure route distinguishers and route targets to direct traffic to peers.
- Map VLANs to VNIs.

When you implement this style of overlay on a leaf device, you:

- Configure Ethernet Segment Identifier (ESI) settings.
- Enable EVPN with VXLAN encapsulation in the default switching instance.
- Establish route targets and route distinguishers.
- Map VLANs to VNIs.

For an overview of VLAN-aware CRB overlays, see the Centrally-Routed Bridging Overlay section in "Data Center Fabric Blueprint Architecture Components" on page 9.

If you need to implement more than 4094 VLANs, you can use a CRB overlay with virtual switches (available on switches in the QFX10000 line) or MAC-VRF instances. See "Configuring a VLAN-Aware CRB Overlay with Virtual Switches or MAC-VRF Instances" on page 175. With MAC-VRF instances, you expand your options to either isolate traffic between tenant systems or to enable routing and forwarding between tenant systems.

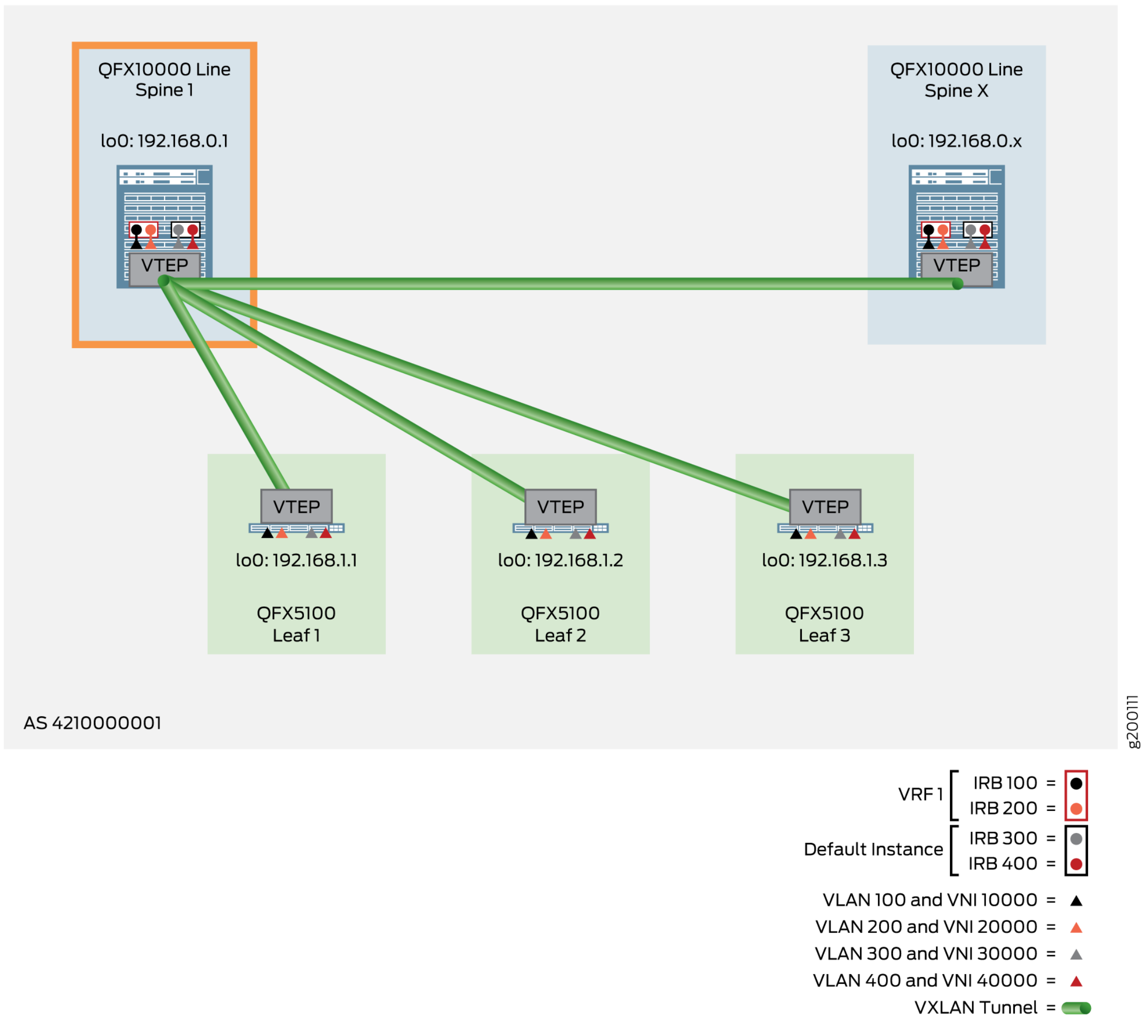
The following sections provide the detailed steps of how to configure and verify the VLAN-aware CRB overlay in the default switching instance:

## Configuring a VLAN-Aware CRB Overlay in the Default Instance on the Spine Device

To configure a VLAN-aware CRB overlay in the default switching instance on a spine device, perform the following:

**NOTE**: The following example shows the configuration for Spine 1, as shown in Figure 44 on page 147.

**Figure 44: VLAN-Aware CRB Overlay in the Default Instance – Spine Device**



1. Ensure the IP fabric underlay is in place. To configure an IP fabric on a spine device, see "IP Fabric Underlay Network Design and Implementation" on page 80.
2. Confirm that your IBGP overlay is up and running. To configure an IBGP overlay on your spine devices, see "Configure IBGP for the Overlay" on page 96.
3. Configure the VTEP tunnel endpoint as the loopback address, and add a route distinguisher and a route target (target:64512:1111). Also, keep your configuration simple by using the auto route target option, which uses one target for both import and export.

   *Spine 1*:

```
set switch-options vtep-source-interface lo0.0
set switch-options route-distinguisher 192.168.0.1:1
set switch-options vrf-target target:64512:1111
set switch-options vrf-target auto
```

4. Configure IRB interfaces for each VNI and the corresponding virtual gateway address (which uses .254 in the 4th octet for each prefix). Include VXLAN features, such as `proxy-macip-advertisement` and `virtual-gateway-accept-data`, to improve performance and manageability.

   **NOTE:**

   - We strongly recommend you set the `proxy-macip-advertisement` option on the spine devices in a CRB fabric. This option enables one central gateway (the spine device) to send both MAC address and IP address information (ARP entries) that it learns locally to the other central gateways. This operation is called ARP synchronization. Setting this option ensures that ARP synchronization happens efficiently if any leaf devices in the fabric advertise only the MAC addresses in their EVPN Type 2 route advertisements for their connected hosts. This setting improves convergence times and traffic handling in the fabric.

   - You must configure both the `virtual-gateway-accept-data` statement and the preferred IPv4 and IPv6 addresses to use the ping operation and verify connectivity to the virtual gateway IP address from the end system.

   *Spine 1*:

```
set interfaces irb unit 100 family inet address 10.1.0.1/24 virtual-gateway-address 10.1.0.254
set interfaces irb unit 100 family inet address 10.1.0.1/24 preferred
set interfaces irb unit 100 proxy-macip-advertisement
set interfaces irb unit 100 virtual-gateway-accept-data
set interfaces irb unit 100 family inet6 address 2001:db8::10:1:0:1/112 virtual-gateway-
address 2001:db8::10:1:0:254
set interfaces irb unit 100 family inet6 address fe80::10:1:0:254/112
set interfaces irb unit 200 family inet address 10.1.1.1/24 virtual-gateway-address 10.1.1.254
set interfaces irb unit 200 family inet address 10.1.1.1/24 preferred
set interfaces irb unit 200 proxy-macip-advertisement
set interfaces irb unit 200 virtual-gateway-accept-data
set interfaces irb unit 200 family inet6 address 2001:db8::10:1:1:1/112 virtual-gateway-
address 2001:db8::10:1:1:254
set interfaces irb unit 200 family inet6 address fe80::10:1:1:254/112
set interfaces irb unit 300 family inet address 10.1.2.1/24 virtual-gateway-address 10.1.2.254
set interfaces irb unit 300 family inet address 10.1.2.1/24 preferred
set interfaces irb unit 300 proxy-macip-advertisement
set interfaces irb unit 300 virtual-gateway-accept-data
set interfaces irb unit 300 family inet6 address 2001:db8::10:1:2:1/112 virtual-gateway-
address 2001:db8::10:1:2:254
set interfaces irb unit 300 family inet6 address fe80::10:1:2:254/112
set interfaces irb unit 400 family inet address 10.1.3.1/24 virtual-gateway-address 10.1.3.254
set interfaces irb unit 400 family inet address 10.1.3.1/24 preferred
set interfaces irb unit 400 proxy-macip-advertisement
set interfaces irb unit 400 virtual-gateway-accept-data
set interfaces irb unit 400 family inet6 address 2001:db8::10:1:3:1/112 virtual-gateway-
address 2001:db8::10:1:3:254
set interfaces irb unit 400 family inet6 address fe80::10:1:3:254/112
```

5. Configure a secondary logical unit on the loopback interface for the default switching instance.

   *Spine 1*:

```
set interfaces lo0 unit 1 family inet address 192.168.0.101/32
```

6. Configure EVPN with VXLAN encapsulation. Include the `no-gateway-community` option to advertise the virtual gateway and IRB MAC addresses to the EVPN peer devices so that Ethernet-only PE devices can learn these MAC addresses.

   *Spine 1*:

```
set protocols evpn encapsulation vxlan
set protocols evpn default-gateway no-gateway-community
set protocols evpn extended-vni-list all
```

7. Configure mapping between VLANs and VXLAN VNIs.

   *Spine 1*:

```
set vlans VNI_10000 vlan-id 100
set vlans VNI_10000 l3-interface irb.100
set vlans VNI_10000 vxlan vni 10000
set vlans VNI_20000 vlan-id 200
set vlans VNI_20000 l3-interface irb.200
set vlans VNI_20000 vxlan vni 20000
set vlans VNI_30000 vlan-id 300
set vlans VNI_30000 l3-interface irb.300
set vlans VNI_30000 vxlan vni 30000
set vlans VNI_40000 vlan-id 400
set vlans VNI_40000 l3-interface irb.400
set vlans VNI_40000 vxlan vni 40000
```

8. Configure a routing instance named VRF 1, and map IRB interfaces irb.100 (VNI 10000) and irb.200 (VNI 20000) to this instance.

   **NOTE**: Because the irb.300 (VNI 30000) and irb.400 (VNI 40000) interfaces are not configured inside a routing instance, they are part of the default switching instance for the spine devices. The end result of your configuration should match the diagram shown in Figure 44 on page 147.

   *Spine 1*:

```
set routing-instances VRF_1 instance-type vrf
set routing-instances VRF_1 interface irb.100
set routing-instances VRF_1 interface irb.200
set routing-instances VRF_1 interface lo0.1
set routing-instances VRF_1 route-distinguisher 192.168.0.1:100
set routing-instances VRF_1 vrf-target target:62273:10000
```

## Verifying the VLAN-Aware CRB Overlay in the Default Instance for the Spine Device

Issue the following commands to verify that the overlay is working properly on your spine devices:

1. Verify the IRB interfaces are operational for both IPv4 and IPv6.

```
user@spine-1> show interfaces terse irb
Interface               Admin Link Proto    Local                 Remote
irb                     up    up
irb.100                 up    up   inet     10.1.0.1/24
                                   inet6    2001:db8::10:1:0:1/112
                                            fe80::10:1:0:254/112
irb.200                 up    up   inet     10.1.1.1/24
                                   inet6    2001:db8::10:1:1:1/112
                                            fe80::10:1:1:254/112
irb.300                 up    up   inet     10.1.2.1/24
                                   inet6    2001:db8::10:1:2:1/112
                                            fe80::10:1:2:254/112
irb.400                 up    up   inet     10.1.3.1/24
                                   inet6    2001:db8::10:1:3:1/112
                                            fe80::10:1:3:254/112
```

2. Verify that the VTEP interfaces are up.

```
user@spine-1> show interfaces terse vtep
Interface               Admin Link Proto    Local                 Remote
vtep                    up    up
vtep.32768              up    up
vtep.32769              up    up   eth-switch
vtep.32770              up    up   eth-switch
vtep.32771              up    up   eth-switch
vtep.32772              up    up   eth-switch
...
vtep.32804              up    up   eth-switch
---(more)---

user@spine-1> show interfaces terse vtep | match eth-switch | count
Count: 109 lines
```

3. Verify the endpoint destination IP address for the VTEP interfaces. The spine devices display their VTEPs as loopback addresses in the range of 192.168.0.*x* (1 - 4) and the leaf devices display their VTEPs as loopback addresses in the range of 192.168.1.*x* (1 - 96).

```
user@spine-1> show interfaces vtep
Physical interface: vtep, Enabled, Physical link is Up
  Interface index: 240, SNMP ifIndex: 504
  Type: Software-Pseudo, Link-level type: VxLAN-Tunnel-Endpoint, MTU: Unlimited, Speed: 
Unlimited
  Device flags   : Present Running
  Link type      : Full-Duplex
  Link flags     : None
  Last flapped   : Never
    Input packets : 0
    Output packets: 0

  Logical interface vtep.32768 (Index 670) (SNMP ifIndex 505)
    Flags: Up SNMP-Traps 0x4000 Encapsulation: ENET2
    VXLAN Endpoint Type: Source, VXLAN Endpoint Address: 192.168.0.1, L2 Routing Instance: 
default-switch, L3 Routing Instance: default
    Input packets : 0
    Output packets: 0

...
```

```
  Logical interface vtep.32771 (Index 802) (SNMP ifIndex 536)
    Flags: Up SNMP-Traps Encapsulation: ENET2
    VXLAN Endpoint Type: Remote, VXLAN Endpoint Address: 192.168.1.4, L2 Routing Instance: 
default-switch, L3 Routing Instance: default
    Input packets : 1979
    Output packets: 9867
    Protocol eth-switch, MTU: Unlimited
---(more)---
```

4. Verify that the spine device has all the routes to the leaf devices.

```
user@spine-2> show route 192.168.1.1

inet.0: 446 destinations, 19761 routes (446 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

192.168.1.1/32     *[BGP/170] 00:06:29, localpref 100
                      AS path: 4200000011 I, validation-state: unverified
                    > to 172.16.1.5 via ae1.0
                    [BGP/170] 00:06:22, localpref 100
                      AS path: 4200000023 4200000004 4200000011 I, validation-state: 
unverified
                    > to 172.16.13.5 via ae13.0

                     ...

                     [BGP/170] 00:06:18, localpref 100
                      AS path: 4200000032 4200000004 4200000011 I, validation-state: 
unverified
                    > to 172.16.22.5 via ae22.0
---(more)---
```

5. Verify that each end system resolves the virtual gateway MAC address for a subnet using the gateway IRB address on the central gateways (spine devices).

```
user@spine-1> show arp no-resolve vpn VRF_1
MAC Address       Address         Interface         Flags
06:4b:8c:cd:13:f8 10.1.0.2        irb.100 [vtep.32796]     none  ## Spine 2 IRB interface
06:4b:8c:cd:c4:38 10.1.0.3        irb.100 [vtep.32878]     none  ## Spine 3 IRB interface
06:38:e1:6f:30:29 10.1.0.4        irb.100 [vtep.32821]     none  ## Spine 4 IRB 
interface02:0c:10:01:02:01 10.1.0.201      irb.100 [.local..11]     none  ## End system 
behind the QFX5100s
```

```
02:0c:10:01:02:02 10.1.0.202      irb.100 [.local..11]     none  ## End system behind the
QFX5110s
02:0c:10:01:02:03 10.1.0.203      irb.100 [.local..11]     none  ## End system behind the
QFX5200s
02:0c:10:01:02:04 10.1.0.204      irb.100 [.local..11]     none  ## End system behind the
QFX10002s
00:00:5e:00:01:01 10.1.0.254      irb.100                  permanent published gateway
                                                           ## Virtual gateway IP and MAC
address
06:4b:8c:cd:13:f8 10.1.1.2        irb.200 [vtep.32796]     none
06:4b:8c:cd:c4:38 10.1.1.3        irb.200 [vtep.32878]     none
06:38:e1:6f:30:29 10.1.1.4        irb.200 [vtep.32821]     none
0e:ad:10:02:00:01 10.1.1.101      irb.200 [vtep.32776]     none
user@spine-1> show ipv6 neighbors
IPv6 Address                 Linklayer Address  State       Exp Rtr Secure Interface
2001:db8::10:1:0:2           06:4b:8c:cd:13:f8  stale       325 no  no      irb.100
[vtep.32796]
2001:db8::10:1:0:3           06:4b:8c:cd:c4:38  stale       514 yes no      irb.100
[vtep.32878]
2001:db8::10:1:0:4           06:38:e1:6f:30:29  stale       326 no  no      irb.100
[vtep.32821]
2001:db8::10:1:0:201         02:0c:10:01:02:01  stale       1114 no no      irb.100
[.local..11]
2001:db8::10:1:0:202         02:0c:10:01:02:02  stale       443 no  no      irb.100
[.local..11]
2001:db8::10:1:0:203         02:0c:10:01:02:03  stale       853 no  no      irb.100
[.local..11]
2001:db8::10:1:0:204         02:0c:10:01:02:04  stale       1181 no no      irb.100
[.local..11]
2001:db8::10:1:0:254         00:00:5e:00:02:01  reachable   0   no  no      irb.100
2001:db8::10:1:1:2           06:4b:8c:cd:13:f8  stale       325 no  no      irb.200
[vtep.32796]
2001:db8::10:1:1:3           06:4b:8c:cd:c4:38  stale       514 yes no      irb.200
[vtep.32878]
2001:db8::10:1:1:4           06:38:e1:6f:30:29  stale       326 no  no      irb.200
[vtep.32821]
2001:db8::10:1:1:201         02:0c:10:02:02:01  stale       1121 no no      irb.200
[.local..11]
2001:db8::10:1:1:202         02:0c:10:02:02:02  stale       423 no  no      irb.200
[.local..11]
2001:db8::10:1:1:203         02:0c:10:02:02:03  stale       1081 no no      irb.200
[.local..11]
```

```
2001:db8::10:1:1:204        02:0c:10:02:02:04  stale       1167 no no      irb.200
[.local..11]
```

6. Verify the switching table for VNI 10000 to see entries for end systems and the other spine devices.

```
user@spine-1> show ethernet-switching table vlan-id 100

MAC flags (S - static MAC, D - dynamic MAC, L - locally learned, P - Persistent static
           SE - statistics enabled, NM - non configured MAC, R - remote PE MAC, O - ovsdb MAC)



Ethernet switching table : 105 entries, 105 learned
Routing instance : default-switch
   Vlan                MAC                 MAC      Logical                Active
   name                address             flags    interface              source
   VNI_10000           00:00:5e:00:01:01   DR       esi.2453
05:19:17:f3:41:00:00:27:10:00

## Entries for the spine devices
   VNI_10000           06:4b:8c:cd:13:f8   D        vtep.32796
192.168.0.2
   VNI_10000           06:4b:8c:cd:c4:38   D        vtep.32878
192.168.0.3
   VNI_10000           06:38:e1:6f:30:29   D        vtep.32821
192.168.0.4
## The next four MAC addresses belong to the end systems
connected to Leaf 1 - 3 (QFX5100), Leaf 4-6 (QFX5110), Leaf 7-9 (QFX5200),
and Leaf 10-12 (QFX10002).
   VNI_10000           02:0c:10:01:02:01   DR       esi.2443
00:00:00:00:00:00:51:00:00:01
   VNI_10000           02:0c:10:01:02:02   DR       esi.2497
00:00:00:00:00:00:51:10:00:01
   VNI_10000           02:0c:10:01:02:03   DR       esi.2427
00:00:00:00:00:00:52:00:00:01
   VNI_10000           02:0c:10:01:02:04   DR       esi.2610
00:00:00:00:00:01:00:00:00:01
   ...
   VNI_10000           0e:ad:10:01:00:02   D        vtep.32814             192.168.1.96
```

7. Verify MAC address and ARP information learned from the leaf devices over the control plane.

```
user@spine-1> show evpn database mac-address 02:0c:10:01:02:01 extensive
Instance: default-switch

VN Identifier: 10000, MAC address:: 02:0c:10:01:02:01
  Source: 00:00:00:00:00:00:51:00:00:01, Rank: 1, Status: Active
    Remote origin: 192.168.1.2  ## Leaf 2 and Leaf 3 advertised this route
    Remote origin: 192.168.1.3
    Timestamp: Jul 13 23:35:37 (0x59686639)
    State: <Remote-To-Local-Adv-Done>
    IP address: 10.1.0.201  ## MAC Address + IP
      Flags: <Proxy>
      Remote origin: 192.168.1.2
      Remote origin: 192.168.1.3
    IP address: 2001:db8::10:1:0:201   ## MAC Address + IPv6
      Remote origin: 192.168.1.2
      Remote origin: 192.168.1.3    History db:
      Time                  Event
      Jul 13 23:35:38 2017  Applying remote state to peer 192.168.1.2
      Jul 13 23:35:38 2017  Remote peer 192.168.1.2 updated
      Jul 13 23:35:38 2017  MAC+IP not updated, source l2ald is not owner (type2)
      Jul 13 23:35:38 2017  Updated
      Jul 13 23:35:38 2017  No change to MAC state
      Jul 13 23:35:38 2017  Applying remote state to peer 192.168.1.3
      Jul 13 23:35:38 2017  Remote peer 192.168.1.3 updated
      Jul 13 23:35:38 2017  MAC+IP not updated, source l2ald is not owner (type2)
      Jul 13 23:35:38 2017  Updated
      Jul 13 23:35:38 2017  No change to MAC state
```

8. Verify the remote VXLAN tunnel end points.

```
user@spine-1> show ethernet-switching vxlan-tunnel-end-point remote
Logical System Name       Id  SVTEP-IP         IFL   L3-Idx
<default>                 0   192.168.0.1      lo0.0    0
 RVTEP-IP         IFL-Idx   NH-Id
 192.168.1.1      827       2444
    VNID          MC-Group-IP
    10000         0.0.0.0
    20000         0.0.0.0
    30000         0.0.0.0
    40000         0.0.0.0
 RVTEP-IP         IFL-Idx   NH-Id
...
RVTEP-IP         IFL-Idx   NH-Id
 192.168.1.96      812       2428
    VNID          MC-Group-IP
    10000         0.0.0.0
    20000         0.0.0.0
    30000         0.0.0.0
    40000         0.0.0.0
```

9. Verify that MAC addresses are learned through the VXLAN tunnel.

```
user@spine-1> show ethernet-switching 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, P -Pinned MAC)

Logical system   : <default>
Routing instance : default-switch
 Bridging domain : VNI_10000+100, VLAN : 100, VNID : 10000
   MAC                 MAC      Logical          Remote VTEP
   address             flags    interface        IP address
   02:0c:10:01:02:03   DR       esi.2427         192.168.1.8 192.168.1.7 192.168.1.9
   02:0c:10:01:02:01   DR       esi.2443         192.168.1.2 192.168.1.3
## This next entry shows that the virtual gateway MAC address
of 00:00:5e:00:01:01 has been learned by
the other spine devices.
   00:00:5e:00:01:01   DR       esi.2453         192.168.0.3 192.168.0.4 192.168.0.2
   02:0c:10:01:02:02   DR       esi.2497         192.168.1.6 192.168.1.4
   02:0c:10:01:02:04   DR       esi.2610         192.168.1.12 192.168.1.10 192.168.1.11
   06:4b:8c:cd:13:f8   D        vtep.32796       192.168.0.2
---(more)---
```

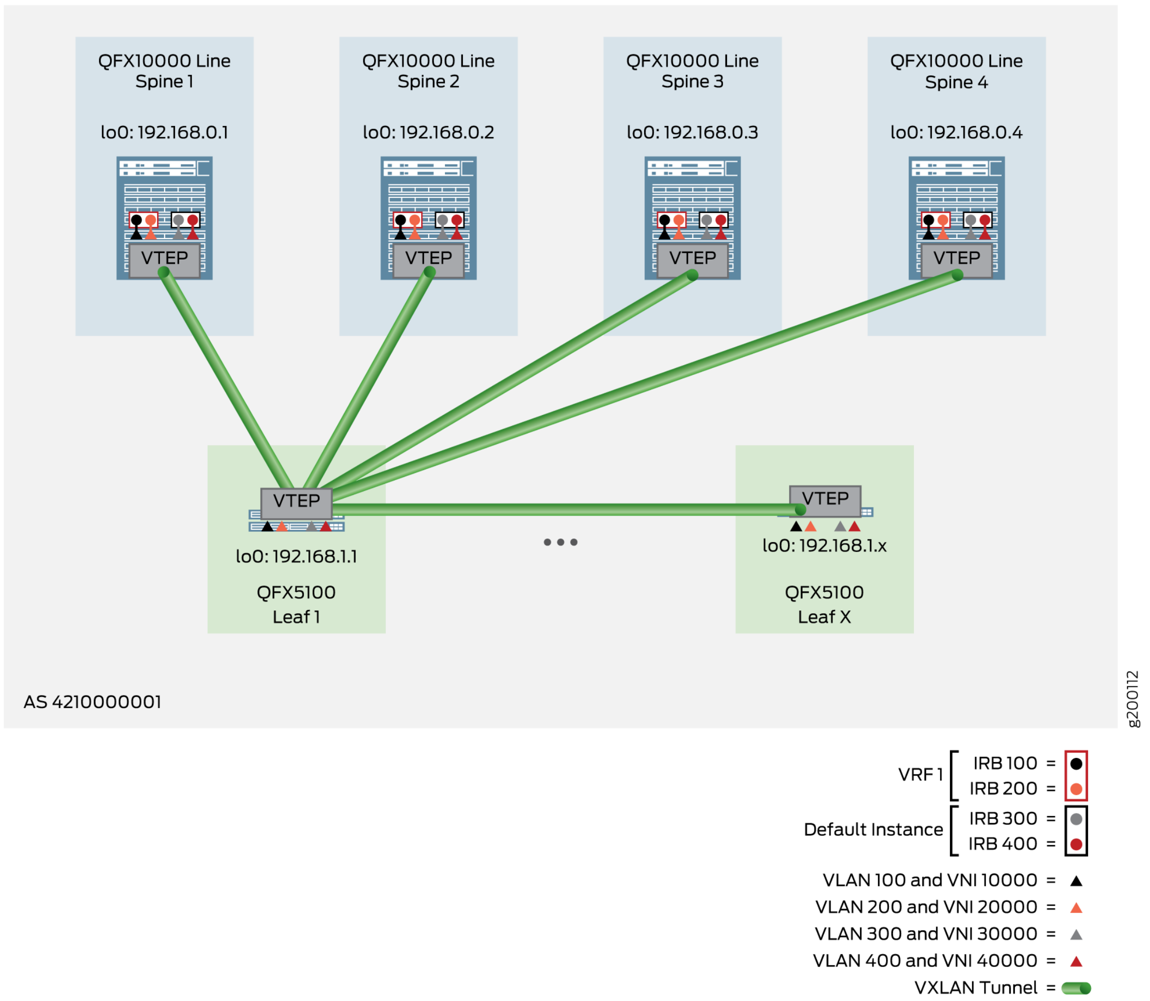
## Configuring a VLAN-Aware CRB Overlay in the Default Instance on the Leaf Device

To configure a VLAN-aware CRB overlay in the default switching instance on a leaf device, perform the following:

**NOTE:**

- The following example shows the configuration for Leaf 1, as shown in Figure 45 on page 157.

**Figure 45: VLAN-Aware CRB Overlay in the Default Instance – Leaf Device**



1. Ensure the IP fabric underlay is in place. To configure an IP fabric on a leaf device, see "IP Fabric Underlay Network Design and Implementation" on page 80.
2. Confirm that your IBGP overlay is up and running. To configure an IBGP overlay on your leaf device, see "Configure IBGP for the Overlay" on page 96.
3. Configure the EVPN protocol with VXLAN encapsulation, and specify the VTEP source interface (in this case, the loopback interface of the leaf device).

*Leaf 1*:

```
set protocols evpn encapsulation vxlan
set protocols evpn extended-vni-list all
set switch-options vtep-source-interface lo0.0
```

4. Define an EVPN route target and route distinguisher, and use the `auto` option to derive route targets automatically. Setting these parameters specifies how the routes are imported and exported. The import and export of routes from a routing or bridging table is the basis for dynamic overlays. In this case, members of the global BGP community with a route target of target:64512:1111 participate in the exchange of EVPN-VXLAN information.

   *Leaf 1*:

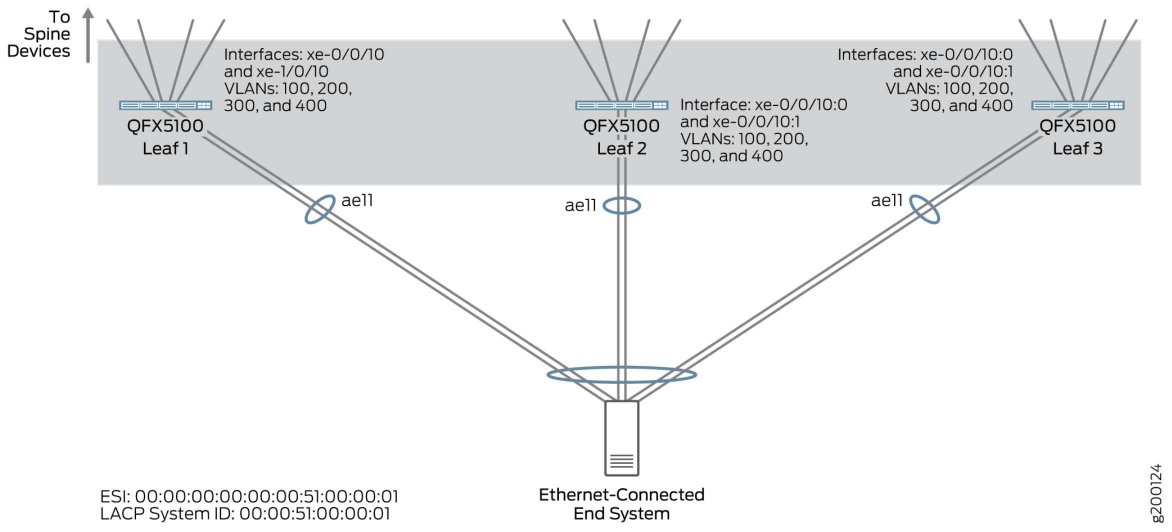
```
set switch-options route-distinguisher 192.168.1.1:1
set switch-options vrf-target target:64512:1111
set switch-options vrf-target auto
```

5. Configure ESI settings on all similar leaf devices. Because the end systems in this reference design are multihomed to three leaf devices per device type cluster (such as QFX5100), you must configure the same ESI identifier and LACP system identifier on all three leaf devices for each unique end system. Unlike other topologies where you would configure a different LACP system identifier per leaf device and have VXLAN select a single designated forwarder, use the same LACP system identifier to allow the 3 leaf devices to appear as a single LAG to a multihomed end system. In addition, use the same aggregated Ethernet interface number for all ports included in the ESI.

   The configuration for Leaf 1 is shown below, but you must replicate this configuration on both Leaf 2 and Leaf 3 per the topology shown in Figure 46 on page 159.

   **TIP**: When you create an ESI number, always set the high order octet to 00 to indicate the ESI is manually created. The other 9 octets can be any hexadecimal value from 00 to FF.

**Figure 46: ESI Topology for Leaf 1, Leaf 2, and Leaf 3**


To Spine Devices
Interfaces: xe-0/0/10 and xe-1/0/10 VLANs: 100, 200, 300, and 400
QFX5100 Leaf 1
QFX5100 Leaf 2
Interface: xe-0/0/10:0 and xe-0/0/10:1 VLANs: 100, 200, 300, and 400
Interfaces: xe-0/0/10:0 and xe-0/0/10:1 VLANs: 100, 200, 300, and 400
QFX5100 Leaf 3
ae11
ae11
ae11
ESI: 00:00:00:00:00:00:51:00:00:01
LACP System ID: 00:00:51:00:00:01
Ethernet-Connected End System
g200124

*Leaf 1*:

```
set interfaces ae11 esi 00:00:00:00:00:00:51:00:00:01
set interfaces ae11 esi all-active
set interfaces ae11 aggregated-ether-options lacp system-id 00:00:51:00:00:01
set interfaces xe-0/0/10 ether-options 802.3ad ae11
set interfaces xe-1/0/10 ether-options 802.3ad ae11
```

6. Configure VLANs and map them to VNIs. This step enables the VLANs to participate in VNIs across the EVPN-VXLAN domain.

   *Leaf 1*:

```
set vlans VNI_10000 vlan-id 100
set vlans VNI_10000 vxlan vni 10000
set vlans VNI_20000 vlan-id 200
set vlans VNI_20000 vxlan vni 20000
set vlans VNI_30000 vlan-id 300
set vlans VNI_30000 vxlan vni 30000
set vlans VNI_40000 vlan-id 400
set vlans VNI_40000 vxlan vni 40000
```

## Verifying the VLAN-Aware CRB Overlay in the Default Instance for the Leaf Device

Issue the following commands to verify that the overlay is working properly on your leaf devices:

1. Verify the interfaces are operational.

```
user@leaf-1> show interfaces terse | match ae.*
xe-0/0/10.0             up    up   aenet    --> ae11.0
et-0/0/48.0             up    up   aenet    --> ae1.0
et-0/0/49.0             up    up   aenet    --> ae2.0
et-0/0/50.0             up    up   aenet    --> ae3.0
et-0/0/51.0             up    up   aenet    --> ae4.0
xe-1/0/10.0             up    up   aenet    --> ae11.0
et-1/0/48.0             up    up   aenet    --> ae1.0
et-1/0/49.0             up    up   aenet    --> ae2.0
et-1/0/50.0             up    up   aenet    --> ae3.0
et-1/0/51.0             up    up   aenet    --> ae4.0
ae1                     up    up  ## To Spine 1
ae1.0                   up    up   inet     172.16.1.1/30
ae2                     up    up  ## To Spine 2
ae2.0                   up    up   inet     172.16.1.5/30
ae3                     up    up  ## To Spine 3
ae3.0                   up    up   inet     172.16.1.9/30
ae4                     up    up  ## To Spine 4
ae4.0                   up    up   inet     172.16.1.13/30
ae11                    up    up  ## To End System
ae11.0                  up    up   eth-switch
user@leaf-1> show lacp interfaces
Aggregated interface: ae1
    LACP state:       Role   Exp   Def  Dist  Col  Syn  Aggr  Timeout  Activity
      et-0/0/48      Actor    No    No   Yes  Yes  Yes   Yes     Fast    Active
      et-0/0/48    Partner    No    No   Yes  Yes  Yes   Yes     Fast    Active
      et-1/0/48      Actor    No    No   Yes  Yes  Yes   Yes     Fast    Active
      et-1/0/48    Partner    No    No   Yes  Yes  Yes   Yes     Fast    Active
    LACP protocol:        Receive State  Transmit State          Mux State
      et-0/0/48                 Current   Fast periodic Collecting distributing
      et-1/0/48                 Current   Fast periodic Collecting distributing



...

Aggregated interface: ae11
    LACP state:       Role   Exp   Def  Dist  Col  Syn  Aggr  Timeout  Activity
```

```
        xe-0/0/10      Actor    No    No    Yes  Yes  Yes   Yes     Fast    Active
        xe-0/0/10    Partner    No    No    Yes  Yes  Yes   Yes     Fast    Active
        xe-1/0/10      Actor    No    No    Yes  Yes  Yes   Yes     Fast    Active
        xe-1/0/10    Partner    No    No    Yes  Yes  Yes   Yes     Fast    Active
      LACP protocol:        Receive State  Transmit State          Mux State
        xe-0/0/10                 Current   Fast periodic Collecting distributing
        xe-1/0/10                 Current   Fast periodic Collecting distributing
```

2. Verify that the EVPN routes are being learned through the overlay.

**NOTE**:

- Only selected excerpts of this output are displayed.
- The format of the EVPN routes is *EVPN-route-type:route-distinguisher:vni:mac-address*.

```
user@leaf-1> show route table bgp.evpn.0 evpn-ethernet-tag-id 10000

bgp.evpn.0: 828 destinations, 3169 routes (827 active, 0 holddown, 4 hidden)
+ = Active Route, - = Last Active, * = Both

## Spine 1: Virtual Gateway MAC Address for IPv4
2:192.168.0.1:1::10000::00:00:5e:00:01:01/304 MAC/IP
                   *[BGP/170] 09:12:00, localpref 100, from 192.168.0.1
                      AS path: I, validation-state: unverified
                    > to 172.16.1.2 via ae1.0
                    [BGP/170] 07:33:39, localpref 100, from 192.168.0.2
                      AS path: I, validation-state: unverified
                    > to 172.16.1.2 via ae1.0
                    [BGP/170] 07:31:15, localpref 100, from 192.168.0.3
                      AS path: I, validation-state: unverified
                    > to 172.16.1.2 via ae1.0
                    [BGP/170] 07:29:41, localpref 100, from 192.168.0.4
                      AS path: I, validation-state: unverified
                    > to 172.16.1.2 via ae1.0
## Spine 1: Virtual Gateway MAC Address for IPv6
2:192.168.0.1:1::10000::00:00:5e:00:02:01/304 MAC/IP
                   *[BGP/170] 09:12:00, localpref 100, from 192.168.0.1
                      AS path: I, validation-state: unverified
                    > to 172.16.1.2 via ae1.0
                    [BGP/170] 07:33:39, localpref 100, from 192.168.0.2
```

```
                      AS path: I, validation-state: unverified
                    > to 172.16.1.2 via ae1.0
                    [BGP/170] 07:31:15, localpref 100, from 192.168.0.3
                      AS path: I, validation-state: unverified
                    > to 172.16.1.2 via ae1.0
                    [BGP/170] 07:29:41, localpref 100, from 192.168.0.4
                      AS path: I, validation-state: unverified
                    > to 172.16.1.2 via ae1.0
## Spine 1: IRB MAC Address
2:192.168.0.1:1::10000::06:4b:8c:67:0f:f0/304 MAC/IP
                   *[BGP/170] 09:12:00, localpref 100, from 192.168.0.1
                      AS path: I, validation-state: unverified
                    > to 172.16.1.2 via ae1.0
                    [BGP/170] 07:33:39, localpref 100, from 192.168.0.2
                      AS path: I, validation-state: unverified
                    > to 172.16.1.2 via ae1.0
                    [BGP/170] 07:31:15, localpref 100, from 192.168.0.3
                      AS path: I, validation-state: unverified
                    > to 172.16.1.2 via ae1.0
                    [BGP/170] 07:29:41, localpref 100, from 192.168.0.4
                      AS path: I, validation-state: unverified
                    > to 172.16.1.2 via ae1.0
## Spine 1: ARP for the virtual gateway
2:192.168.0.1:1::10000::00:00:5e:00:01:01::10.1.0.254/304 MAC/IP
                   *[BGP/170] 09:12:00, localpref 100, from 192.168.0.1
                      AS path: I, validation-state: unverified
                    > to 172.16.1.2 via ae1.0
                    [BGP/170] 07:33:39, localpref 100, from 192.168.0.2
                      AS path: I, validation-state: unverified
                    > to 172.16.1.2 via ae1.0
                    [BGP/170] 07:31:15, localpref 100, from 192.168.0.3
                      AS path: I, validation-state: unverified
                    > to 172.16.1.2 via ae1.0
                    [BGP/170] 07:29:41, localpref 100, from 192.168.0.4
                      AS path: I, validation-state: unverified
                    > to 172.16.1.2 via ae1.0
## Spine 1: IRB IPv4 ARP
2:192.168.0.1:1::10000::06:4b:8c:67:0f:f0::10.1.0.1/304 MAC/IP
                   *[BGP/170] 00:04:50, localpref 100, from 192.168.0.1
                      AS path: I, validation-state: unverified
                    > to 172.16.1.2 via ae1.0
                    [BGP/170] 00:04:50, localpref 100, from 192.168.0.2
                      AS path: I, validation-state: unverified
                    > to 172.16.1.2 via ae1.0
                    [BGP/170] 00:04:50, localpref 100, from 192.168.0.3
                      AS path: I, validation-state: unverified
                    > to 172.16.1.2 via ae1.0
                    [BGP/170] 00:04:50, localpref 100, from 192.168.0.4
                      AS path: I, validation-state: unverified
                    > to 172.16.1.2 via ae1.0
## Spine 2: ARP for the virtual gateway
2:192.168.0.2:1::10000::00:00:5e:00:01:01::10.1.0.254/304 MAC/IP
                   *[BGP/170] 07:55:22, localpref 100, from 192.168.0.2
                      AS path: I, validation-state: unverified
                    > to 172.16.1.6 via ae2.0
                    [BGP/170] 07:33:39, localpref 100, from 192.168.0.1
                      AS path: I, validation-state: unverified
                    > to 172.16.1.6 via ae2.0
                    [BGP/170] 07:31:11, localpref 100, from 192.168.0.3
                      AS path: I, validation-state: unverified
                    > to 172.16.1.6 via ae2.0
                    [BGP/170] 07:29:37, localpref 100, from 192.168.0.4
                      AS path: I, validation-state: unverified
                    > to 172.16.1.6 via ae2.0
## Spine 2: IRB IPv4 ARP
2:192.168.0.2:1::10000::06:4b:8c:cd:13:f8::10.1.0.2/304 MAC/IP
                   *[BGP/170] 07:55:22, localpref 100, from 192.168.0.2
                      AS path: I, validation-state: unverified
                    > to 172.16.1.6 via ae2.0
                    [BGP/170] 07:33:39, localpref 100, from 192.168.0.1
                      AS path: I, validation-state: unverified
                    > to 172.16.1.6 via ae2.0
                    [BGP/170] 07:31:11, localpref 100, from 192.168.0.3
                      AS path: I, validation-state: unverified
                    > to 172.16.1.6 via ae2.0
                    [BGP/170] 07:29:37, localpref 100, from 192.168.0.4
                      AS path: I, validation-state: unverified
                    > to 172.16.1.6 via ae2.0
## Spine 1: IPv6 ARP for the virtual gateway
2:192.168.0.1:1::10000::00:00:5e:00:02:01::2001:db8::10:1:0:254/304 MAC/IP
                   *[BGP/170] 09:12:00, localpref 100, from 192.168.0.1
                      AS path: I, validation-state: unverified
                    > to 172.16.1.2 via ae1.0
                    [BGP/170] 07:33:39, localpref 100, from 192.168.0.2
                      AS path: I, validation-state: unverified
                    > to 172.16.1.2 via ae1.0
                    [BGP/170] 07:31:15, localpref 100, from 192.168.0.3
                      AS path: I, validation-state: unverified
                    > to 172.16.1.2 via ae1.0
                    [BGP/170] 07:29:41, localpref 100, from 192.168.0.4
                      AS path: I, validation-state: unverified
                    > to 172.16.1.2 via ae1.0
## Spine 1: IRB IPv6 ARP
2:192.168.0.1:1::10000::06:4b:8c:67:0f:f0::2001:db8::10:1:0:1/304 MAC/IP
                   *[BGP/170] 09:12:00, localpref 100, from 192.168.0.1
                      AS path: I, validation-state: unverified
                    > to 172.16.1.2 via ae1.0
                    [BGP/170] 07:33:39, localpref 100, from 192.168.0.2
                      AS path: I, validation-state: unverified
                    > to 172.16.1.2 via ae1.0
                    [BGP/170] 07:31:15, localpref 100, from 192.168.0.3
                      AS path: I, validation-state: unverified
                    > to 172.16.1.2 via ae1.0
                    [BGP/170] 07:29:41, localpref 100, from 192.168.0.4
                      AS path: I, validation-state: unverified
                    > to 172.16.1.2 via ae1.0

...
```

3. Verify on Leaf 1 and Leaf 3 that the Ethernet switching table has installed both the local MAC addresses and the remote MAC addresses learned through the overlay.

**NOTE**: To identify end systems learned remotely from the EVPN overlay, look for the MAC address, ESI logical interface, and ESI number. For example, Leaf 1 learns about an end system with the MAC address of `02:0c:10:03:02:02` through `esi.1885`. This end system has an ESI number of `00:00:00:00:00:00:51:10:00:01`. Consequently, this matches the ESI number configured for Leaf 4, 5, and 6 (QFX5110 switches), so we know that this end system is multihomed to these three leaf devices.

```
user@leaf-1> show ethernet-switching table vlan-id 300

MAC flags (S - static MAC, D - dynamic MAC, L - locally learned, P - Persistent static
           SE - statistics enabled, NM - non configured MAC, R - remote PE MAC, O - ovsdb
MAC)


Ethernet switching table : 10 entries, 10 learned
```

```
Routing instance : default-switch
   Vlan                MAC                 MAC      Logical                Active
   name                address             flags    interface              source
   VNI_30000           00:00:5e:00:01:01   DR       esi.1679
05:19:17:f3:41:00:00:75:30:00
   VNI_30000           00:00:5e:00:02:01   DR       esi.1679
05:19:17:f3:41:00:00:75:30:00
   VNI_30000           06:4b:8c:67:0f:f0   D        vtep.32770
192.168.0.1
   VNI_30000           06:4b:8c:cd:13:f8   D        vtep.32783
192.168.0.2
   VNI_30000           06:4b:8c:cd:c4:38   D        vtep.32769
192.168.0.3
   VNI_30000           06:38:e1:6f:30:29   D        vtep.32879
192.168.0.4
## Learned locally
   VNI_30000           02:0c:10:03:02:01   DL       ae11.0
## Learned from the QFX5110 switches - Leaf 4 to 6
   VNI_30000           02:0c:10:03:02:02   DR       esi.1885
00:00:00:00:00:00:51:10:00:01
## Learned from the QFX5200 switches - Leaf 7 to 9
   VNI_30000           02:0c:10:03:02:03   DR       esi.1887
00:00:00:00:00:00:52:00:00:01
## Learned from the QFX10002 switches - Leaf 10 to 12
   VNI_30000           02:0c:10:03:02:04   DR       esi.1892
00:00:00:00:00:01:00:00:00:01
## IPv4 virtual gateway MAC address learned over the overlay
and distributed to the leaf devices by Spine 1, 2, 3 and 4
00:00:5e:00:01:01
# IPv6 virtual gateway MAC address learned over Overlay
00:00:5e:00:02:01
## IRB MAC address prefix for Spine 1, 2, and 3 (Physical
MAC address)
06:4b:*
## End System MAC address, connected locally to the leaf
device
02:0c:10:03:02:01
## MAC address learned over the overlay, these end systems
are also multihomed
02:0c:10:03:02:02,03,04
user@leaf-3> show ethernet-switching table vlan-id 100

MAC flags (S - static MAC, D - dynamic MAC, L - locally learned, P - Persistent static
          SE - statistics enabled, NM - non configured MAC, R - remote PE MAC, O - ovsdb
MAC)



Ethernet switching table : 106 entries, 106 learned
Routing instance : default-switch
   Vlan                MAC                 MAC      Logical                Active
   name                address             flags    interface              source
## 00:00:5e:00:01:01 is the virtual gateway MAC address
for the spine devices and is reachable over the dynamically created
logical link esi.1679. As a result, you can use this ESI number to
filter future command output by using esi.1679 to find the virtual
gateway.
   VNI_10000           00:00:5e:00:01:01   DR       esi.1769
05:19:17:f3:41:00:00:27:10:00
   VNI_10000           00:00:5e:00:02:01   DR       esi.1769
05:19:17:f3:41:00:00:27:10:00
   VNI_10000           06:4b:8c:67:0f:f0   D        vtep.32781
192.168.0.1
   VNI_10000           06:4b:8c:cd:13:f8   D        vtep.32782
192.168.0.2
   VNI_10000           06:4b:8c:cd:c4:38   D        vtep.32775
192.168.0.3
## Learned locally
   VNI_10000           02:0c:10:01:02:01   DL       ae11.0
## Learned through the overlay
   VNI_10000           02:0c:10:01:02:02   DR       esi.1760
00:00:00:00:00:00:51:10:00:01
   VNI_10000           02:0c:10:01:02:03   DR       esi.1782
00:00:00:00:00:00:52:00:00:01
   VNI_10000           02:0c:10:01:02:04   DR       esi.1758
00:00:00:00:00:01:00:00:00:01
   VNI_10000           06:38:e1:6f:30:29   D        vtep.32783
192.168.0.4
   VNI_10000           0e:ad:10:01:00:01   D        vtep.32821             192.168.1.85
```

4. Verify on Leaf 1 that the virtual gateway ESI (esi.1679) is reachable by all the spine devices.

```
user@leaf-1> show ethernet-switching vxlan-tunnel-end-point esi | find esi.1679
ESI                            RTT                       VLNBH INH     ESI-IFL   LOC-IFL
#RVTEPs
05:19:17:f3:41:00:00:75:30:00 default-switch           1679  131072  esi.1679
```

```
4
    RVTEP-IP             RVTEP-IFL      VENH     MASK-ID   FLAGS
    192.168.0.4          vtep.32879     1890     3         2
    192.168.0.2          vtep.32783     1795     2         2
    192.168.0.1          vtep.32770     1682     1         2
    192.168.0.3          vtep.32769     1764     0         2
```

5. Verify the remote EVPN routes coming from VNI 10000 and MAC address 02:0c:10:01:02:02. In this case, they are coming from Leaf 4 (192.168.1.4) by way of Spine 1 (192.168.0.1).

**NOTE**: The format of the EVPN routes is *EVPN-route-type:route-distinguisher:vni:mac-address*.

```
user@leaf-1> show route table bgp.evpn.0 evpn-ethernet-tag-id 10000 evpn-mac-address
02:0c:10:01:02:02

bgp.evpn.0: 910 destinations, 3497 routes (904 active, 0 holddown, 24 hidden)
+ = Active Route, - = Last Active, * = Both

2:192.168.1.4:1::10000::02:0c:10:01:02:02/304 MAC/IP
                   *[BGP/170] 00:11:37, localpref 100, from 192.168.0.1
                      AS path: I, validation-state: unverified
                    > to 172.16.1.10 via ae3.0
                    [BGP/170] 00:11:37, localpref 100, from 192.168.0.2
                      AS path: I, validation-state: unverified
                    > to 172.16.1.10 via ae3.0
                    [BGP/170] 00:11:37, localpref 100, from 192.168.0.3
                      AS path: I, validation-state: unverified
                    > to 172.16.1.10 via ae3.0
                    [BGP/170] 00:11:37, localpref 100, from 192.168.0.4
                      AS path: I, validation-state: unverified
                    > to 172.16.1.10 via ae3.0
user@leaf-1> show route table bgp.evpn.0 evpn-ethernet-tag-id 10000 evpn-mac-address
02:0c:10:01:02:02 detail

bgp.evpn.0: 925 destinations, 3557 routes (919 active, 0 holddown, 24 hidden)
2:192.168.1.4:1::10000::02:0c:10:01:02:02/304 MAC/IP (4 entries, 0 announced)
        *BGP    Preference: 170/-101
                Route Distinguisher: 192.168.1.4:1
                Next hop type: Indirect, Next hop index: 0
                Address: 0xb3a2170
```

```
                  Next-hop reference count: 160
                  Source: 192.168.0.1
                  Protocol next hop: 192.168.1.4
                  Indirect next hop: 0x2 no-forward INH Session ID: 0x0
                  State: <Active Int Ext>
                  Local AS: 4210000001 Peer AS: 4210000001
                  Age: 13:42      Metric2: 0
                  Validation State: unverified
                  Task: BGP_4210000001.192.168.0.1
                  AS path: I (Originator)
                  Cluster list:  192.168.0.10
                  Originator ID: 192.168.1.4
                  Communities: target:62273:268445456 encapsulation:vxlan(0x8)
                  Import Accepted
                  Route Label: 10000
                  ESI: 00:00:00:00:00:00:51:10:00:01
                  Localpref: 100
                  Router ID: 192.168.0.1
                  Secondary Tables: default-switch.evpn.0
...
## This output has been abbreviated. In a full set of output,
there should also be routes sourced by Spine 2 (192.168.0.2), Spine
3 (192.168.0.3), and Spine 4 (192.168.0.4).
```

6. Verify the source and destination address of each VTEP interface and view their status.

**NOTE**: There are 96 leaf devices and four spine devices, so there are 100 VTEP interfaces in this reference design - one VTEP interface per device.

```
user@leaf-1> show ethernet-switching vxlan-tunnel-end-point source
Logical System Name       Id  SVTEP-IP         IFL   L3-Idx
<default>                 0   192.168.1.1      lo0.0    0
    L2-RTT                   Bridge Domain              VNID     MC-Group-IP
    default-switch           VNI_10000+100              10000    0.0.0.0
    default-switch           VNI_20000+200              20000    0.0.0.0
    default-switch           VNI_30000+300              30000    0.0.0.0
    default-switch           VNI_40000+400              40000    0.0.0.0
user@leaf-1> show interfaces terse vtep
Interface                Admin Link Proto    Local                 Remote
vtep                     up    up
vtep.32768               up    up
vtep.32769              up    up   eth-switch
vtep.32770              up    up   eth-switch
vtep.32771              up    up   eth-switch
vtep.32772              up    up   eth-switch
...
vtep.32869              up    up   eth-switch
user@leaf-1> show interfaces vtep
Physical interface: vtep, Enabled, Physical link is Up
  Interface index: 646, SNMP ifIndex: 503
  Type: Software-Pseudo, Link-level type: VxLAN-Tunnel-Endpoint, MTU: Unlimited, Speed: 
Unlimited
  Device flags   : Present Running
  Link type      : Full-Duplex
  Link flags     : None
  Last flapped   : Never
    Input packets : 0
    Output packets: 0

  Logical interface vtep.32768 (Index 554) (SNMP ifIndex 648)
    Flags: Up SNMP-Traps 0x4000 Encapsulation: ENET2
    VXLAN Endpoint Type: Source, VXLAN Endpoint Address: 192.168.1.1, L2 Routing Instance: 
default-switch, L3 Routing Instance: default
    Input packets : 0
    Output packets: 0



...  Logical interface vtep.32814 (Index 613) (SNMP ifIndex 903)
    Flags: Up SNMP-Traps Encapsulation: ENET2
    VXLAN Endpoint Type: Remote, VXLAN Endpoint Address: 192.168.1.96, L2 Routing Instance: 
default-switch, L3 Routing Instance: default
    Input packets : 0
    Output packets: 6364
    Protocol eth-switch, MTU: Unlimited
      Flags: Trunk-Mode
```

7. Verify that each VNI maps to the associated VXLAN tunnel.

```
user@leaf-1> show ethernet-switching vxlan-tunnel-end-point remote
                 0   192.168.1.1      lo0.0    0
 RVTEP-IP         IFL-Idx   NH-Id
 192.168.0.1      587       1792
    VNID          MC-Group-IP
    10000         0.0.0.0
    20000         0.0.0.0
    30000         0.0.0.0
    40000         0.0.0.0

...

RVTEP-IP         IFL-Idx   NH-Id
192.168.1.96     613       1820
    VNID          MC-Group-IP
    10000         0.0.0.0
    20000         0.0.0.0
    30000         0.0.0.0
    40000         0.0.0.0
```

8. Verify that MAC addresses are learned through the VXLAN tunnels.

```
user@leaf-1> show ethernet-switching 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, P -Pinned MAC)

Logical system   : <default>
Routing instance : default-switch
 Bridging domain : VNI_10000+100, VLAN : 100, VNID : 10000
   MAC                  MAC      Logical          Remote VTEP
   address              flags    interface        IP address
   02:0c:10:01:02:04    DR       esi.1764         192.168.1.11 192.168.1.12 192.168.1.10
   02:0c:10:01:02:02    DR       esi.1771         192.168.1.6 192.168.1.4
   02:0c:10:01:02:03    DR       esi.1774         192.168.1.7
   00:00:5e:00:01:01    DR       esi.1781         192.168.0.4 192.168.0.2 192.168.0.1
192.168.0.3
   06:4b:8c:cd:c4:38    D        vtep.32779       192.168.0.3
   06:4b:8c:67:0f:f0    D        vtep.32781       192.168.0.1
   06:4b:8c:cd:13:f8    D        vtep.32782       192.168.0.2
   06:38:e1:6f:30:29    D        vtep.32783       192.168.0.4
---(more)---
```

9. Verify multihoming information of the gateway and the aggregated Ethernet interfaces.

```
user@leaf-1> show ethernet-switching vxlan-tunnel-end-point esi
## Local AE link – QFX5100 leaf devices
```

```
ESI                           RTT                     VLNBH INH     ESI-IFL   LOC-IFL
#RVTEPs
00:00:00:00:00:00:51:00:00:01 default-switch          1768  131078  esi.1768  ae11.0,
2
    RVTEP-IP             RVTEP-IFL      VENH     MASK-ID   FLAGS
    192.168.1.2          vtep.32780     1782     1         2
    192.168.1.3          vtep.32772     1767     0         2
## Remote AE Link for QFX5110 leaf devices
ESI                           RTT                     VLNBH INH     ESI-IFL   LOC-IFL
#RVTEPs
00:00:00:00:00:00:51:10:00:01 default-switch          1771  131081  esi.1771
3
    RVTEP-IP             RVTEP-IFL      VENH     MASK-ID   FLAGS
    192.168.1.6          vtep.32771     1766     2         2
    192.168.1.4          vtep.32770     1765     1         2
    192.168.1.5          vtep.32774     1770     0         2
## Remote AE Link for QFX5200 leaf devices
ESI                           RTT                     VLNBH INH     ESI-IFL   LOC-IFL
#RVTEPs
00:00:00:00:00:00:52:00:00:01 default-switch          1774  131084  esi.1774
3
    RVTEP-IP             RVTEP-IFL      VENH     MASK-ID   FLAGS
    192.168.1.9          vtep.32778     1776     2         2
    192.168.1.8          vtep.32777     1775     1         2
    192.168.1.7          vtep.32776     1773     0         2
## Remote AE Link for QFX10002 leaf devices
ESI                           RTT                     VLNBH INH     ESI-IFL   LOC-IFL
#RVTEPs
00:00:00:00:00:01:00:00:00:01 default-switch          1764  131074  esi.1764
3
    RVTEP-IP             RVTEP-IFL      VENH     MASK-ID   FLAGS
    192.168.1.11         vtep.32775     1772     2         2
    192.168.1.12         vtep.32773     1769     1         2
    192.168.1.10         vtep.32769     1759     0         2
## ESI multihoming to the VTEP for each segment
ESI                           RTT                     VLNBH INH     ESI-IFL   LOC-IFL
#RVTEPs
05:19:17:f3:41:00:00:27:10:00 default-switch          1781  131091  esi.1781
4
    RVTEP-IP             RVTEP-IFL      VENH     MASK-ID   FLAGS
    192.168.0.4          vtep.32783     1796     3         2
    192.168.0.2          vtep.32782     1793     2         2
    192.168.0.1          vtep.32781     1792     1         2
    192.168.0.3          vtep.32779     1777     0         2

...
```

10. Verify that the VXLAN tunnel from one leaf to another leaf is load balanced with equal cost multipathing (ECMP) over the underlay.

```
user@leaf-1> show route forwarding-table table default-switch extensive | find vtep.32770

Destination:  vtep.32770
  Route type: interface
  Route reference: 0                   Route interface-index: 576
  Multicast RPF nh index: 0
  P2mpidx: 0
  Flags: sent to PFE
  Nexthop:
  Next-hop type: composite             Index: 1765     Reference: 12
  Next-hop type: indirect              Index: 131076   Reference: 3
  Next-hop type: unilist               Index: 131193   Reference: 238
  Nexthop: 172.16.1.2
  Next-hop type: unicast               Index: 1791     Reference: 10
  Next-hop interface: ae1.0         Weight: 0x0
  Nexthop: 172.16.1.6
  Next-hop type: unicast               Index: 1794     Reference: 10
  Next-hop interface: ae2.0         Weight: 0x0
  Nexthop: 172.16.1.10
  Next-hop type: unicast               Index: 1758     Reference: 10
  Next-hop interface: ae3.0         Weight: 0x0
  Nexthop: 172.16.1.14
  Next-hop type: unicast               Index: 1795     Reference: 10
  Next-hop interface: ae4.0         Weight: 0x0
```

11. Verify that remote MAC addresses are reachable through ECMP.

```
user@leaf-1> show route forwarding-table table default-switch extensive destination
02:0c:10:01:02:03/48
Routing table: default-switch.evpn-vxlan [Index 4]
Bridging domain: VNI_10000.evpn-vxlan [Index 3]
VPLS:
Enabled protocols: Bridging, ACKed by all peers, EVPN VXLAN,

Destination:  02:0c:10:01:02:03/48
```

```
    Learn VLAN: 0                           Route type: user
    Route reference: 0                      Route interface-index: 582
    Multicast RPF nh index: 0
    P2mpidx: 0
    IFL generation: 169                     Epoch: 0
    Sequence Number: 0                      Learn Mask: 0x400000000000000003000000000000000000000
    L2 Flags: control_dyn
    Flags: sent to PFE
    Nexthop:
    Next-hop type: composite                Index: 1773     Reference: 12
    Next-hop type: indirect                 Index: 131085   Reference: 3
    Next-hop type: unilist                  Index: 131193   Reference: 238
    Nexthop: 172.16.1.2
    Next-hop type: unicast                  Index: 1791     Reference: 10
    Next-hop interface: ae1.0               Weight: 0x0
    Nexthop: 172.16.1.6
    Next-hop type: unicast                  Index: 1794     Reference: 10
    Next-hop interface: ae2.0               Weight: 0x0
    Nexthop: 172.16.1.10
    Next-hop type: unicast                  Index: 1758     Reference: 10
    Next-hop interface: ae3.0               Weight: 0x0
    Nexthop: 172.16.1.14
    Next-hop type: unicast                  Index: 1795     Reference: 10
    Next-hop interface: ae4.0               Weight: 0x0
```

**NOTE**: Though the MAC address is reachable over multiple VTEP interfaces, QFX5100, QFX5110, QFX5120-32C, and QFX5200 switches do not support ECMP across the overlay because of a merchant ASIC limitation. Only the QFX10000 line of switches contain a custom Juniper Networks ASIC that supports ECMP across both the overlay and the underlay.

```
user@leaf-1> show ethernet-switching table vlan-id 100 | match 02:0c:10:01:02:03
   VNI_10000           02:0c:10:01:02:03   DR       esi.1774
00:00:00:00:00:00:52:00:00:01
user@leaf-1> show route forwarding-table table default-switch extensive destination
02:0c:10:01:02:03/48
Routing table: default-switch.evpn-vxlan [Index 9]
Bridging domain: VNI_10000.evpn-vxlan [Index 3]
VPLS:
Enabled protocols: Bridging, ACKed by all peers, EVPN VXLAN,
Destination:  02:0c:10:01:02:03/48
  Learn VLAN: 0                         Route type: user
  Route reference: 0                    Route interface-index: 550
  Multicast RPF nh index: 0
  P2mpidx: 0
  IFL generation: 0                     Epoch: 0
  Sequence Number: 0                    Learn Mask: 0x400000000000000001000000000000000000000
  L2 Flags: control_dyn, esi
  Flags: sent to PFE
  Next-hop type: indirect               Index: 2097173  Reference: 5
  Nexthop:
  Next-hop type: composite              Index: 1947     Reference: 2
  Nexthop:
  Next-hop type: composite              Index: 1948     Reference: 8
  Next-hop type: indirect               Index: 2097174  Reference: 3
  Next-hop type: unilist                Index: 2097280  Reference: 241
  Nexthop: 172.16.10.2
  Next-hop type: unicast                Index: 1950     Reference: 11
  Next-hop interface: ae1.0             Weight: 0x0
  Nexthop: 172.16.10.6
  Next-hop type: unicast                Index: 1956     Reference: 10
  Next-hop interface: ae2.0             Weight: 0x0
  Nexthop: 172.16.10.10
  Next-hop type: unicast                Index: 1861     Reference: 10
  Next-hop interface: ae3.0             Weight: 0x0
  Nexthop: 172.16.10.14
  Next-hop type: unicast                Index: 1960     Reference: 10
  Next-hop interface: ae4.0             Weight: 0x0
```

12. Verify which device is the Designated Forwarder (DF) for broadcast, unknown, and multicast (BUM) traffic coming from the VTEP tunnel.

**NOTE**: Because the DF IP address is listed as 192.168.1.2, Leaf 2 is the DF.

```
user@leaf-1> show evpn instance esi 00:00:00:00:00:00:51:00:00:01 designated-forwarder
Instance: default-switch
  Number of ethernet segments: 12
    ESI: 00:00:00:00:00:00:51:00:00:01
      Designated forwarder: 192.168.1.2
```

SEE ALSO

*Overview of VLAN Services for EVPN*

*EVPN Overview*

*Understanding VXLANs*

*Understanding EVPN with VXLAN Data Plane Encapsulation*

*Configuring EVPN Routing Instances*

*Configuring Aggregated Ethernet LACP (CLI Procedure)*

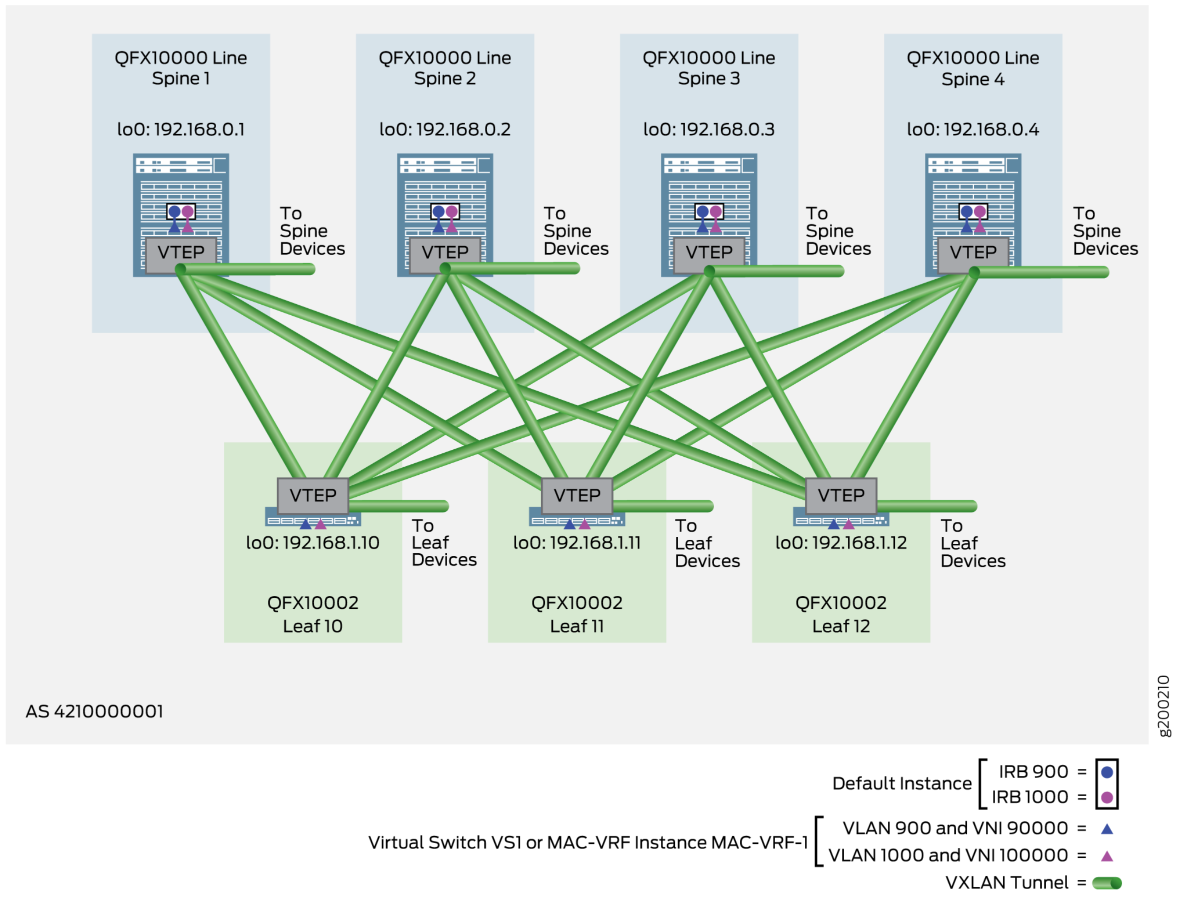
## Configuring a VLAN-Aware CRB Overlay with Virtual Switches or MAC-VRF Instances

IN THIS SECTION

You can configure a VLAN-aware CRB overlay model using virtual switches or MAC-VRF instances. With either of these models, you can configure multiple switching instances where each switching instance can support up to 4094 VLANs per instance.

The configuration method for VLANs (at the leaf devices) and IRB interfaces (at the spine devices) is similar to the default instance method for VLAN-aware CRB overlays. The main difference is that you configure certain elements inside the virtual switching instances or MAC-VRF instances. See Figure 47 on page 176.

**Figure 47: VLAN-Aware CRB Overlay — Virtual Switch Instance or MAC-VRF Instance**



When you implement this style of overlay on a spine device, you:

- Configure a virtual switch or MAC-VRF instance with:
  - The loopback interface as the VTEP source interface.
  - Route distinguishers and route targets.
  - EVPN with VXLAN encapsulation.
  - VLAN to VNI mappings and Layer 3 IRB interface associations.
- Configure virtual gateways, virtual MAC addresses, and corresponding IRB interfaces (to provide routing between VLANs).

To implement this overlay style on a leaf device:

- Configure a virtual switch or a MAC-VRF instance with:
  - The loopback interface as the VTEP source interface.

  - Route distinguishers and route targets.
  - EVPN with VXLAN encapsulation.
  - VLAN to VNI mappings.
- Set the following end system-facing elements:
  - An Ethernet segment ID (ESI).
  - Flexible VLAN tagging and extended VLAN bridge encapsulation.
  - LACP settings.
  - VLAN IDs.

For an overview of VLAN-aware CRB overlays, see the Centrally-Routed Bridging Overlay section in "Data Center Fabric Blueprint Architecture Components" on page 9.

For information on MAC-VRF instances, see "MAC-VRF Instances for Multitenancy in Network Virtualization Overlays" on page 24 and MAC-VRF Routing Instance Type Overview.

**NOTE**:

- For a simpler method that works on all leaf platforms used in this reference design, see "Configuring a VLAN-Aware CRB Overlay in the Default Instance" on page 143

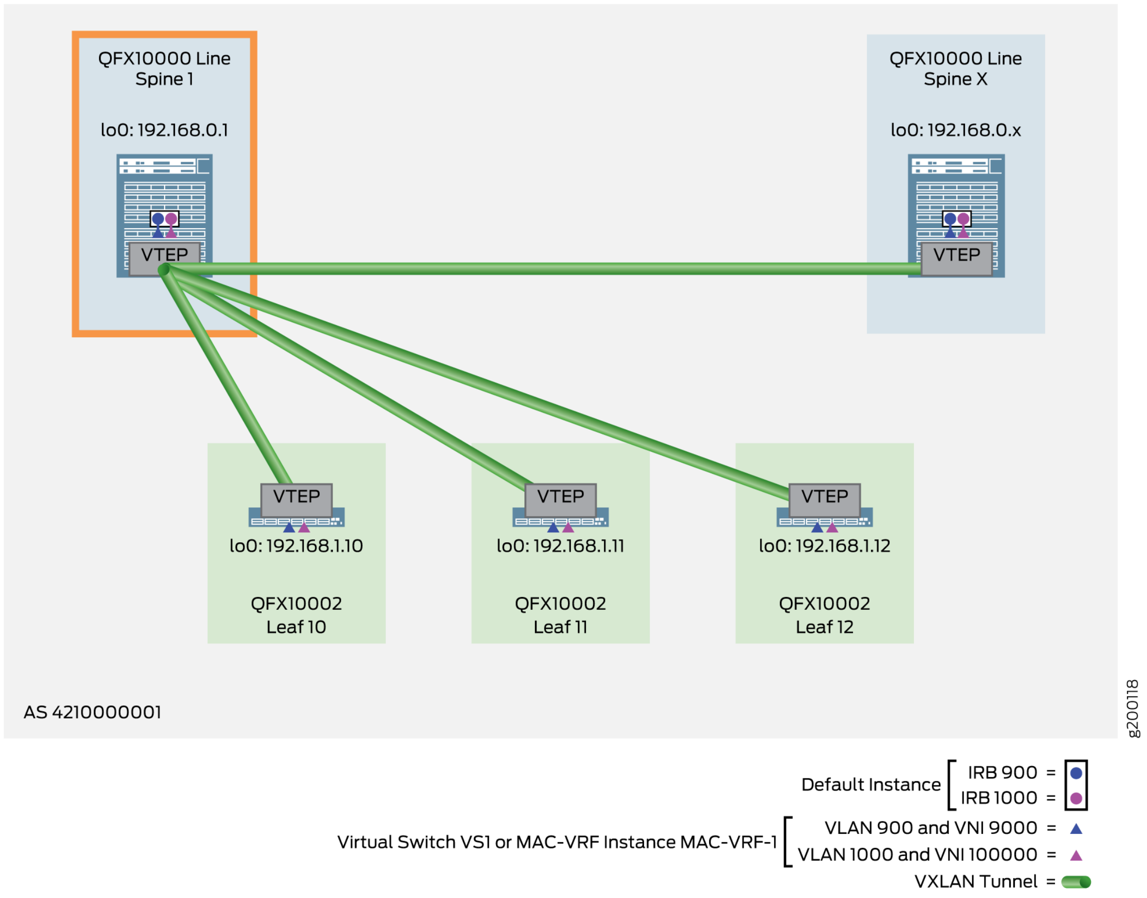
The following sections provide the detailed steps of how to configure and verify the VLAN-aware CRB overlay with virtual switches or MAC-VRF instances.

## Configuring the VLAN-Aware CRB Overlay with Virtual Switches or MAC-VRF Instances on a Spine Device

To configure a VLAN-aware style of CRB overlay on a spine device, perform the following:

**NOTE**: The following example shows the configuration for Spine 1, as shown in Figure 48 on page 178.

**Figure 48: VLAN-Aware CRB Overlay with Virtual Switches or a MAC-VRF Instance – Spine Device**



1. Ensure the IP fabric underlay is in place. To configure an IP fabric on spine devices, see "IP Fabric Underlay Network Design and Implementation" on page 80.
2. Confirm that your IBGP overlay is up and running. To configure an IBGP overlay on your spine devices, see "Configure IBGP for the Overlay" on page 96.
3. (QFX5130 and QFX5700 switches only) On any QFX5130 or QFX5700 switches in the fabric that you configure with EVPN-VXLAN, set the `host-profile` unified forwarding profile option to support EVPN with VXLAN encapsulation (see *Layer 2 Forwarding Tables* for details):

```
set system packet-forwarding-options forwarding-profile host-profile
```

4. Configure a virtual switch instance (VS1) or a MAC-VRF instance (MAC-VRF-1) for a VLAN-aware service. With the VLAN-aware service type, you can configure the instance with one or more VLANs. Include VTEP information, VXLAN encapsulation, VLAN to VNI mapping, associated IRB interfaces, and other instance details (such as a route distinguisher and a route target) as part of the configuration.

For a virtual switch instance, use `instance-type virtual-switch`. Using the VLAN-aware model, configure VLANs VNI_90000 and VNI_100000 in the virtual switch instance with the associated IRB interfaces.

*Spine 1 (Virtual Switch Instance)*:

```
set routing-instances VS1 vtep-source-interface lo0.0
set routing-instances VS1 instance-type virtual-switch
set routing-instances VS1 route-distinguisher 192.168.0.1:900
set routing-instances VS1 vrf-target target:62273:90000
set routing-instances VS1 vrf-target auto
set routing-instances VS1 protocols evpn encapsulation vxlan
set routing-instances VS1 protocols evpn extended-vni-list all
set routing-instances VS1 protocols evpn default-gateway no-gateway-community
set routing-instances VS1 vlans VNI_90000 vlan-id none
set routing-instances VS1 vlans VNI_90000 l3-interface irb.900
set routing-instances VS1 vlans VNI_90000 vxlan vni 90000
set routing-instances VS1 vlans VNI_100000 vlan-id none
set routing-instances VS1 vlans VNI_100000 l3-interface irb.1000
set routing-instances VS1 vlans VNI_100000 vxlan vni 100000
```

With MAC-VRF instances, use `instance-type mac-vrf`. You also configure the service type when you create the MAC-VRF instance. Here we configure `service-type vlan-aware` with the two VLANs VNI_90000 and VNI_100000 and their associated IRB interfaces in the MAC-VRF instance.

*Spine 1 (MAC-VRF Instance)*:

```
set routing-instances MAC-VRF-1 vtep-source-interface lo0.0
set routing-instances MAC-VRF-1 instance-type mac-vrf
set routing-instances MAC-VRF-1 service-type vlan-aware
set routing-instances MAC-VRF-1 route-distinguisher 192.168.0.1:900
set routing-instances MAC-VRF-1 vrf-target target:62273:90000
set routing-instances MAC-VRF-1 vrf-target auto
set routing-instances MAC-VRF-1 protocols evpn encapsulation vxlan
set routing-instances MAC-VRF-1 protocols evpn extended-vni-list all
set routing-instances MAC-VRF-1 protocols evpn default-gateway no-gateway-community
set routing-instances MAC-VRF-1 vlans VNI_90000 vlan-id none
set routing-instances MAC-VRF-1 vlans VNI_90000 l3-interface irb.900
set routing-instances MAC-VRF-1 vlans VNI_90000 vxlan vni 90000
set routing-instances MAC-VRF-1 vlans VNI_100000 vlan-id none
set routing-instances MAC-VRF-1 vlans VNI_100000 l3-interface irb.1000
set routing-instances MAC-VRF-1 vlans VNI_100000 vxlan vni 100000
```

5. (MAC-VRF instances only) Enable shared tunnels on the device.

   A device can have problems with VTEP scaling when the configuration uses multiple MAC-VRF instances. As a result, to avoid this problem, we require that you enable the shared tunnels feature on the QFX5000 line of switches with a MAC-VRF instance configuration. When you configure the shared-tunnels option, the device minimizes the number of next-hop entries to reach remote VTEPs. The following statement globally enables shared VXLAN tunnels on the device:

   ```
   set forwarding-options evpn-vxlan shared-tunnels
   ```

   This statement is optional on the QFX10000 line of switches, which can handle higher VTEP scaling than QFX5000 switches.

   **NOTE**: This setting requires you to reboot the device.

6. Configure spine devices with one or more VLANs for the VLAN-aware method. Include settings for the IPv4 and IPv6 virtual gateways and virtual MAC addresses. This example shows the configuration for Spine 1 with IRB interfaces and virtual gateways for VLANs VNI_90000 and VNI_100000.

   *Spine 1*:

   ```
   set interfaces irb unit 900 proxy-macip-advertisement
   set interfaces irb unit 900 virtual-gateway-accept-data
   set interfaces irb unit 900 family inet address 10.1.8.1/24 virtual-gateway-address 10.1.8.254
   set interfaces irb unit 900 family inet6 address 2001:db8::10:1:8:1/112 virtual-gateway-
   address 2001:db8::10:1:8:254
   set interfaces irb unit 900 family inet6 address fe80::10:1:8:1/112
   set interfaces irb unit 900 family inet6 nd6-stale-time 600
   set interfaces irb unit 900 virtual-gateway-v4-mac 00:00:5e:90:00:00
   set interfaces irb unit 900 virtual-gateway-v6-mac 00:00:5e:90:00:00
   set interfaces irb unit 1000 proxy-macip-advertisement
   set interfaces irb unit 1000 virtual-gateway-accept-data
   set interfaces irb unit 1000 family inet address 10.1.9.1/24 virtual-gateway-address
   10.1.9.254
   set interfaces irb unit 1000 family inet6 address 2001:db8::10:1:9:1/112 virtual-gateway-
   address 2001:db8::10:1:9:254
   set interfaces irb unit 1000 family inet6 address fe80::10:1:9:1/112
   set interfaces irb unit 1000 family inet6 nd6-stale-time 600
   ```

```
set interfaces irb unit 1000 virtual-gateway-v4-mac 00:00:5e:a0:00:00
set interfaces irb unit 1000 virtual-gateway-v6-mac 00:00:5e:a0:00:00
```

## Verifying the VLAN-Aware Model for a CRB Overlay with Virtual Switches or MAC-VRF Instances on a Spine Device

To verify this style of overlay on a spine device, run the commands in this section.

Most commands here show output for a virtual switch instance configuration. With a MAC-VRF instance configuration, you can alternatively use:

- `show mac-vrf forwarding` commands that are aliases for the `show ethernet-switching` commands in this section.
- The `show mac-vrf routing database` command, which is an alias for the `show evpn database` command in this section.
- The `show mac-vrf routing instance` command, which is an alias for the `show evpn instance` command in this section.

See MAC-VRF Routing Instance Type Overview for tables of `show mac-vrf forwarding` and `show ethernet-switching` command mappings, and `show mac-vrf routing` command aliases for `show evpn` commands.

Otherwise, you can use the commands in this section for either virtual switch instances or MAC-VRF instances.

The output with a MAC-VRF instance configuration displays similar information for MAC-VRF routing instances as this section shows for virtual switch instances. One main difference you might see is in the output with MAC-VRF instances on devices where you enable the shared tunnels feature. With shared tunnels enabled, you see VTEP interfaces in the following format:

```
vtep-index.shared-tunnel-unit
```

where:

- *index* is the index associated with the MAC-VRF routing instance.
- *shared-tunnel-unit* is the unit number associated with the shared tunnel remote VTEP logical interface.

For example, if a device has a MAC-VRF instance with index 26 and the instance connects to two remote VTEPs, the shared tunnel VTEP logical interfaces might look like this:

```
vtep-26.32823
vtep-26.32824
```

1. Verify the IRB interfaces for VNIs 90000 and 100000 are operational for both IPv4 and IPv6.

```
user@spine-1> show interfaces terse irb | find irb\.900
irb.900                 up    up   inet     10.1.8.1/24
                                            10.1.8.254/24
                                   inet6    2001:db8::10:1:8:1/112
                                            2001:db8::10:1:8:254/112
                                            fe80::10:1:8:1/112
irb.1000                up    up   inet     10.1.9.1/24
                                            10.1.9.254/24
                                   inet6    2001:db8::10:1:9:1/112
                                            2001:db8::10:1:9:254/112
                                            fe80::10:1:9:1/112
```

2. (MAC-VRF instances only) Verify the VLANs you configured as part of the MAC-VRF instance.

```
user@spine-1> show mac-vrf forwarding instance MAC-VRF-1
Information for routing instance and VLAN:

Flags (DL - disable learning, SE - stats enabled,
       AD - packet action drop, LH - MAC limit hit,
       MI - mac+ip limit hit)

Inst Logical   Routing            VLAN name          Index IRB   Flags Tag
type system    instance                                    index
RTT  Default   MAC-VRF-1                             26
vlan Default   MAC-VRF-1          VNI-90000          3403  4204        900
vlan Default   MAC-VRF-1          VNI-100000         3425  4203        1000
.
.
.

user@spine-1> show vlans VNI-90000
Routing instance        VLAN name             Tag          Interfaces
MAC-VRF-1               VNI-90000             NA
```

```
                                                              esi.105902*
                                                              esi.89032*
                                                              vtep-26.32823*
                                                              vtep-26.32824*
                                                              vtep-26.32827*
```

3. Verify switching details about the EVPN routing instance. This output includes information about the route distinguisher (192.168.1.10:900), VXLAN encapsulation, ESI (00:00:00:00:00:01:00:00:00:02), verification of the VXLAN tunnels for VLANs 900 and 1000, EVPN neighbors (Spine 2 - 4, and Leaf 10 - 12), and the source VTEP IP address (192.168.0.1).

```
user@spine-1> show evpn instance VS1 extensive
Instance: VS1
  Route Distinguisher: 192.168.0.1:900
  Encapsulation type: VXLAN
  MAC database status                     Local  Remote
    MAC advertisements:                       2      14
    MAC+IP advertisements:                   14      26
    Default gateway MAC advertisements:       4       0
  Number of local interfaces: 0 (0 up)
  Number of IRB interfaces: 2 (2 up)
    Interface name  VLAN   VNI    Status  L3 context
    irb.1000               100000  Up     master
    irb.900                90000   Up     master
  Number of bridge domains: 2
    VLAN  Domain ID   Intfs / up    IRB intf   Mode             MAC sync  IM route label  SG
sync  IM core nexthop
    8191  90000          0    0     irb.900    Extended         Enabled   90000
Disabled
    8191  100000         0    0     irb.1000   Extended         Enabled   100000
Disabled
  Number of neighbors: 6
    Address               MAC    MAC+IP        AD        IM        ES Leaf-label
    192.168.0.2             4        10         2         2         0
    192.168.0.3             4        10         2         2         0
    192.168.0.4             4        10         2         2         0
    192.168.1.10            1         2         2         2         0
    192.168.1.11            0         0         2         2         0
    192.168.1.12            1         2         2         2         0
  Number of ethernet segments: 3
    ESI: 00:00:00:00:00:01:00:00:00:02
      Status: Resolved
```

```
        Number of remote PEs connected: 3
          Remote PE        MAC label  Aliasing label  Mode
          192.168.1.12     90000      0               all-active
          192.168.1.11     90000      0               all-active
          192.168.1.10     100000     0               all-active
      ESI: 05:19:17:f3:41:00:01:5f:90:00
        Local interface: irb.900, Status: Up/Forwarding
        Number of remote PEs connected: 3
          Remote PE        MAC label  Aliasing label  Mode
          192.168.0.3      90000      0               all-active
          192.168.0.2      90000      0               all-active
          192.168.0.4      90000      0               all-active
      ESI: 05:19:17:f3:41:00:01:86:a0:00
        Local interface: irb.1000, Status: Up/Forwarding
        Number of remote PEs connected: 3
          Remote PE        MAC label  Aliasing label  Mode
          192.168.0.3      100000     0               all-active
          192.168.0.2      100000     0               all-active
          192.168.0.4      100000     0               all-active
    Router-ID: 192.168.0.1
    Source VTEP interface IP: 192.168.0.1
```

4. Verify the MAC address table on the leaf device.

**NOTE**:

- 00:00:5e:90:00:00 and 00:00:5e:a0:00:00 are the IP subnet gateways on the spine device.
- 02:0c:10:09:02:01 and 02:0c:10:08:02:01 are end systems connected through the leaf device.

```
user@spine-1> show ethernet-switching table instance VS1

MAC flags (S - static MAC, D - dynamic MAC, L - locally learned, P - Persistent static
           SE - statistics enabled, NM - non configured MAC, R - remote PE MAC, O - ovsdb MAC)


Ethernet switching table : 5 entries, 5 learned
Routing instance : VS1
   Vlan                MAC                 MAC      Logical               Active
   name                address             flags    interface             source
   VNI_100000         00:00:5e:a0:00:00   DR       esi.2454
05:19:17:f3:41:00:01:86:a0:00
   VNI_100000         06:4b:8c:cd:13:f8   D        vtep.32773
192.168.0.2
   VNI_100000         06:4b:8c:cd:c4:38   D        vtep.32787
192.168.0.3
   VNI_100000         02:0c:10:09:02:01   DR       esi.2467
00:00:00:00:00:01:00:00:00:02
   VNI_100000         06:38:e1:6f:30:29   D        vtep.32796
192.168.0.4

MAC flags (S - static MAC, D - dynamic MAC, L - locally learned, P - Persistent static
           SE - statistics enabled, NM - non configured MAC, R - remote PE MAC, O - ovsdb MAC)



Ethernet switching table : 5 entries, 5 learned
Routing instance : VS1
   Vlan               MAC                 MAC      Logical              Active
   name               address             flags    interface            source
   VNI_90000          00:00:5e:90:00:00   DR       esi.2455
05:19:17:f3:41:00:01:5f:90:00
   VNI_90000          06:4b:8c:cd:13:f8   D        vtep.32773
192.168.0.2
   VNI_90000          06:4b:8c:cd:c4:38   D        vtep.32787
192.168.0.3
   VNI_90000          02:0c:10:08:02:01   DR       esi.2467
00:00:00:00:00:01:00:00:00:02
   VNI_90000          06:38:e1:6f:30:29   D        vtep.32796           192.168.0.4
```

5. Verify the end system MAC address is reachable from all three leaf devices.

```
user@spine-1> show ethernet-switching vxlan-tunnel-end-point esi | find esi.2467
00:00:00:00:00:01:00:00:00:02 VS1                      2467  2097182 esi.2467            3
    RVTEP-IP             RVTEP-IFL      VENH     MASK-ID   FLAGS
    192.168.1.10         vtep.32789     2522     2         2
    192.168.1.11         vtep.32782     2475     1         2
    192.168.1.12         vtep.32779     2466     0         2
ESI                            RTT                      VLNBH INH     ESI-IFL   LOC-IFL
#RVTEPs
```

6. Verify the end system is reachable through the forwarding table.

```
user@spine-1> show route forwarding-table table VS1 destination 02:0c:10:09:02:01/48 extensive
Routing table: VS1.evpn-vxlan [Index 11]
Bridging domain: VNI_100000.evpn-vxlan [Index 9]
VPLS:
Enabled protocols: Bridging, ACKed by all peers, EVPN VXLAN,

Destination:  02:0c:10:09:02:01/48
  Learn VLAN: 0                        Route type: user
  Route reference: 0                   Route interface-index: 676
  Multicast RPF nh index: 0
  P2mpidx: 0
  IFL generation: 0                    Epoch: 0
  Sequence Number: 0                   Learn Mask: 0x400000000000000006000000000000000000
  L2 Flags: control_dyn, esi
  Flags: sent to PFE
  Next-hop type: indirect              Index: 2097182  Reference: 3
  Nexthop:
  Next-hop type: composite             Index: 2467     Reference: 2
  Nexthop:
  Next-hop type: composite             Index: 2522     Reference: 6
  Next-hop type: indirect              Index: 2097154  Reference: 5
  Nexthop: 172.16.10.1
  Next-hop type: unicast               Index: 2172     Reference: 11
  Next-hop interface: ae10.0
  Nexthop:
  Next-hop type: composite             Index: 2475     Reference: 6
  Next-hop type: indirect              Index: 2097193  Reference: 5
  Nexthop: 172.16.11.1
  Next-hop type: unicast               Index: 2194     Reference: 11
  Next-hop interface: ae11.0
  Nexthop:
  Next-hop type: composite             Index: 2466     Reference: 6
  Next-hop type: indirect              Index: 2097195  Reference: 5
  Nexthop: 172.16.12.1
  Next-hop type: unicast               Index: 2916     Reference: 11
  Next-hop interface: ae12.0
```

7. Verify end system information (MAC address, IP address, etc.) has been added to the IPv4 ARP table and IPv6 neighbor table.

```
user@spine-1> show arp no-resolve expiration-time | match "irb.900|irb.1000"
06:4b:8c:cd:13:f8 10.1.8.2        irb.900 [vtep.32773]     none 1035
06:4b:8c:cd:c4:38 10.1.8.3        irb.900 [vtep.32787]     none 1064
06:38:e1:6f:30:29 10.1.8.4        irb.900 [vtep.32796]     none 964
02:0c:10:08:02:01 10.1.8.201      irb.900 [.local..11]     none 781
06:4b:8c:cd:13:f8 10.1.9.2        irb.1000 [vtep.32773]    none 910
06:4b:8c:cd:c4:38 10.1.9.3        irb.1000 [vtep.32787]    none 1344
06:38:e1:6f:30:29 10.1.9.4        irb.1000 [vtep.32796]    none 1160
02:0c:10:09:02:01 10.1.9.201      irb.1000 [.local..11]    none 1099
user@spine-1> show ipv6 neighbors | match "irb.900|irb.1000"
2001:db8::10:1:8:2           06:4b:8c:cd:13:f8  stale       969 yes no      irb.900
[vtep.32773]
2001:db8::10:1:8:3           06:4b:8c:cd:c4:38  stale       1001 yes no     irb.900
[vtep.32787]
2001:db8::10:1:8:4           06:38:e1:6f:30:29  stale       971 yes no      irb.900
[vtep.32796]
2001:db8::10:1:8:201         02:0c:10:08:02:01  stale       1178 no no      irb.900
[.local..11]
2001:db8::10:1:9:2           06:4b:8c:cd:13:f8  stale       955 yes no      irb.1000
[vtep.32773]
2001:db8::10:1:9:3           06:4b:8c:cd:c4:38  stale       1006 yes no     irb.1000
[vtep.32787]
2001:db8::10:1:9:4           06:38:e1:6f:30:29  stale       990 yes no      irb.1000
[vtep.32796]
2001:db8::10:1:9:201         02:0c:10:09:02:01  stale       1199 no no      irb.1000
[.local..11]
fe80::10:1:8:2               06:4b:8c:cd:13:f8  stale       991 yes no      irb.900
[vtep.32773]
fe80::10:1:8:3               06:4b:8c:cd:c4:38  stale       989 yes no      irb.900
[vtep.32787]
fe80::10:1:8:4               06:38:e1:6f:30:29  stale       966 yes no      irb.900
[vtep.32796]
fe80::10:1:9:2               06:4b:8c:cd:13:f8  stale       978 yes no      irb.1000
[vtep.32773]
fe80::10:1:9:3               06:4b:8c:cd:c4:38  stale       994 yes no      irb.1000
[vtep.32787]
fe80::10:1:9:4               06:38:e1:6f:30:29  stale       1006 yes no     irb.1000
[vtep.32796]
```

8. Verify that the EVPN database contains the MAC address (02:0c:10:08:02:01) and ARP information learned from an end system connected to the leaf device.

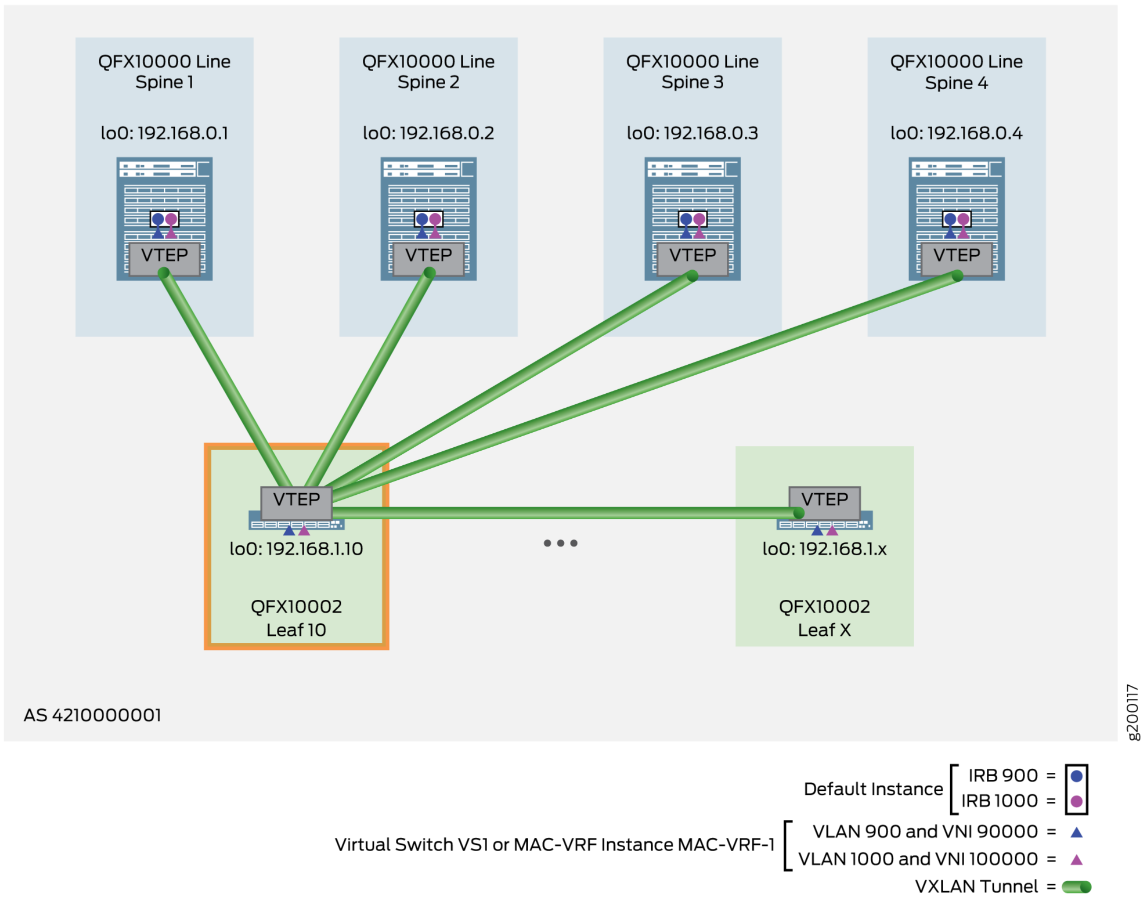
```
user@spine-1> show evpn database mac-address 02:0c:10:08:02:01 extensive
Instance: VS1

VN Identifier: 90000, MAC address:: 02:0c:10:08:02:01
  Source: 00:00:00:00:00:01:00:00:00:02, Rank: 1, Status: Active
    Remote origin: 192.168.1.10
    Remote origin: 192.168.1.11
    Remote origin: 192.168.1.12
    Timestamp: Sep 10 23:47:37 (0x59b63189)
    State: <Remote-To-Local-Adv-Done>
    IP address: 10.1.8.201
      Flags: <Proxy>
      Remote origin: 192.168.1.10
      Remote origin: 192.168.1.11
      Remote origin: 192.168.1.12
    IP address: 2001:db8::10:1:8:201
      Remote origin: 192.168.1.10
      Remote origin: 192.168.1.11
      Remote origin: 192.168.1.12    History db:
      Time                  Event
      Sep 10 23:47:39 2017  Applying remote state to peer 192.168.1.11
      Sep 10 23:47:39 2017  Remote peer 192.168.1.11 updated
      Sep 10 23:47:39 2017  MAC+IP not updated, source l2ald is not owner (type2)
      Sep 10 23:47:39 2017  Updated
      Sep 10 23:47:39 2017  No change to MAC state
      Sep 10 23:47:39 2017  Applying remote state to peer 192.168.1.12
      Sep 10 23:47:39 2017  Remote peer 192.168.1.12 updated
      Sep 10 23:47:39 2017  MAC+IP not updated, source l2ald is not owner (type2)
      Sep 10 23:47:39 2017  Updated
      Sep 10 23:47:39 2017  No change to MAC state
```

## Configuring the VLAN-Aware CRB Overlay with Virtual Switches or MAC-VRF Instances on a Leaf Device

To configure a VLAN-aware CRB overlay in a virtual switch or a MAC-VRF instance on a leaf device, perform the following:

**NOTE**: The following example shows the configuration for Leaf 10, as shown in Figure 49 on page 189.

**Figure 49: VLAN-Aware CRB Overlay with Virtual Switches or MAC-VRF Instances – Leaf Device**



1. Ensure the IP fabric underlay is in place. To configure an IP fabric on leaf devices, see "IP Fabric Underlay Network Design and Implementation" on page 80.
2. Confirm that your IBGP overlay is up and running. To configure an IBGP overlay on your leaf devices, see "Configure IBGP for the Overlay" on page 96.
3. Configure a virtual switch instance (VS1) or a MAC-VRF instance (MAC-VRF-1) to enable EVPN-VXLAN. Also, map VLANs 900 and 1000 to VNIs 90000 and 100000 in the instance.

   For a virtual switch instance, use `instance-type virtual-switch`.

*Leaf 10 (Virtual Switch Instance)*:

```
set routing-instances VS1 vtep-source-interface lo0.0
set routing-instances VS1 instance-type virtual-switch
set routing-instances VS1 route-distinguisher 192.168.1.10:900
set routing-instances VS1 vrf-target target:62273:90000
set routing-instances VS1 vrf-target auto
set routing-instances VS1 protocols evpn encapsulation vxlan
set routing-instances VS1 protocols evpn extended-vni-list all
set routing-instances VS1 protocols evpn default-gateway no-gateway-community
set routing-instances VS1 vlans VNI_90000 interface ae12.900
set routing-instances VS1 vlans VNI_90000 vxlan vni 90000
set routing-instances VS1 vlans VNI_100000 interface ae12.1000
set routing-instances VS1 vlans VNI_100000 vxlan vni 100000
```

With MAC-VRF instances, use `instance-type mac-vrf`. You also configure the service type when you create the MAC-VRF instance. Here we configure `service-type vlan-aware` with the two VLANs VNI_90000 and VNI_100000, and their VNI mappings.

*Leaf 10 (MAC-VRF Instance)*:

```
set routing-instances MAC-VRF-1 vtep-source-interface lo0.0
set routing-instances MAC-VRF-1 instance-type mac-vrf
set routing-instances MAC-VRF-1 service-type vlan-aware
set routing-instances MAC-VRF-1 route-distinguisher 192.168.1.10:900
set routing-instances MAC-VRF-1 vrf-target target:62273:90000
set routing-instances MAC-VRF-1 vrf-target auto
set routing-instances MAC-VRF-1 protocols evpn encapsulation vxlan
set routing-instances MAC-VRF-1 protocols evpn extended-vni-list all
set routing-instances MAC-VRF-1 protocols evpn default-gateway no-gateway-community
set routing-instances MAC-VRF-1 vlans VNI_90000 interface ae12.900
set routing-instances MAC-VRF-1 vlans VNI_90000 vxlan vni 90000
set routing-instances MAC-VRF-1 vlans VNI_100000 interface ae12.1000
set routing-instances MAC-VRF-1 vlans VNI_100000 vxlan vni 100000
```

4. (MAC-VRF instances only) Enable shared tunnels on the device.

   A device can have problems with VTEP scaling when the configuration uses multiple MAC-VRF instances. As a result, to avoid this problem, we require that you enable the shared tunnels feature on the QFX5000 line of switches with a MAC-VRF instance configuration. When you configure the shared-tunnels option, the device minimizes the number of next-hop entries to reach remote VTEPs. The following statement globally enables shared VXLAN tunnels on the device:

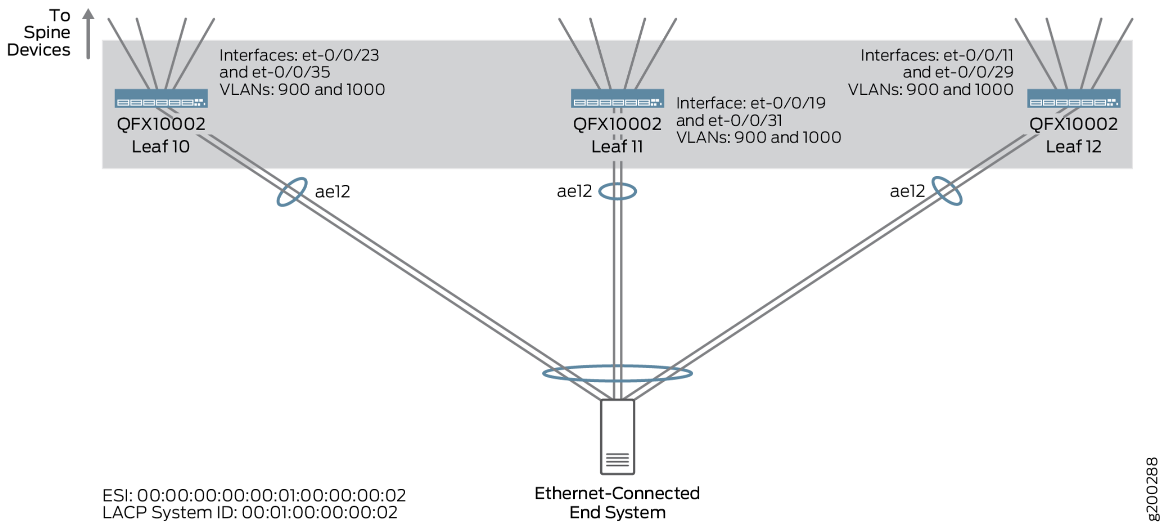
```
set forwarding-options evpn-vxlan shared-tunnels
```

This statement is optional on the QFX10000 line of switches, which can handle higher VTEP scaling than QFX5000 switches.

**NOTE**: This setting requires you to reboot the device.

5. Configure the leaf device to communicate with the end system. In this example, configure an aggregated Ethernet interface on Leaf 10—in this case, ae12 with two member interfaces. With the interface definition, include LACP options, an ESI in all-active mode, and VLANs 900 and 1000 (which this example uses for the VLAN-aware service type). Figure 50 on page 191 illustrates the topology.

**Figure 50: ESI Topology for Leaf 10, Leaf 11, and Leaf 12**


To Spine Devices
Interfaces: et-0/0/23 and et-0/0/35
VLANs: 900 and 1000
QFX10002
Leaf 10
QFX10002
Leaf 11
Interface: et-0/0/19 and et-0/0/31
VLANs: 900 and 1000
Interfaces: et-0/0/11 and et-0/0/29
VLANs: 900 and 1000
QFX10002
Leaf 12
ae12
ae12
ae12
ESI: 00:00:00:00:00:01:00:00:00:02
LACP System ID: 00:01:00:00:00:02
Ethernet-Connected End System
g200288

*Leaf 10*:

```
set interfaces ae12 flexible-vlan-tagging
set interfaces ae12 encapsulation extended-vlan-bridge
set interfaces ae12 esi 00:00:00:00:00:01:00:00:00:02
set interfaces ae12 esi all-active
set interfaces ae12 aggregated-ether-options minimum-links 1
set interfaces ae12 aggregated-ether-options lacp active
set interfaces ae12 aggregated-ether-options lacp periodic fast
set interfaces ae12 aggregated-ether-options lacp system-id 00:01:00:00:00:02
set interfaces ae12 aggregated-ether-options lacp hold-time up 300
set interfaces ae12 unit 900 vlan-id 900
set interfaces ae12 unit 1000 vlan-id 1000
set interfaces et-0/0/23 ether-options 802.3ad ae12
set interfaces et-0/0/35 ether-options 802.3ad ae12
```

Note that in this example, you configure the aggregated Ethernet interface to support the service provider configuration style. See Flexible Ethernet Service Encapsulation for more information on the service provider style configuration for switch interfaces.

## Verifying the VLAN-Aware CRB Overlay with Virtual Switches or MAC-VRF Instances on a Leaf Device

To verify this style of overlay on a leaf device, run the commands in this section.

Most commands here show output for a virtual switch instance configuration. With a MAC-VRF instance configuration, you can alternatively use:

- `show mac-vrf forwarding` commands that are aliases for the `show ethernet-switching` commands in this section.
- The `show mac-vrf routing instance` command, which is an alias for the `show evpn instance` command in this section.

See MAC-VRF Routing Instance Type Overview for tables of `show mac-vrf forwarding` and `show ethernet-switching` command mappings, and `show mac-vrf routing` command aliases for `show evpn` commands.

Otherwise, you can use the commands in this section for either virtual switch instances or MAC-VRF instances.

The output with a MAC-VRF instance configuration displays similar information for MAC-VRF routing instances as this section shows for virtual switch instances. One main difference you might see is in the output with MAC-VRF instances on devices where you enable the shared tunnels feature. With shared tunnels enabled, you see VTEP interfaces in the following format:

```
vtep-index.shared-tunnel-unit
```

where:

- *index* is the index associated with the MAC-VRF routing instance.

- *shared-tunnel-unit* is the unit number associated with the shared tunnel remote VTEP logical interface.

For example, if a device has a MAC-VRF instance with index 26 and the instance connects to two remote VTEPs, the shared tunnel VTEP logical interfaces might look like this:

```
vtep-26.32823
vtep-26.32824
```

1. Verify that the aggregated Ethernet interface is operational on the leaf device.

   ```
   user@leaf-10> show interfaces terse ae12
   Interface               Admin Link Proto    Local                 Remote
   ae12                    up    up
   ae12.900                up    up   eth-switch
   ae12.1000               up    up   eth-switch
   ae12.32767              up    up
   ```

2. (MAC-VRF instances only) Verify the VLANs you configured as part of the MAC-VRF instance.

   ```
   user@leaf-10> show mac-vrf forwarding instance MAC-VRF-1
   Information for routing instance and VLAN:

   Flags (DL - disable learning, SE - stats enabled,
          AD - packet action drop, LH - MAC limit hit,
          MI - mac+ip limit hit)

   Inst Logical   Routing             VLAN name          Index IRB   Flags Tag
   type system    instance                                     index
   RTT  Default   MAC-VRF-1                              27
   vlan Default   MAC-VRF-1           VNI-90000          3424              NA
   vlan Default   MAC-VRF-1           VNI-100000         3425              NA
   .
   .
   .

   user@leaf-10> show vlans VNI-90000
   Routing instance        VLAN name             Tag          Interfaces
   MAC-VRF-1               VNI-90000             NA
                                                              ae12.900*
                                                              esi.19955*
   ```

```
                                                                            esi.20938*
                                                                            vtep-27.32820*
                                                                            vtep-27.32821*
                                                                            vtep-27.32822*
                                                                            xe-0/0/18:2.900*
```

3. Verify switching details about the EVPN routing instance. This output includes information about the route distinguisher (192.168.1.10:900), VXLAN encapsulation, ESI (00:00:00:00:00:01:00:00:00:02), verification of the VXLAN tunnels for VLANs 900 and 1000, EVPN neighbors (Spine 1 - 4, and Leaf 11 and 12), and the source VTEP IP address (192.168.1.10).

```
user@leaf-10> show evpn instance VS1 extensive
Instance: VS1
  Route Distinguisher: 192.168.1.10:900
  Encapsulation type: VXLAN
  MAC database status                     Local  Remote
    MAC advertisements:                       0      20
    MAC+IP advertisements:                    0      32
    Default gateway MAC advertisements:       0       0
  Number of local interfaces: 2 (2 up)
    Interface name  ESI                            Mode             Status     AC-Role
    ae12.1000       00:00:00:00:00:01:00:00:00:02  all-active       Up         Root
    ae12.900        00:00:00:00:00:01:00:00:00:02  all-active       Up         Root
  Number of IRB interfaces: 0 (0 up)
  Number of bridge domains: 2
    VLAN  Domain ID   Intfs / up    IRB intf   Mode             MAC sync  IM route label  SG
sync  IM core nexthop
    None  90000          1    1                Extended         Enabled   90000
Disabled
    None  100000         1    1                Extended         Enabled   100000
Disabled
  Number of neighbors: 6
    Address               MAC    MAC+IP        AD        IM        ES Leaf-label
    192.168.0.1             4        10         2         2         0
    192.168.0.2             4        10         2         2         0
    192.168.0.3             4        10         2         2         0
    192.168.0.4             4        10         2         2         0
    192.168.1.11            2         4         2         2         0
    192.168.1.12            2         4         2         2         0
  Number of ethernet segments: 3
    ESI: 00:00:00:00:00:01:00:00:00:02
      Status: Resolved by IFL ae12.900
```

```
        Local interface: ae12.1000, Status: Up/Forwarding
        Number of remote PEs connected: 2
          Remote PE        MAC label  Aliasing label  Mode
          192.168.1.12     100000     0               all-active
          192.168.1.11     90000      0               all-active
        DF Election Algorithm: MOD based
        Designated forwarder: 192.168.1.10
        Backup forwarder: 192.168.1.11
        Backup forwarder: 192.168.1.12
        Last designated forwarder update: Sep 10 23:22:07
      ESI: 05:19:17:f3:41:00:01:5f:90:00
        Status: Resolved
        Number of remote PEs connected: 4
          Remote PE        MAC label  Aliasing label  Mode
          192.168.0.1      90000      0               all-active
          192.168.0.3      90000      0               all-active
          192.168.0.2      90000      0               all-active
          192.168.0.4      90000      0               all-active
      ESI: 05:19:17:f3:41:00:01:86:a0:00
        Status: Resolved
        Number of remote PEs connected: 4
          Remote PE        MAC label  Aliasing label  Mode
          192.168.0.1      100000     0               all-active
          192.168.0.3      100000     0               all-active
          192.168.0.2      100000     0               all-active
          192.168.0.4      100000     0               all-active
    Router-ID: 192.168.1.10
    Source VTEP interface IP: 192.168.1.10
```

4. View the MAC address table on the leaf device to confirm that spine device and end system MAC addresses appear in the table.

**NOTE:**

- 00:00:5e:90:00:00 and 00:00:5e:a0:00:00 are the IP subnet gateways on the spine device.
- 02:0c:10:09:02:01 and 02:0c:10:08:02:01 are end systems connected through the leaf device.

```
user@leaf-10> show ethernet-switching table instance VS1
```

```
MAC flags (S - static MAC, D - dynamic MAC, L - locally learned, P - Persistent static
           SE - statistics enabled, NM - non configured MAC, R - remote PE MAC, O - ovsdb MAC)


Ethernet switching table : 6 entries, 6 learned
Routing instance : VS1
   Vlan                MAC                 MAC      Logical                Active
   name                address             flags    interface              source
   VNI_100000          00:00:5e:a0:00:00   DR       esi.2139
05:19:17:f3:41:00:01:86:a0:00
   VNI_100000          06:4b:8c:67:0f:f0   D        vtep.32799
192.168.0.1
   VNI_100000          06:4b:8c:cd:13:f8   D        vtep.32798
192.168.0.2
   VNI_100000          06:4b:8c:cd:c4:38   D        vtep.32804
192.168.0.3
   VNI_100000          02:0c:10:09:02:01   DR       ae12.1000
   VNI_100000          06:38:e1:6f:30:29   D        vtep.32807
192.168.0.4

MAC flags (S - static MAC, D - dynamic MAC, L - locally learned, P - Persistent static
           SE - statistics enabled, NM - non configured MAC, R - remote PE MAC, O - ovsdb MAC)


Ethernet switching table : 6 entries, 6 learned
Routing instance : VS1
   Vlan                MAC                 MAC      Logical                Active
   name                address             flags    interface              source
   VNI_90000           00:00:5e:90:00:00   DR       esi.2144
05:19:17:f3:41:00:01:5f:90:00
   VNI_90000           06:4b:8c:67:0f:f0   D        vtep.32799
192.168.0.1
   VNI_90000           06:4b:8c:cd:13:f8   D        vtep.32798
192.168.0.2
   VNI_90000           06:4b:8c:cd:c4:38   D        vtep.32804
192.168.0.3
   VNI_90000           02:0c:10:08:02:01   DR       ae12.900
   VNI_90000           06:38:e1:6f:30:29   D        vtep.32807             192.168.0.4
```

5. Verify that the IP subnet gateway ESIs discovered in Step 3 (esi.2144 for VNI 90000 and esi.2139 for VNI 100000) are reachable from all four spine devices.

```
user@leaf-10> show ethernet-switching vxlan-tunnel-end-point esi | find esi.2144
05:19:17:f3:41:00:01:5f:90:00 VS1                      2144  2097224 esi.2144           4
    RVTEP-IP             RVTEP-IFL      VENH     MASK-ID   FLAGS
    192.168.0.4          vtep.32807     2033     3         2
    192.168.0.2          vtep.32798     2092     2         2
    192.168.0.3          vtep.32804     2174     1         2
    192.168.0.1          vtep.32799     2093     0         2
ESI                           RTT                      VLNBH INH     ESI-IFL   LOC-IFL
#RVTEPs
user@leaf-10> show ethernet-switching vxlan-tunnel-end-point esi | find esi.2139
05:19:17:f3:41:00:01:86:a0:00 VS1                      2139  2097221 esi.2139           4
    RVTEP-IP             RVTEP-IFL      VENH     MASK-ID   FLAGS
    192.168.0.4          vtep.32807     2033     3         2
    192.168.0.2          vtep.32798     2092     2         2
    192.168.0.3          vtep.32804     2174     1         2
    192.168.0.1          vtep.32799     2093     0         2
```

6. Verify the IP subnet gateway on the spine device (00:00:5e:a0:00:00) is reachable through the forwarding table.

```
user@leaf-10> show route forwarding-table table VS1 destination 00:00:5e:a0:00:00/48 extensive
Routing table: VS1.evpn-vxlan [Index 10]
Bridging domain: VNI_100000.evpn-vxlan [Index 15]
VPLS:
Enabled protocols: Bridging, ACKed by all peers, EVPN VXLAN,

Destination:  00:00:5e:a0:00:00/48
  Learn VLAN: 0                        Route type: user
  Route reference: 0                   Route interface-index: 571
  Multicast RPF nh index: 0
  P2mpidx: 0
  IFL generation: 0                    Epoch: 0
  Sequence Number: 0                   Learn Mask: 0x400000000000000f000000000000000000000
  L2 Flags: control_dyn, esi
  Flags: sent to PFE
  Next-hop type: indirect              Index: 2097221  Reference: 2
  Nexthop:
  Next-hop type: composite             Index: 2139     Reference: 2
  Nexthop:
  Next-hop type: composite             Index: 2033     Reference: 9
  Next-hop type: indirect              Index: 2097223  Reference: 5
  Nexthop: 172.16.10.14
  Next-hop type: unicast               Index: 2106     Reference: 10
  Next-hop interface: ae4.0
  Nexthop:
  Next-hop type: composite             Index: 2092     Reference: 9
  Next-hop type: indirect              Index: 2097172  Reference: 5
  Nexthop: 172.16.10.6
  Next-hop type: unicast               Index: 1951     Reference: 11
  Next-hop interface: ae2.0
  Nexthop:
  Next-hop type: composite             Index: 2174     Reference: 9
  Next-hop type: indirect              Index: 2097174  Reference: 5
  Nexthop: 172.16.10.10
  Next-hop type: unicast               Index: 2143     Reference: 11
  Next-hop interface: ae3.0
  Nexthop:
  Next-hop type: composite             Index: 2093     Reference: 9
  Next-hop type: indirect              Index: 2097165  Reference: 5
  Nexthop: 172.16.10.2
  Next-hop type: unicast               Index: 2153     Reference: 11
  Next-hop interface: ae1.0
```

SEE ALSO

*Overview of VLAN Services for EVPN*

*EVPN Overview*

*Understanding VXLANs*

*Understanding EVPN with VXLAN Data Plane Encapsulation*

*Configuring EVPN Routing Instances*

MAC-VRF Routing Instance Type Overview

*Configuring Aggregated Ethernet LACP (CLI Procedure)*

## Centrally-Routed Bridging Overlay — Release History

Table 5 on page 199 provides a history of all of the features in this section and their support within this reference design.

**Table 5: CRB Overlay in the Data Center Fabric Reference Design– Release History**

| Release | Description |
|---|---|
| 19.1R2 | QFX10002-60C and QFX5120-32C switches running Junos OS Release 19.1R2 and later releases in the same release train support all features documented in this section. |
| 17.3R3-S2 | Adds support for Contrail Enterprise Multicloud, where you can configure CRB overlays from the Contrail Command GUI. |
| 17.3R3-S1 | All devices in the reference design that support Junos OS Release 17.3R3-S1 and later releases in the same release train also support all features documented in this section |

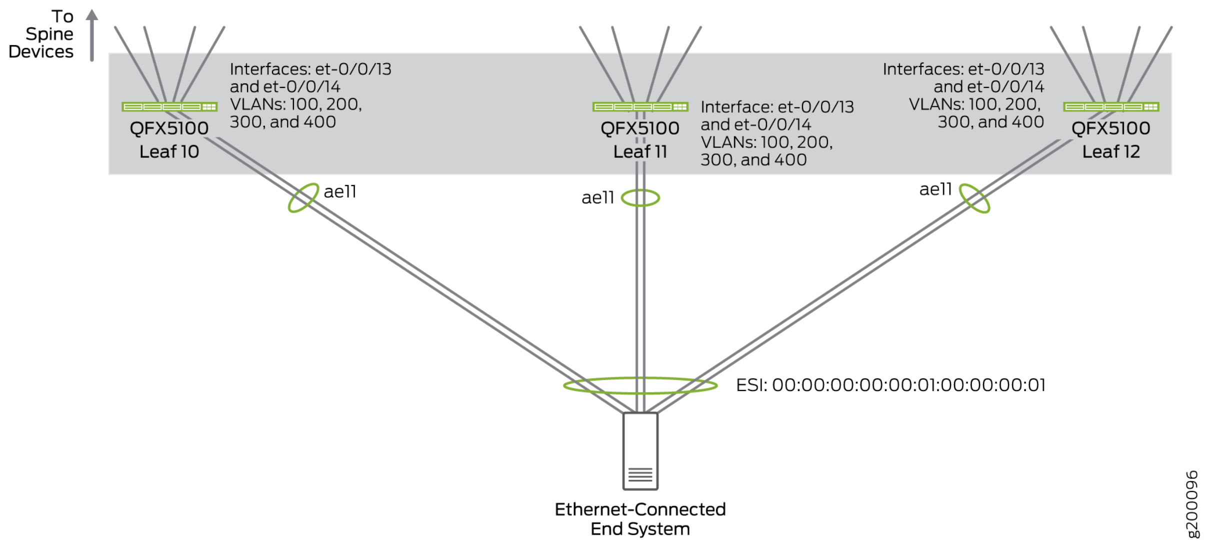
# Multihoming an Ethernet-Connected End System Design and Implementation

**IN THIS SECTION**

For an overview of multihoming an Ethernet-connected end system in this reference design, see the Multihoming Support for Ethernet-Connected End Systems section in "Data Center Fabric Blueprint Architecture Components" on page 9.

Figure 51 on page 200 illustrates the multihomed Ethernet-connected end system in this procedure:

**Figure 51: Ethernet-Connected Multihoming Example Overview**


To Spine Devices
Interfaces: et-0/0/13 and et-0/0/14 VLANs: 100, 200, 300, and 400
QFX5100 Leaf 10
Interface: et-0/0/13 and et-0/0/14 VLANs: 100, 200, 300, and 400
QFX5100 Leaf 11
Interfaces: et-0/0/13 and et-0/0/14 VLANs: 100, 200, 300, and 400
QFX5100 Leaf 12
ae11
ae11
ae11
ESI: 00:00:00:00:00:01:00:00:00:01
Ethernet-Connected End System
g200096

## Configuring a Multihomed Ethernet-Connected End System using EVPN Multihoming with VLAN Trunking

EVPN multihoming is used in this building block to connect an Ethernet-connected end system into the overlay network. EVPN multihoming works by treating two or more physical multihomed links as a single Ethernet segment identified by an EVPN Ethernet Segment ID (ESI). The set of physical links belonging to the same Ethernet segment are treated as one aggregated Ethernet interface. The member links—much like member links in a traditional aggregated Ethernet interface—provide redundant paths to and from the end system while also ensuring overlay network traffic is load-balanced across the multiple paths.

LACP with the fast timer mode is used to improve fault detection and disablement of impaired members of an Ethernet segment. MicroBFD may also be used to further improve fault isolation but may not scale to support all end-system facing ports. Furthermore, support for microBFD must exist at the end system.

The reference design tested an Ethernet-connected server was connected to a single leaf or multihomed to 2 or 3 leaf devices to verify that traffic can be properly handled in multihomed setups with more than 2 leaf devices; in practice, an Ethernet-connected server can be multihomed to a large number of leaf devices.

To configure a multihomed Ethernet-connected server:

1. (Aggregated Ethernet interfaces only) Create the aggregated Ethernet interfaces to connect each leaf device to the server. Enable LACP with a fast period interval for each aggregated Ethernet interface.

*Leaf 10*:

```
set interfaces et-0/0/13 ether-options 802.3ad ae11
set interfaces et-0/0/14 ether-options 802.3ad ae11
set interfaces ae11 aggregated-ether-options lacp active
set interfaces ae11 aggregated-ether-options lacp periodic fast
```

*Leaf 11*:

```
set interfaces et-0/0/13 ether-options 802.3ad ae11
set interfaces et-0/0/14 ether-options 802.3ad ae11
set interfaces ae11 aggregated-ether-options lacp active
set interfaces ae11 aggregated-ether-options lacp periodic fast
```

*Leaf 12*:

```
set interfaces et-0/0/13 ether-options 802.3ad ae11
set interfaces et-0/0/14 ether-options 802.3ad ae11
set interfaces ae11 aggregated-ether-options lacp active
set interfaces ae11 aggregated-ether-options lacp periodic fast
```

**NOTE**: The three leaf devices in this step use the same aggregated Ethernet interface name—ae11—and member link interfaces—et-0/0/13 and et-0/0/14— to organize and simplify network administration.

Avoid using different AE names at each VTEP for the same ESI as this will require configuring the LACP admin-key so that the end system can identify the multihomed links as part of the same LAG.

2. Configure each interface into a trunk interface. Assign VLANs to each trunk interface.

**NOTE**: If you are connecting your end system to the leaf device with a single link, replace the interface name—for example, *ae11*—with a physical interface name—for example, *et-0/0/13*—for the remainder of this procedure.

*Leaf 10*:

```
set interfaces ae11 unit 0 family ethernet-switching interface-mode trunk
set interfaces ae11 unit 0 family ethernet-switching vlan members 100
set interfaces ae11 unit 0 family ethernet-switching vlan members 200
set interfaces ae11 unit 0 family ethernet-switching vlan members 300
set interfaces ae11 unit 0 family ethernet-switching vlan members 400
```

*Leaf 11*:

```
set interfaces ae11 unit 0 family ethernet-switching interface-mode trunk
set interfaces ae11 unit 0 family ethernet-switching vlan members 100
set interfaces ae11 unit 0 family ethernet-switching vlan members 200
set interfaces ae11 unit 0 family ethernet-switching vlan members 300
set interfaces ae11 unit 0 family ethernet-switching vlan members 400
```

*Leaf 12*:

```
set interfaces ae11 unit 0 family ethernet-switching interface-mode trunk
set interfaces ae11 unit 0 family ethernet-switching vlan members 100
set interfaces ae11 unit 0 family ethernet-switching vlan members 200
set interfaces ae11 unit 0 family ethernet-switching vlan members 300
set interfaces ae11 unit 0 family ethernet-switching vlan members 400
```

3. Configure the multihomed links with an ESI.

   Assign each multihomed interface into the ethernet segment—which is identified using the Ethernet Segment Identifier (ESI)—that is hosting the Ethernet-connected server. Ensure traffic is passed over all multihomed links by configuring each link as *all-active*.

   The ESI values must match on all multihomed interfaces.

   *Leaf 10* :

```
set interfaces ae11 esi 00:00:00:00:00:01:00:00:00:01
set interfaces ae11 esi all-active
```

   *Leaf 11*:

```
set interfaces ae11 esi 00:00:00:00:00:01:00:00:00:01
set interfaces ae11 esi all-active
```

*Leaf 12*:

```
set interfaces ae11 esi 00:00:00:00:00:01:00:00:00:01
set interfaces ae11 esi all-active
```

4. Enable LACP and configure a system identifier.

   The LACP system identifier must match on all multihomed interfaces.

   *Leaf 10*:

```
set interfaces ae11 aggregated-ether-options lacp system-id 00:01:00:00:00:01
```

   *Leaf 11*:

```
set interfaces ae11 aggregated-ether-options lacp system-id 00:01:00:00:00:01
```

   *Leaf 12*:

```
set interfaces ae11 aggregated-ether-options lacp system-id 00:01:00:00:00:01
```

5. After committing the configuration, verify that the links on each leaf switch are in the *Up* state

   Example:

```
user@leaf10# run show interfaces terse ae11
Interface               Admin Link Proto    Local                 Remote
ae11                    up    up
ae11.0                  up    up   eth-switch
```

6. Verify that LACP is operational on the multihomed links.

```
user@leaf10# run show lacp interfaces ae11
Aggregated interface: ae11
    LACP state:       Role   Exp   Def  Dist  Col  Syn  Aggr  Timeout  Activity
      et-0/0/13      Actor    No    No   Yes  Yes  Yes   Yes     Fast    Active
      et-0/0/13    Partner    No    No   Yes  Yes  Yes   Yes     Fast    Active
      et-0/0/14      Actor    No    No   Yes  Yes  Yes   Yes     Fast    Active
      et-0/0/14    Partner    No    No   Yes  Yes  Yes   Yes     Fast    Active
    LACP protocol:        Receive State  Transmit State          Mux State
```

```
        et-0/0/13                 Current    Fast periodic Collecting distributing
        et-0/0/14                 Current    Fast periodic Collecting distributing
```

## Enabling Storm Control

Storm control can be enabled as part of this building block. Storm control is used to prevent BUM traffic storms by monitoring BUM traffic levels and taking a specified action to limit BUM traffic forwarding when a specified traffic level—called the storm control level—is exceeded. See Understanding Storm Control for additional information on the feature.

In this reference design, storm control is enabled on server-facing aggregated Ethernet interfaces to rate limit broadcast, unknown unicast, and multicast (BUM) traffic. If the amount of BUM traffic exceeds 1% of the available bandwidth on the aggregated Ethernet interface, storm control drops BUM traffic to prevent broadcast storms.

To enable storm control:

1. Create the storm control profile that will be used to enable the feature. The interfaces that are configured using the storm control profile are specified in this step.

   *Leaf Device*:

   ```
   set interfaces ae11 unit 0 family ethernet-switching storm-control STORM-CONTROL
   ```

2. Set the storm control configuration within the profile.

   In this reference design, storm control is configured to strategically drop BUM traffic when the amount of BUM traffic exceeds 1% of all available interface bandwidth.

   ```
   set forwarding-options storm-control-profiles STORM-CONTROL all bandwidth-percentage 1
   ```

   **NOTE**: Dropping BUM traffic is the only supported storm control action in the Cloud Data Center architecture.

   **NOTE**: The storm control settings in this version of the reference design drop multicast traffic that exceeds the configured storm control threshold. If your network supports multicast-based applications, consider using a storm control configuration—such as the `no-multicast` option in the `storm-control-profiles` statement—that is not represented in this reference design.

Storm control settings in support of multicast-based applications will be included in a future version of this reference design.

3. To verify storm control activity, filter system log messages related to storm control by entering the `show log messages | match storm` command.

```
user@leaf10> show log messages | match storm
Sep 27 11:35:34 leaf1-qfx5100 l2ald[1923]: L2ALD_ST_CTL_IN_EFFECT: ae11.0: storm control in
effect on the port
```

## Multihoming a Ethernet-Connected End System—Release History

Table 6 on page 205 provides a history of all of the features in this section and their support within this reference design.

**Table 6: Release History**

| Release | Description |
|---|---|
| 19.1R2 | QFX10002-60C and QFX5120-32C switches running Junos OS Release 19.1R2 and later releases in the same release train support all features documented in this section. |
| 18.4R2 | QFX5120-48Y switches running Junos OS Release 18.4R2 and later releases in the same release train support all features documented in this section. |
| 18.1R3-S3 | QFX5110 switches running Junos OS Release 18.1R3-S3 and later releases in the same release train support all features documented in this section. |
| 17.3R3-S1 | All devices in the reference design that support Junos OS Release 17.3R3-S1 and later releases in the same release train also support all features documented in this section. |

### RELATED DOCUMENTATION

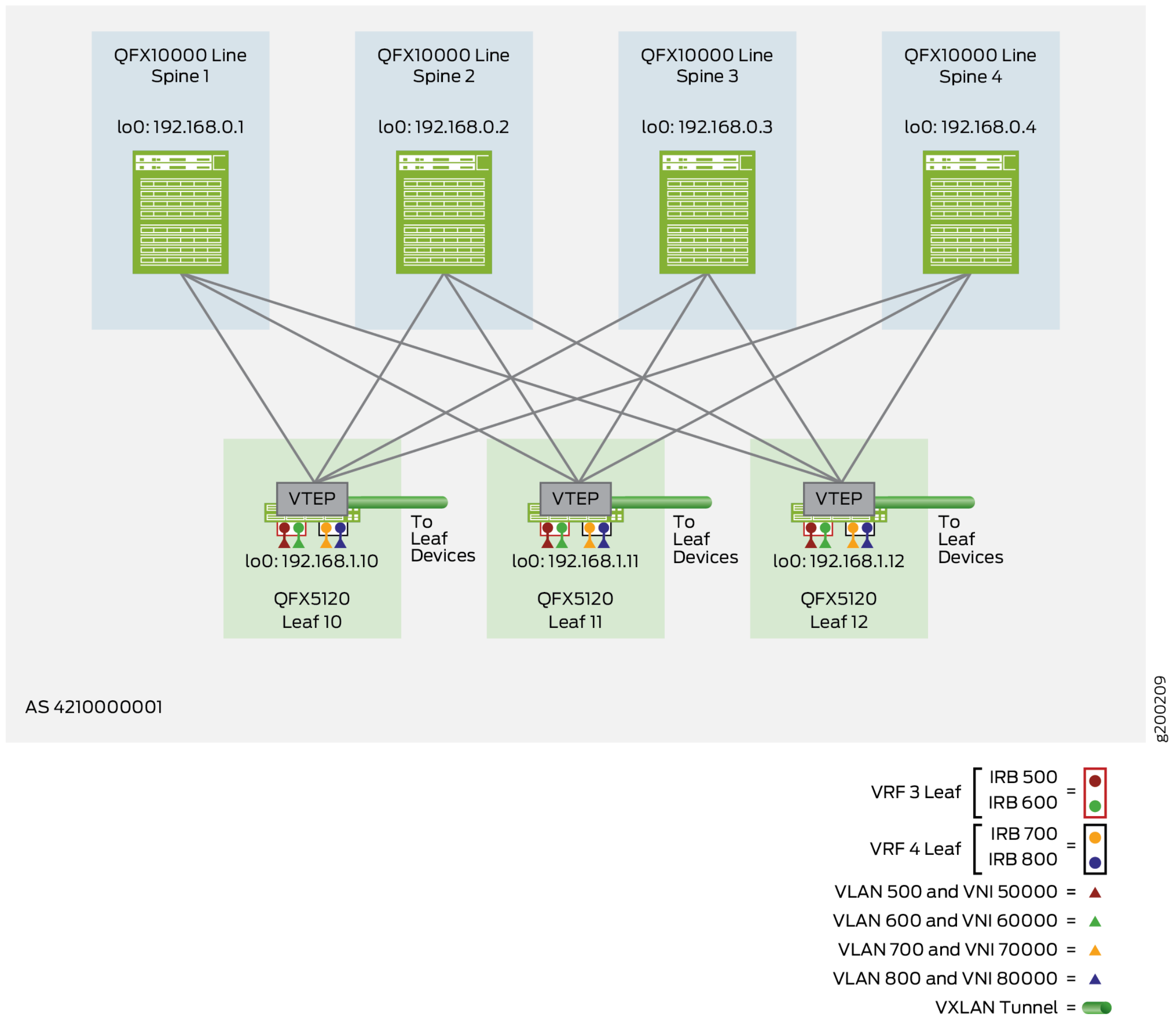
*EVPN Multihoming Overview*

# Edge-Routed Bridging Overlay Design and Implementation

**IN THIS SECTION**

A second overlay option for this reference design is the edge-routed bridging overlay, as shown in Figure 52 on page 207.

**Figure 52: Edge-Routed Bridging Overlay**



The edge-routed bridging overlay performs routing at IRB interfaces located at the edge of the overlay (most often at the leaf devices). As a result, Ethernet bridging and IP routing happen as close to the end systems as possible, but still support Ethernet dependent applications at the end system level.

You can configure edge-routed bridging overlay architectures using the traditional IPv4 EBGP underlay with IPv4 IBGP overlay, peering, or alternatively (with supported platforms) use an IPv6 EBGP underlay with IPv6 EBGP overlay peering. The configuration procedures in this section describe the configuration differences, where applicable, with an IPv6 Fabric instead of an IPv4 Fabric.

For a list of devices that we support as lean spine and leaf devices in an edge-routed bridging overlay, see the "Data Center EVPN-VXLAN Fabric Reference Designs—Supported Hardware Summary" on page 51. That list includes which devices support an IPv6 Fabric when serving in different device roles.

Lean spine devices handle only IP traffic, which removes the need to extend the bridging overlay to the lean spine devices. With this limited role, on these devices you configure only the IP fabric underlay and BGP overlay peering (either an IPv4 Fabric or an IPv6 Fabric).

On the leaf devices, you can configure an edge-routed bridging overlay using the default switch instance or using MAC-VRF instances.

**NOTE**: Keep the following in mind when you configure an ERB overlay for an EVPN-VXLAN network:

- On any Junos OS Evolved devices, we support EVPN-VXLAN configurations with MAC-VRF instances only.
- We support the IPv6 Fabric infrastructure design with MAC-VRF instances only.
- On the QFX5130 and QFX5700 switches in the EVPN fabric, make sure you configure the `host-profile` unified forwarding profile to support an EVPN-VXLAN environment (see *Layer 2 Forwarding Tables* for details):

  ```
  set system packet-forwarding-options forwarding-profile host-profile
  ```

Some configuration steps that affect the Layer 2 configuration differ with MAC-VRF instances. Likewise, a few steps differ for IPv6 Fabric configurations. The leaf device configuration includes the following steps:

- Configure the default instance with the loopback interface as a VTEP source interface. Or if your configuration uses MAC-VRF instances, configure a MAC-VRF instance with the loopback interface as a VTEP source interface. If your fabric uses an IPv6 Fabric, you configure the VTEP source interface as an IPv6 interface. In each MAC-VRF instance, you also configure a service type, a route distinguisher, and a route target.
- Configure a leaf-to-end system aggregated Ethernet interface as a trunk to carry multiple VLANs. With MAC-VRF instances, you also include the interface in the MAC-VRF instance.
- Establish LACP and ESI functionality.
- Map VLANs to VXLAN network identifiers (VNIs). For a MAC-VRF instance configuration, you configure the VLAN to VNI mappings in the MAC-VRF instance.
- Configure proxy-macip-advertisement, virtual gateways, and static MAC addresses on the IRB interfaces.
- Configure EVPN/VXLAN in the default instance or in the MAC-VRF instance.

- Enable Layer 3 (L3) tenant virtual routing and forwarding (VRF) instances and IP prefix route properties for EVPN Type 5.
- Optionally enable symmetric IRB routing with EVPN Type 2 routes on the leaf devices. Symmetric Type 2 routing avoids scaling issue when your EVPN network has many VLANs and many attached hosts or servers. With symmetric Type 2 routing, on each leaf device you only need to configure the VLANs that leaf devices serves. We support symmetric Type 2 routing only with MAC-VRF EVPN instances.

For an overview of edge-routed bridging overlays, see the Edge-Routed Bridging Overlay section in "Data Center Fabric Blueprint Architecture Components" on page 9.

For more information about MAC-VRF instances and using them in an example customer use case with an edge-routed bridging overlay, see EVPN-VXLAN DC IP Fabric MAC-VRF L2 Services.

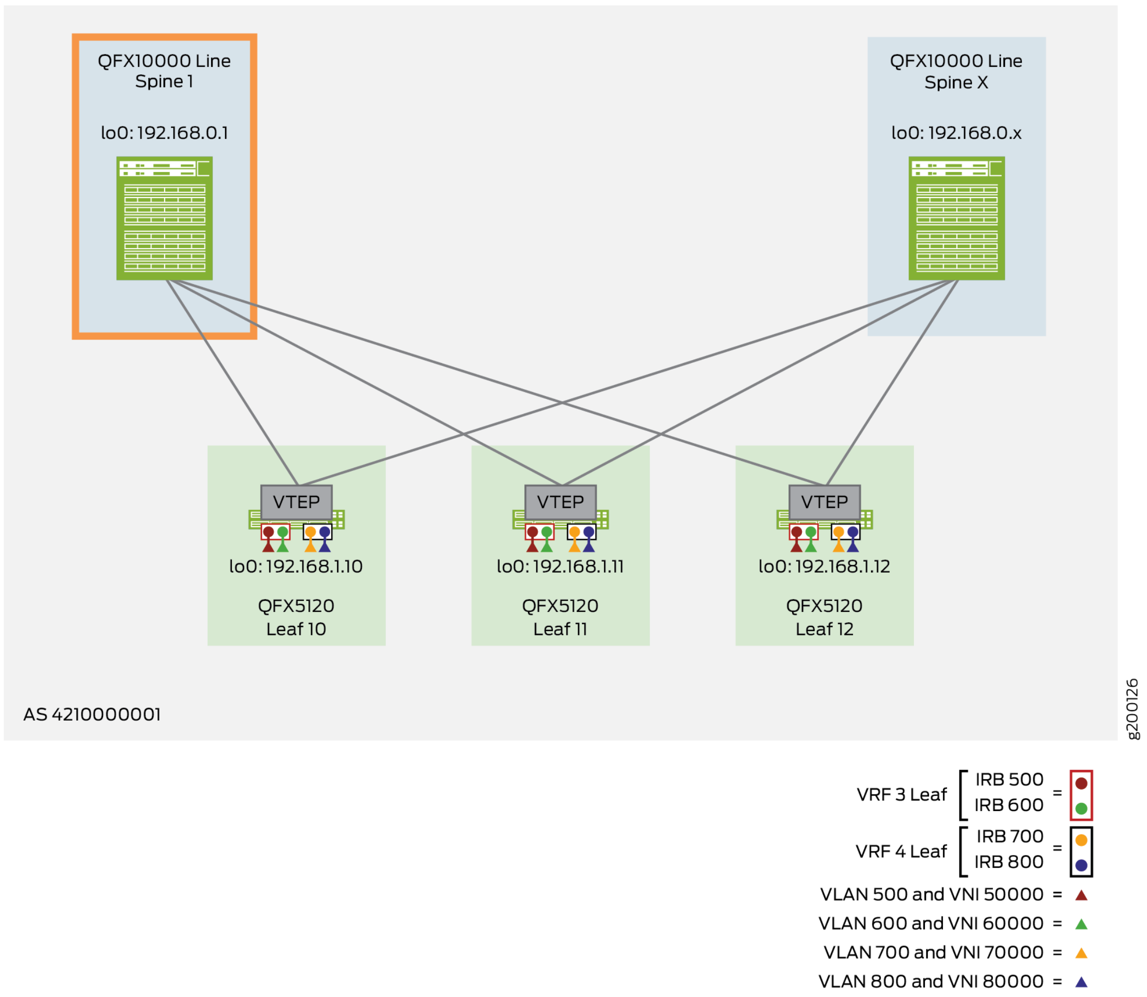
The following sections show the steps to configure and verify the edge-routed bridging overlay:

## Configure an Edge-Routed Bridging Overlay on a Lean Spine Device

To enable the edge-routed bridging overlay on a lean spine device, perform the following:

**NOTE**: The following example shows the configuration for Spine 1, as shown in Figure 53 on page 210.

**Figure 53: Edge-Routed Bridging Overlay – Lean Spine Devices**



1. Ensure the IP fabric underlay is in place. To see the steps required to configure an IP fabric on a spine device, see "IP Fabric Underlay Network Design and Implementation" on page 80.

   If you are using an IPv6 Fabric, see "IPv6 Fabric Underlay and Overlay Network Design and Implementation with EBGP" on page 103 instead. Those instructions include how to configure the IPv6 underlay connectivity with EBGP and IPv6 overlay peering.

2. Confirm that your IBGP overlay is up and running. To configure an IBGP overlay on your spine device, see "Configure IBGP for the Overlay" on page 96.

   If you are using an IPv6 Fabric, you don't need this step. Step 1 also covers how to configure the EBGP IPv6 overlay peering that corresponds to the IPv6 underlay connectivity configuration.

## Verify the Edge-Routed Bridging Overlay on a Lean Spine Device

To verify that IBGP is functional on a lean spine device, use the `show bgp summary` command as described in "Configure IBGP for the Overlay" on page 96. In the output that displays, ensure that the state of the lean spine device and its peers is `Establ` (established).

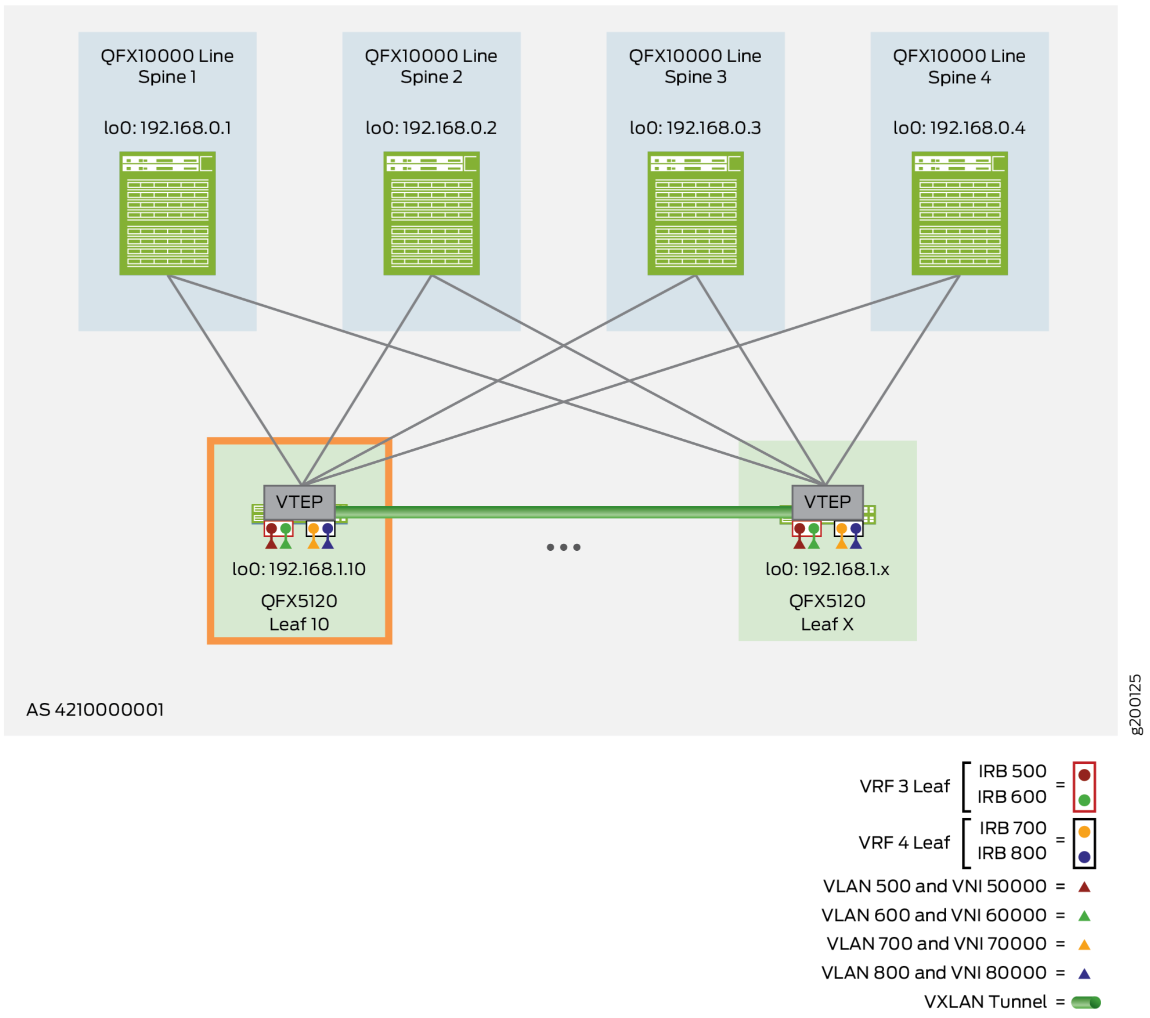
Use the same command if you have an IPv6 Fabric. In the output, look for the IPv6 addresses of the peer device interconnecting interfaces (for underlay EBGP peering) or peer device loopback addresses (for overlay EBGP peering). Ensure that the state is `Establ` (established).

## Configure an Edge-Routed Bridging Overlay on a Leaf Device

To enable the edge-routed bridging overlay on a leaf device, perform the following:

**NOTE**: The following example shows the configuration for Leaf 10, as shown in Figure 54 on page 212.

**Figure 54: Edge-Routed Bridging Overlay – Leaf Devices**



1. Configure the fabric underlay and overlay:

   For an IP Fabric underlay using IPv4:

   - Ensure the IP fabric underlay is in place. To configure an IP fabric on a leaf device, see "IP Fabric Underlay Network Design and Implementation" on page 80.

   - Confirm that your IBGP overlay peering is up and running. To configure IBGP overlay peering on your leaf device, see "Configure IBGP for the Overlay" on page 96.

   For an IPv6 fabric underlay with EBGP IPv6 overlay peering:

   - Ensure the IPv6 underlay is in place and the EBGP overlay peering is up and running. To configure an IPv6 Fabric, see "IPv6 Fabric Underlay and Overlay Network Design and Implementation with EBGP" on page 103.

2. Configure the loopback interface as a VTEP source interface.

   If your configuration uses the default instance, you use statements at the `[edit switch-options]` hierarchy level, as follows:

   *Leaf 10 (Default Instance)*:

   ```
   set switch-options vtep-source-interface lo0.0
   set switch-options route-distinguisher 192.168.1.10:1
   set switch-options vrf-target target:64512:1111
   set switch-options vrf-target auto
   ```

   If your configuration uses MAC-VRF instances, define a routing instance of type `mac-vrf`, and configure these statements at that MAC-VRF routing instance hierarchy level. You also must configure a service type for the MAC-VRF instance. We use the `vlan-aware` service type here so you can associate multiple VLANs with the MAC-VRF instance. This setting is consistent with the alternative configuration that uses the default instance.

   *Leaf 10 (MAC-VRF Instance)*:

   ```
   set routing-instances MAC-VRF-1 vtep-source-interface lo0.0
   set routing-instances MAC-VRF-1 instance-type mac-vrf
   set routing-instances MAC-VRF-1 service-type vlan-aware
   set routing-instances MAC-VRF-1 route-distinguisher 192.168.1.10:1
   set routing-instances MAC-VRF-1  vrf-target target:64512:1111
   ```

   If you have an IPv6 Fabric (supported only with MAC-VRF instances), in this step you include the `inet6` option when you configure the VTEP source interface to use the device loopback address. This option enables IPv6 VXLAN tunneling in the fabric. This is the only difference in the MAC-VRF configuration with an IPv6 Fabric as compared to the MAC-VRF configuration with an IPv4 Fabric.

   *Leaf 10 (MAC-VRF Instance with an IPv6 Fabric)*:

   ```
   set routing-instances MAC-VRF-1 vtep-source-interface lo0.0 inet6
   set routing-instances MAC-VRF-1 instance-type mac-vrf
   set routing-instances MAC-VRF-1 service-type vlan-aware
   set routing-instances MAC-VRF-1 route-distinguisher 192.168.1.10:1
   set routing-instances MAC-VRF-1  vrf-target target:64512:1111
   ```

3. (MAC-VRF instances only) Enable shared tunnels on devices in the QFX5000 line running Junos OS.

A device can have problems with VTEP scaling when the configuration uses multiple MAC-VRF instances. As a result, to avoid this problem, we require that you enable the shared tunnels feature on the QFX5000 line of switches running Junos OS with a MAC-VRF instance configuration. When you configure the shared-tunnels option, the device minimizes the number of next-hop entries to reach remote VTEPs. This statement is optional on the QFX10000 line of switches running Junos OS because those devices can handle higher VTEP scaling than QFX5000 switches. You also don't need to configure this option on devices running Junos OS Evolved, where shared tunnels are enabled by default.

Include the following statement to globally enable shared VXLAN tunnels on the device:

```
set forwarding-options evpn-vxlan shared-tunnels
```

**NOTE**: This setting requires you to reboot the device.

4. (Required on PTX10000 Series routers only) Enable tunnel termination globally (in other words, on all interfaces) on the device:

```
set forwarding-options tunnel-termination
```

5. Configure the leaf-to-end system aggregated Ethernet interface as a trunk carrying four VLANs. Include the appropriate ESI and LACP values for your topology.

   *Leaf 10*:

```
set interfaces ae11 esi 00:00:00:00:00:01:00:00:00:01
set interfaces ae11 esi all-active
set interfaces ae11 aggregated-ether-options lacp active
set interfaces ae11 aggregated-ether-options lacp periodic fast
set interfaces ae11 aggregated-ether-options lacp system-id 00:01:00:00:00:01
set interfaces ae11 unit 0 family ethernet-switching interface-mode trunk
set interfaces ae11 unit 0 family ethernet-switching vlan members VNI_50000
set interfaces ae11 unit 0 family ethernet-switching vlan members VNI_60000
set interfaces ae11 unit 0 family ethernet-switching vlan members VNI_70000
set interfaces ae11 unit 0 family ethernet-switching vlan members VNI_80000
```

**NOTE**: When configuring ESI-LAGs on the QFX5000 line of switches that serve as leaf devices in an edge-routed bridging overlay, keep in mind that we currently support only the Enterprise style of interface configuration, which is shown in this step.

If your configuration uses MAC-VRF instances, you must also add the configured aggregated Ethernet interface to the MAC-VRF instance:

```
set routing-instances MAC-VRF-1 interface ae11.0
```

6. Configure the mapping of VLANs to VNIs and associate one IRB interface per VLAN.

   This step shows the VLAN to VNI mapping and IRB interface association either in the default instance or in a MAC-VRF instance configuration.

   *Leaf 10 (Default Instance)*:

```
set vlans VNI_50000 vlan-id 500
set vlans VNI_50000 l3-interface irb.500
set vlans VNI_50000 vxlan vni 50000
set vlans VNI_60000 vlan-id 600
set vlans VNI_60000 l3-interface irb.600
set vlans VNI_60000 vxlan vni 60000
set vlans VNI_70000 vlan-id 700
set vlans VNI_70000 l3-interface irb.700
set vlans VNI_70000 vxlan vni 70000
set vlans VNI_80000 vlan-id 800
set vlans VNI_80000 l3-interface irb.800
set vlans VNI_80000 vxlan vni 80000
```

   *Leaf 10 (MAC-VRF Instance)*:

   The only difference with a MAC-VRF instance configuration is that you configure these statements in the MAC-VRF instance at the `[edit routing-instances` *`mac-vrf-instance-name`*`]` hierarchy level.

```
set routing-instances MAC-VRF-1 vlans VNI_50000 vlan-id 500
set routing-instances MAC-VRF-1 vlans VNI_50000 l3-interface irb.500
set routing-instances MAC-VRF-1 vlans VNI_50000 vxlan vni 50000
set routing-instances MAC-VRF-1 vlans VNI_60000 vlan-id 600
set routing-instances MAC-VRF-1  vlans VNI_60000 l3-interface irb.600
set routing-instances MAC-VRF-1 vlans VNI_60000 vxlan vni 60000
set routing-instances MAC-VRF-1 vlans VNI_70000 vlan-id 700
set routing-instances MAC-VRF-1 vlans VNI_70000 l3-interface irb.700
set routing-instances MAC-VRF-1 vlans VNI_70000 vxlan vni 70000
set routing-instances MAC-VRF-1 vlans VNI_80000 vlan-id 800
set routing-instances MAC-VRF-1 vlans VNI_80000 l3-interface irb.800
set routing-instances MAC-VRF-1 vlans VNI_80000 vxlan vni 80000
```

7. Configure the IRB interfaces for VNIs 50000 and 60000 with both IPv4 and IPv6 dual stack addresses for both the IRB IP address and virtual gateway IP address.

   There are two methods for configuring gateways for IRB interfaces:

   - Method 1: Unique IRB IP Address with Virtual Gateway IP Address, which is shown in Step 7.
   - Method 2: IRB with Anycast IP Address and MAC Address, which is shown in Step 8.

   *Leaf 10*:

```
set interfaces irb unit 500 family inet address 10.1.4.1/24 virtual-gateway-address
10.1.4.254
set interfaces irb unit 500 family inet6 address 2001:db8::10:1:4:1/112 virtual-gateway-
address 2001:db8::10:1:4:254
set interfaces irb unit 500 family inet6 address fe80:10:1:4::1/64 virtual-gateway-address
fe80:10:1:4::254
set interfaces irb unit 600 family inet address 10.1.5.1/24 virtual-gateway-address
10.1.5.254
set interfaces irb unit 600 family inet6 address 2001:db8::10:1:5:1/112 virtual-gateway-
address 2001:db8::10:1:5:254
set interfaces irb unit 600 family inet6 address fe80:10:1:5::1/64 virtual-gateway-address
fe80:10:1:5::254
set interfaces irb unit 500 virtual-gateway-v4-mac 00:00:5e:00:00:04
set interfaces irb unit 500 virtual-gateway-v6-mac 00:00:5e:00:00:04
set interfaces irb unit 600 virtual-gateway-v4-mac 00:00:5e:00:00:04
set interfaces irb unit 600 virtual-gateway-v6-mac 00:00:5e:00:00:04
```

8. Configure the IRB interfaces for VNIs 70000 and 80000 with a dual stack Anycast IP address.

   *Leaf 10*:

```
set interfaces irb unit 700 family inet address 10.1.6.254/24
set interfaces irb unit 700 family inet6 address 2001:db8::10:1:6:254/112
set interfaces irb unit 700 family inet6 address fe80::10:1:6:254/112
set interfaces irb unit 700 mac 0a:fe:00:00:00:01
set interfaces irb unit 800 family inet address 10.1.7.254/24
set interfaces irb unit 800 family inet6 address 2001:db8::10:1:7:254/112
set interfaces irb unit 800 family inet6 address fe80::10:1:7:254/112
set interfaces irb unit 800 mac 0a:fe:00:00:00:02
```

For more information about IRB and virtual gateway IP address configuration, see the *IRB Addressing Models in Bridging Overlays* section in "Data Center Fabric Blueprint Architecture Components" on page 9.

9. Enable the ping operation for IRB interfaces 500 and 600, which are configured in Step 6.

```
set interfaces irb unit 500 family inet address 10.1.4.1/24 preferred
set interfaces irb unit 500 family inet6 address 2001:db8::10:1:4:1/112 preferred
set interfaces irb unit 500 virtual-gateway-accept-data
set interfaces irb unit 600 family inet address 10.1.5.1/24 preferred
set interfaces irb unit 600 family inet6 address 2001:db8::10:1:5:1/112 preferred
set interfaces irb unit 600 virtual-gateway-accept-data
```

10. Configure the EVPN protocol with VXLAN encapsulation on the leaf device.

This step shows how to configure either the default instance or a MAC-VRF instance.

*Leaf 10 (Default Instance)*:

```
set protocols evpn encapsulation vxlan
set protocols evpn default-gateway no-gateway-community
set protocols evpn extended-vni-list all
```

*Leaf 10 (MAC-VRF Instance)*:

The only difference with a MAC-VRF configuration is that you configure these statements in the MAC-VRF instance at the `[edit routing-instances` *`mac-vrf-instance-name`*`]` hierarchy level.

```
set routing-instances MAC-VRF-1 protocols evpn encapsulation vxlan
set routing-instances MAC-VRF-1 protocols evpn default-gateway no-gateway-community
set routing-instances MAC-VRF-1 protocols evpn extended-vni-list all
```

11. Configure a policy called EXPORT_HOST_ROUTES to match on and accept /32 and /128 host routes, direct routes, and static routes. You will use this policy in step 13.

```
set policy-options policy-statement EXPORT_HOST_ROUTES term TERM_1 from protocol evpn
set policy-options policy-statement EXPORT_HOST_ROUTES term TERM_1 from route-filter
0.0.0.0/0 prefix-length-range /32-/32
set policy-options policy-statement EXPORT_HOST_ROUTES term TERM_1 then accept
set policy-options policy-statement EXPORT_HOST_ROUTES term TERM_2 from protocol direct
set policy-options policy-statement EXPORT_HOST_ROUTES term TERM_2 then accept
set policy-options policy-statement EXPORT_HOST_ROUTES term TERM_3 from protocol static
set policy-options policy-statement EXPORT_HOST_ROUTES term TERM_3 then accept
set policy-options policy-statement EXPORT_HOST_ROUTES TERM_4 from family inet6
set policy-options policy-statement EXPORT_HOST_ROUTES TERM_4 from protocol evpn
set policy-options policy-statement EXPORT_HOST_ROUTES TERM_4 from route-filter 0::0/0
prefix-length-range /128-/128
set policy-options policy-statement EXPORT_HOST_ROUTES TERM_4 then accept
```

12. Configure the loopback interface with two logical interfaces. (You will assign one logical interface to each VRF routing instance in the next step).

```
set interfaces lo0 unit 3 family inet
set interfaces lo0 unit 4 family inet
```

13. Configure two tenant VRF routing instances, one for VNIs 50000 and 60000 (VRF 3), and one for VNIs 70000 and 80000 (VRF 4). Assign one logical interface from the loopback to each routing instance so that the VXLAN gateway can resolve ARP requests. Configure IP prefix route properties for EVPN Type 5 to advertise ARP routes to the spine devices. Enable load balancing for L3 VPNs (set the `multipath` option).

    *Leaf 10*:

```
set routing-instances VRF_3 instance-type vrf
set routing-instances VRF_3 interface irb.500
set routing-instances VRF_3 interface irb.600
set routing-instances VRF_3 interface lo0.3
set routing-instances VRF_3 route-distinguisher 192.168.1.10:500
set routing-instances VRF_3 vrf-target target:62273:50000
set routing-instances VRF_3 protocols evpn ip-prefix-routes advertise direct-nexthop
set routing-instances VRF_3 protocols evpn ip-prefix-routes encapsulation vxlan
set routing-instances VRF_3 protocols evpn ip-prefix-routes vni 16777214
set routing-instances VRF_3 protocols evpn ip-prefix-routes export EXPORT_HOST_ROUTES
set routing-instances VRF-3 routing-options multipath
set routing-instances VRF-3 routing-options rib VRF-3.inet6.0 multipath
set routing-instances VRF_4 instance-type vrf
set routing-instances VRF_4 interface irb.700
set routing-instances VRF_4 interface irb.800
set routing-instances VRF_4 interface lo0.4
set routing-instances VRF_4 route-distinguisher 192.168.1.10:600
set routing-instances VRF_4 vrf-target target:62273:60000
set routing-instances VRF_4 protocols evpn ip-prefix-routes advertise direct-nexthop
set routing-instances VRF_4 protocols evpn ip-prefix-routes encapsulation vxlan
set routing-instances VRF_4 protocols evpn ip-prefix-routes vni 16777105
set routing-instances VRF_4 protocols evpn ip-prefix-routes export EXPORT_HOST_ROUTES
set routing-instances VRF_4 routing-options multipath
set routing-instances VRF_4 routing-options rib VRF_4.inet6.0 multipath
```

14. (Required on ACX7100 routers only) Set the `reject-asymmetric-vni` option in the VRF routing instances where you enable Type 5 IP prefix routes. This option configures the device to reject EVPN Type 5 route advertisements with asymmetric VNIs—the device doesn't accept traffic from the control plane with a received VNI that doesn't match the locally configured VNI. We support only symmetric VNI routes on these devices.

```
set routing-instances VRF_3 protocols evpn ip-prefix-routes reject-asymmetric-vni
set routing-instances VRF_4 protocols evpn ip-prefix-routes reject-asymmetric-vni
```

15. Set up dummy IPv4 and IPv6 static routes for each tenant VRF instance, which enable the device to advertise at least one EVPN Type 5 route for the VRF. We include this step because all devices with Type 5 routes configured must advertise at least one route for forwarding to work. The device must receive at least one EVPN Type 5 route from a peer device and install an IP forwarding route in the Packet Forwarding Engine. Otherwise, the device doesn't install the next hop required to de-encapsulate the received traffic and doesn't forward the traffic.

    *Leaf 10*:

```
set routing-instances VRF_3 routing-options static route 10.10.20.30/32 discard
set routing-instances VRF-3 routing-options rib VRF-3.inet6.0 static route
2001:db8::101:1:1:1/128 discard
set routing-instances VRF_4 routing-options static route 10.10.20.30/32 discard
set routing-instances VRF-4 routing-options rib VRF-4.inet6.0 static route
2001:db8::101:1:1:1/128 discard
```

16. If you are configuring a QFX5110, QFX5120-48Y, or QFX5120-32C switch, you must perform this step to support pure EVPN Type 5 routes on ingress EVPN traffic.

```
set routing-options forwarding-table chained-composite-next-hop ingress evpn
set forwarding-options vxlan-routing overlay-ecmp
```

**NOTE**: Entering the **overlay-ecmp** statement causes the Packet Forwarding Engine to restart, which interrupts forwarding operations. We recommend using this configuration statement before the EVPN-VXLAN network becomes operational.

17. If you are configuring a QFX5110, QFX5120-48Y, or QFX5120-32C switch, and you expect that there will be more than 8000 ARP table entries and IPv6 neighbor entries, perform this step.

Configure the maximum number of next hops reserved for use in the EVPN-VXLAN overlay network. By default, the switch allocates 8000 next hops for use in the overlay network. See *next-hop* for more details.

**NOTE**: Changing the number of next hops causes the Packet Forwarding Engine to restart, which interrupts forwarding operations. We recommend using this configuration statement before the EVPN-VXLAN network becomes operational.

```
set forwarding-options vxlan-routing next-hop 12288
```

## Verify the Edge-Routed Bridging Overlay on a Leaf Device

To verify that the edge-routed bridging overlay is working, run the following commands.

The commands here show output for a default instance configuration. With a MAC-VRF instance configuration, you can alternatively use:

- `show mac-vrf forwarding` commands that are aliases for the `show ethernet-switching` commands in this section.
- The `show mac-vrf routing database` command, which is an alias for the `show evpn database` command in this section.

The output with a MAC-VRF instance configuration displays similar information for MAC-VRF routing instances as this section shows for the default instance. One main difference you might see is in the output with MAC-VRF instances on devices where you enable the shared tunnels feature. With shared tunnels enabled, you see VTEP interfaces in the following format:

```
vtep-index.shared-tunnel-unit
```

where:

- *index* is the index associated with the MAC-VRF routing instance.
- *shared-tunnel-unit* is the unit number associated with the shared tunnel remote VTEP logical interface.

For example, if a device has a MAC-VRF instance with index 26 and the instance connects to two remote VTEPs, the shared tunnel VTEP logical interfaces might look like this:

```
vtep-26.32823
vtep-26.32824
```

If your configuration uses an IPv6 Fabric, you provide IPv6 address parameters where applicable. Output from the commands that display IP addresses reflect the IPv6 device and interface addresses from the underlying fabric. See "IPv6 Fabric Underlay and Overlay Network Design and Implementation with EBGP" on page 103 for the fabric parameters reflected in command outputs in this section with an IPv6 Fabric.

1. Verify that the aggregated Ethernet interface is operational.

   ```
   user@leaf-10> show interfaces terse ae11
   Interface               Admin Link Proto    Local                 Remote
   ae11                    up    up
   ae11.0                  up    up   eth-switch
   ```

2. Verify the VLAN information (associated ESIs, VTEPs, etc.).

   ```
   user@leaf-10> show vlans
   default-switch          VNI_50000            500
                                                             ae11.0*
                                                             esi.7585*
                                                             esi.7587*
                                                             esi.8133*
                                                             et-0/0/16.0
                                                             vtep.32782*
                                                             vtep.32785*
                                                             vtep.32802*
                                                             vtep.32806*
                                                             vtep.32826*

   ...

   default-switch          VNI_80000            800
                                                             ae11.0*
                                                             esi.7585*
                                                             esi.8133*
                                                             et-0/0/16.0
   ```

```
                                                          vtep.32782*
                                                          vtep.32785*
                                                          vtep.32802*
                                                          vtep.32806*
                                                          vtep.32826*
```

**Note:** esi.7585 is the ESI of the remote aggregated Ethernet link for Leaf 4, Leaf 5, and Leaf 6.

```
user@leaf-10> show ethernet-switching vxlan-tunnel-end-point esi | find esi.7585
00:00:00:00:00:00:51:10:00:01 default-switch          7585  2097663 esi.7585           3
    RVTEP-IP             RVTEP-IFL      VENH     MASK-ID   FLAGS
    192.168.1.5          vtep.32826     8169     2         2
    192.168.1.6          vtep.32782     7570     1         2
    192.168.1.4          vtep.32785     7575     0         2
```

**Note:** esi.7587 is the ESI for all leaf devices that have the same VNI number (Leaf 4, Leaf 5, Leaf 6, Leaf 11, and Leaf 12).

```
user@leaf-10> show ethernet-switching vxlan-tunnel-end-point esi | find esi.7587
05:19:17:f3:41:00:00:c3:50:00 default-switch          7587  2097665 esi.7587           5
    RVTEP-IP             RVTEP-IFL      VENH     MASK-ID   FLAGS
    192.168.1.5          vtep.32826     8169     4         2
    192.168.1.6          vtep.32782     7570     3         2
    192.168.1.12         vtep.32802     8127     2         2
    192.168.1.11         vtep.32806     8131     1         2
    192.168.1.4          vtep.32785     7575     0         2
```

**Note:** esi.8133 is the ESI for the local aggregated Ethernet interface shared with Leaf 11 and Leaf 12.

```
user@leaf-10> show ethernet-switching vxlan-tunnel-end-point esi | find esi.8133
00:00:00:00:00:01:00:00:00:01 default-switch          8133  2098194 esi.8133  ae11.0,   2
    RVTEP-IP             RVTEP-IFL      VENH     MASK-ID   FLAGS
    192.168.1.12         vtep.32802     8127     1         2
    192.168.1.11         vtep.32806     8131     0         2
```

3. Verify the ARP table.

**Note:** 10.1.4.201 and 10.1.5.201 are remote end systems connected to the QFX5110 switches; and 10.1.4.202 and 10.1.5.202 are local end systems connected to Leaf 10 through interface ae11.

```
user@leaf-10> show arp no-resolve vpn VRF_3
MAC Address       Address         Interface         Flags
06:a7:39:9a:7b:c0 10.1.4.4        irb.500 [vtep.32785]     none
06:a7:39:f8:b1:00 10.1.4.5        irb.500 [vtep.32826]     none
06:e0:f3:1b:09:80 10.1.4.6        irb.500 [vtep.32782]     none
02:0c:10:04:02:01 10.1.4.201      irb.500 [.local..9]      none
02:0c:10:04:02:02 10.1.4.202      irb.500 [ae11.0]         none
00:00:5e:00:00:04 10.1.4.254      irb.500                  permanent published gateway
06:a7:39:9a:7b:c0 10.1.5.4        irb.600 [vtep.32785]     none
06:a7:39:f8:b1:00 10.1.5.5        irb.600 [vtep.32826]     none
06:e0:f3:1b:09:80 10.1.5.6        irb.600 [vtep.32782]     none
02:0c:10:05:02:01 10.1.5.201      irb.600 [.local..9]      none
02:0c:10:05:02:02 10.1.5.202      irb.600 [ae11.0]         none
00:00:5e:00:00:04 10.1.5.254      irb.600                  permanent published gateway
Total entries: 12
user@leaf-10> show arp no-resolve vpn VRF_4
MAC Address       Address         Interface         Flags
02:0c:10:06:02:01 10.1.6.201      irb.700 [.local..9]      permanent remote
02:0c:10:06:02:02 10.1.6.202      irb.700 [ae11.0]         none
02:0c:10:07:02:01 10.1.7.201      irb.800 [.local..9]      permanent remote
02:0c:10:07:02:02 10.1.7.202      irb.800 [ae11.0]         permanent remote
Total entries: 4
user@leaf-10> show ipv6 neighbors
IPv6 Address                 Linklayer Address  State       Exp Rtr Secure Interface
2001:db8::10:1:4:2           06:ac:ac:23:0c:4e  stale       523 yes no      irb.500
[vtep.32806]
2001:db8::10:1:4:3           06:ac:ac:24:18:7e  stale       578 yes no      irb.500
[vtep.32802]
2001:db8::10:1:4:201         02:0c:10:04:02:01  stale       1122 no no      irb.500
[.local..9]
2001:db8::10:1:4:202         02:0c:10:04:02:02  stale       1028 no no      irb.500 [ae11.0]
2001:db8::10:1:4:254         00:00:5e:00:00:04   reachable   0   no  no      irb.500
2001:db8::10:1:5:2           06:ac:ac:23:0c:4e  stale       521 yes no      irb.600
[vtep.32806]
2001:db8::10:1:5:3           06:ac:ac:24:18:7e  stale       547 yes no      irb.600
[vtep.32802]
2001:db8::10:1:5:201         02:0c:10:05:02:01  stale       508 no  no      irb.600
[.local..9]
2001:db8::10:1:5:202         02:0c:10:05:02:02  stale       1065 no no      irb.600 [ae11.0]
```

```
2001:db8::10:1:5:254        00:00:5e:00:00:04   reachable   0   no  no      irb.600
2001:db8::10:1:6:202        02:0c:10:06:02:02  reachable   0   no  no      irb.700 [ae11.0]
2001:db8::10:1:7:201        02:0c:10:07:02:01  stale       647 no  no      irb.800
[.local..9]
```

4. Verify the MAC addresses and ARP information in the EVPN database.

   For example, with an IPv4 Fabric:

```
user@leaf-10> show evpn database mac-address 02:0c:10:04:02:01 extensive
Instance: default-switch

VN Identifier: 50000, MAC address:: 02:0c:10:04:02:01
  Source: 00:00:00:00:00:00:51:10:00:01, Rank: 1, Status: Active
    Remote origin: 192.168.1.4
    Remote origin: 192.168.1.5
    Timestamp: Oct 10 16:09:54 (0x59dd5342)
    State: <Remote-To-Local-Adv-Done>
    IP address: 10.1.4.201
      Remote origin: 192.168.1.4
      Remote origin: 192.168.1.5
    IP address: 2001:db8::10:1:4:201
      Flags: <Proxy>
      Remote origin: 192.168.1.4
      Remote origin: 192.168.1.5    History db:
      Time                  Event
      Oct 10 16:09:55 2017  Advertisement route cannot be created (no local state present)
      Oct 10 16:09:55 2017  Updating output state (change flags 0x0)
      Oct 10 16:09:55 2017  Advertisement route cannot be created (no local source present)
      Oct 10 16:09:55 2017  IP host route cannot be created (No remote host route for non-
MPLS instance)
      Oct 10 16:09:55 2017  Updating output state (change flags 0x4000 <IP-Peer-Added>)
      Oct 10 16:09:55 2017  Creating MAC+IP advertisement route for proxy
      Oct 10 16:09:55 2017  Creating MAC+IP advertisement route for proxy
      Oct 10 16:09:55 2017  IP host route cannot be created (No remote host route for non-
MPLS instance)
      Oct 10 16:09:55 2017  Clearing change flags <IP-Added>
      Oct 10 16:09:55 2017  Clearing change flags <IP-Peer-Added>
user@leaf-10> show evpn database mac-address 02:0c:10:04:02:02 extensive
Instance: default-switch

VN Identifier: 50000, MAC address:: 02:0c:10:04:02:02
  Source: 00:00:00:00:00:01:00:00:00:01, Rank: 1, Status: Active
```

```
      Remote origin: 192.168.1.11
      Remote origin: 192.168.1.12
      Timestamp: Oct 10 16:14:32 (0x59dd5458)
      State: <Remote-To-Local-Adv-Done>
      IP address: 10.1.4.202
        Remote origin: 192.168.1.11
        Remote origin: 192.168.1.12
        L3 route: 10.1.4.202/32, L3 context: VRF_3 (irb.500)
      IP address: 2001:db8::10:1:4:202
        Remote origin: 192.168.1.11
        L3 route: 2001:db8::10:1:4:202/128, L3 context: VRF_3 (irb.500)    History db: 
        Time                    Event
        Oct 10 16:14:32 2017  Advertisement route cannot be created (no local state present)
        Oct 10 16:14:32 2017  Sent MAC add with NH 0, interface ae11.0 (index 0), RTT 9, remote 
addr 192.168.1.12, ESI 0100000001, VLAN 0, VNI 50000, flags 0x0, timestamp 0x59dd5458 to L2ALD
        Oct 10 16:14:32 2017  Sent peer 192.168.1.12 record created
        Oct 10 16:14:32 2017  Sent MAC+IP add, interface <none>, RTT 9, IP 10.1.4.202 remote 
peer 192.168.1.12, ESI 0100000001, VLAN 0, VNI 50000, flags 0x80, timestamp 0x59dd5458 to 
L2ALD
        Oct 10 16:14:32 2017  Updating output state (change flags 0x4000 <IP-Peer-Added>)
        Oct 10 16:14:32 2017  Advertisement route cannot be created (no local source present)
        Oct 10 16:14:32 2017  Updating output state (change flags 0x0)
        Oct 10 16:14:32 2017  Advertisement route cannot be created (no local source present)
        Oct 10 16:14:32 2017  Clearing change flags <ESI-Peer-Added>
        Oct 10 16:14:32 2017  Clearing change flags <IP-Peer-Added>
```

Or for example, with an IPv6 Fabric:

```
user@leaf-1> show evpn database mac-address c8:fe:6a:e4:2e:00 l2-domain-id 1000
Instance: MAC-VRF-1
VLAN  DomainId  MAC address        Active source             Timestamp         IP address
      1000       c8:fe:6a:e4:2e:00  2001:db8::192:168:1:2     Jan 19 17:36:46   77.5.241.242

2001:db8::77:0:5f1:242

fe80::cafe:6a05:f1e4:2e00

user@leaf-1> show evpn database mac-address c8:fe:6a:e4:2e:00 l2-domain-id 1000 extensive
Instance: MAC-VRF-1

VN Identifier: 1000, MAC address: c8:fe:6a:e4:2e:00
  State: 0x0
  Source: 2001:db8::192:168:1:2, Rank: 1, Status: Active
    Mobility sequence number: 0 (minimum origin address 2001:db8::192:168:1:2)
    Timestamp: Jan 19 17:36:46.033965 (0x61e8bcae)
    State: <Remote-To-Local-Adv-Done Remote-Pinned>
    MAC advertisement route status: Not created (no local state present)
    IP address: 77.5.241.242
    IP address: 2001:db8::77:0:5f1:242
    IP address: fe80::cafe:6a05:f1e4:2e00
    History db:
      Time                        Event
      Jan 19 16:15:00.440 2022   2001:db8::192:168:1:2 : Remote peer 2001:db8::192:168:1:2 
created
      Jan 19 16:15:00.440 2022   2001:db8::192:168:1:2 : Created
      Jan 19 16:15:00.447 2022   Updating output state (change flags 0x1 <ESI-Added>)
      Jan 19 16:15:00.447 2022   Active ESI changing (not assigned -> 2001:db8::192:168:1:2)
      Jan 19 16:15:00.447 2022   2001:db8::192:168:1:2 : 77.5.241.242 Selected IRB interface 
nexthop
      Jan 19 16:15:00.447 2022   2001:db8::192:168:1:2 : 77.5.241.242 Reject remote ip host 
route 77.5.241.242 in L3 context VRF-1 since no remote-ip-host-routes configured
      Jan 19 16:15:00.447 2022   2001:db8::192:168:1:2 : 2001:db8::77:0:5f1:242 Selected IRB 
interface nexthop
      Jan 19 16:15:00.447 2022   2001:db8::192:168:1:2 : 2001:db8::77:0:5f1:242 Reject remote 
ip host route 2001:db8::77:0:5f1:242 in L3 context VRF-1 since no remote-ip-host-routes 
configured
      Jan 19 16:15:00.447 2022   2001:db8::192:168:1:2 : fe80::cafe:6a05:f1e4:2e00 Selected 
IRB interface nexthop
      Jan 19 16:15:00.447 2022   2001:db8::192:168:1:2 : fe80::cafe:6a05:f1e4:2e00 Reject 
remote ip host route fe80::cafe:6a05:f1e4:2e00 in L3 context VRF-1 since no remote-ip-host-
routes configured
```

5. Verify the IPv4 and IPv6 end system routes appear in the forwarding table.

```
user@leaf-10> show route forwarding-table table VRF_1 destination 10.1.4.202 extensive
Routing table: VRF_3.inet [Index 5]
Internet:
Enabled protocols: Bridging, All VLANs,

Destination:  10.1.4.202/32
  Route type: destination
  Route reference: 0                   Route interface-index: 563
  Multicast RPF nh index: 0
  P2mpidx: 0
```

```
    Flags: sent to PFE
    Nexthop: b0:c:10:4:2:2
    Next-hop type: unicast              Index: 10210    Reference: 1
    Next-hop interface: ae11.0
  user@leaf-10> show route forwarding-table table VRF_1 destination 2001:db8::10:1:4:202
  extensive
  Routing table: VRF_3.inet6 [Index 5]
  Internet6:
  Enabled protocols: Bridging, All VLANs,

  Destination:  2001:db8::10:1:4:202/128
    Route type: destination
    Route reference: 0                  Route interface-index: 563
    Multicast RPF nh index: 0
    P2mpidx: 0
    Flags: sent to PFE
    Nexthop: b0:c:10:4:2:2
    Next-hop type: unicast              Index: 10244    Reference: 1
    Next-hop interface: ae11.0
```

SEE ALSO

MAC-VRF Routing Instance Type Overview

*Example: Configuring an EVPN-VXLAN Edge-Routed Bridging Overlay Within a Data Center*

*Example: Configuring a QFX5110 Switch as Layer 2 and 3 VXLAN Gateways in an EVPN-VXLAN Edge-Routed Bridging Overlay*

## Configure Symmetric IRB Routing with EVPN Type 2 Routes on Leaf Devices

In EVPN-VXLAN ERB overlay networks, by default, leaf devices use an asymmetric IRB model with EVPN Type 2 routes to send traffic between subnets across the VXLAN tunnels, where:

- L3 routing for inter-subnet traffic happens at the ingress device. Then the source VTEP forwards traffic at L2 toward the destination VTEP.
- The traffic arrives at the destination VTEP, and that VTEP forwards the traffic on the destination VLAN.

- For this model to work, you must configure all source and destination VLANs and their corresponding VNIs on all leaf devices.

You can alternatively enable symmetric IRB routing with Type 2 routes (called *symmetric Type 2 routing* here for brevity). We support symmetric Type 2 routing only in ERB overlay fabrics.

To use the symmetric IRB model to route traffic for a VRF, you must enable it in that VRF on all the leaf devices in the fabric. However, you don't need to configure all VLANs and VNIs in the VRF on all leaf devices. On each leaf device, you can configure only the host VLANs that leaf device serves. As a result, using symmetric Type 2 routing helps with scaling when your EVPN network has a large number of VLANs and many attached hosts or servers. We highlight this benefit here when we show how to enable symmetric Type 2 routing in the ERB overlay reference architecture in this chapter.

With the symmetric Type 2 routing model, the VTEPs route between subnets in a tenant VRF instance using the same VNI in either direction. Symmetric Type 2 routing for a VRF shares the L3 VNI tunnel you configure for EVPN Type 5 routing in the VRF. The implementation requires you to also configure EVPN Type 5 routing in the VRF. See *Symmetric Integrated Routing and Bridging with EVPN Type 2 Routes in EVPN-VXLAN Fabrics* for more details on asymmetric and symmetric IRB routing models, and how symmetric Type 2 routing works.

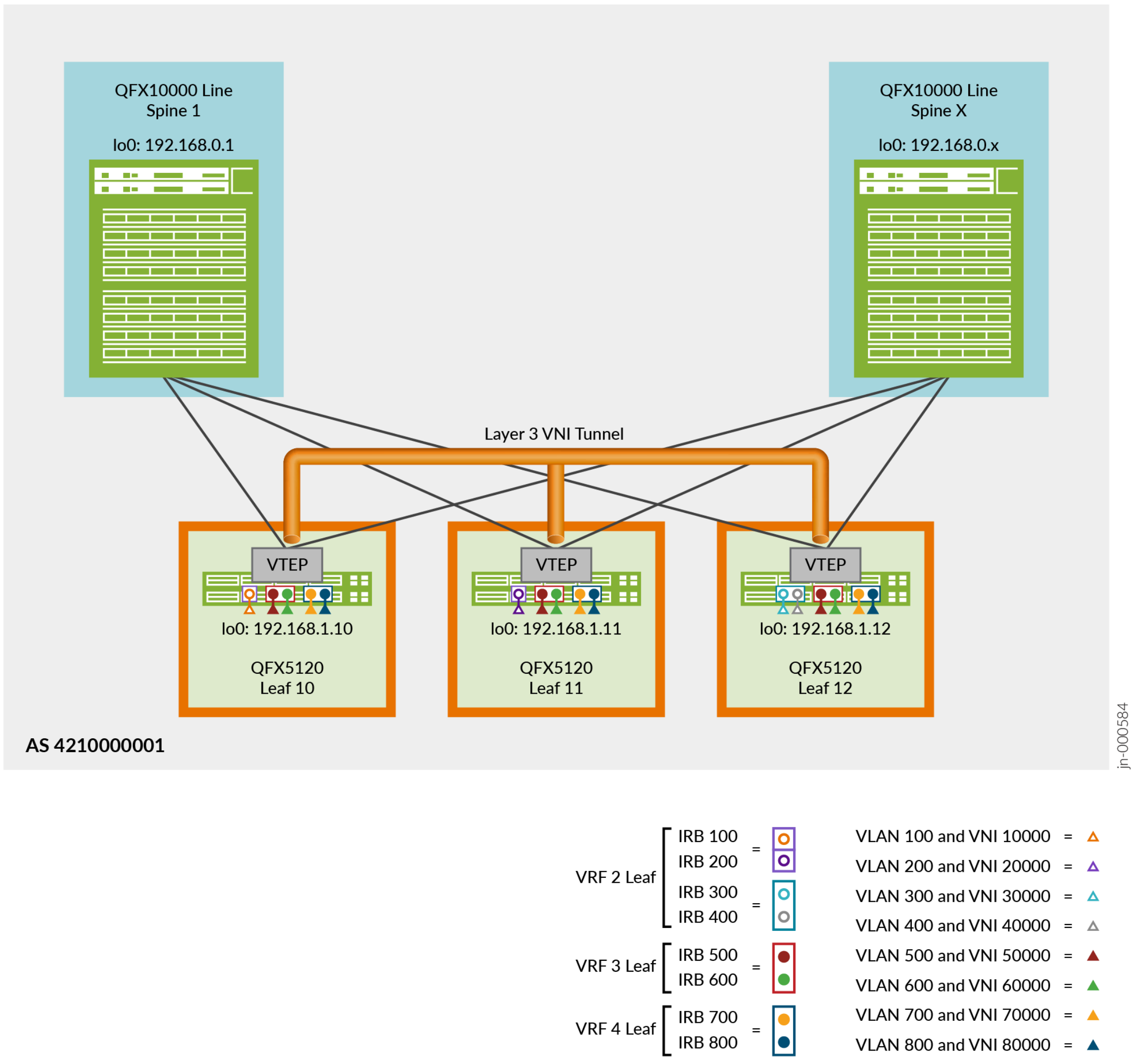
In this section, you update the configuration for the EVPN instance MAC-VRF-1 from "Configure an Edge-Routed Bridging Overlay on a Leaf Device" on page 211 to include four more VLANs, VNI mappings, and corresponding IRB interfaces. You include the IRB interfaces in another tenant VRF called VRF_2 as follows:

- Leaf 10—IRB 100
- Leaf 11—IRB 200
- Leaf 12—IRB 300 and IRB 400

See Figure 55 on page 229.

The VNI for the L3 VNI tunnel in the figure is the VNI you configure for EVPN Type 5 routing in VRF_2. We enable symmetric Type 2 routing with the same VNI in VRF_2. Again, note that with symmetric Type 2 routing, you don't need to configure all VLANs in the VRF on all leaf devices.

**Figure 55: Edge-Routed Bridging Overlay—Leaf Devices with Symmetric IRB Type 2 Routing**



You must configure the EVPN instance using instance type MAC-VRF on the leaf devices in the ERB overlay fabric according to the instructions in "Configure an Edge-Routed Bridging Overlay on a Leaf Device" on page 211. Use a different route distinguisher on each leaf device—in this configuration, the route distinguishers mirror the device lo0.0 loopback addresses on the device. This configuration includes the same MAC-VRF instance and the same instance `vrf-target` value, but these don't need to be the same on all the leaf devices for symmetric Type 2 routing to work.

To enable symmetric Type 2 routing on the leaf devices according to Figure 55 on page 229, perform these additional configuration steps as indicated in each step for Leaf 10, Leaf 11, and Leaf 12.

1. Configure the aggregated Ethernet interface ae11 trunk interface to carry the additional VLANs 100, 200, 300, and 400 (named VNI_10000, VNI_20000, VNI_30000, and VNI_40000, respectively) as the figure shows for the different leaf devices.

   *Leaf 10*:

```
set interfaces ae11 unit 0 family ethernet-switching vlan members VNI_10000
```

   *Leaf 11*:

```
set interfaces ae11 unit 0 family ethernet-switching vlan members VNI_20000
```

   *Leaf 12*:

```
set interfaces ae11 unit 0 family ethernet-switching vlan members VNI_30000
set interfaces ae11 unit 0 family ethernet-switching vlan members VNI_40000
```

2. In the MAC-VRF EVPN instance, configure the VLANs, their associated IRB interfaces, and the VLAN to VNI mappings, as the figure shows.

   *Leaf 10* (VLAN 100):

```
set routing-instances MAC-VRF-1 vlans VNI_10000 vlan-id 100
set routing-instances MAC-VRF-1 vlans VNI_10000 l3-interface irb.100
set routing-instances MAC-VRF-1 vlans VNI_10000 vxlan vni 10000
```

   *Leaf 11* (VLAN 200):

```
set routing-instances MAC-VRF-1 vlans VNI_20000 vlan-id 200
set routing-instances MAC-VRF-1 vlans VNI_20000 l3-interface irb.200
set routing-instances MAC-VRF-1 vlans VNI_20000 vxlan vni 20000
```

   *Leaf 12* (VLANs 300 and 400):

```
set routing-instances MAC-VRF-1 vlans VNI_30000 vlan-id 300
set routing-instances MAC-VRF-1 vlans VNI_30000 l3-interface irb.300
set routing-instances MAC-VRF-1 vlans VNI_30000 vxlan vni 30000
set routing-instances MAC-VRF-1 vlans VNI_40000 vlan-id 400
set routing-instances MAC-VRF-1 vlans VNI_40000 l3-interface irb.400
set routing-instances MAC-VRF-1 vlans VNI_40000 vxlan vni 40000
```

3. Configure the IRB interfaces for VLANs 100, 200, 300 and 400 on their respective leaf devices with IPv4 and IPv6 dual stack addresses for the IRB IP addresses and virtual gateway IP addresses.

   Here we use the same style of IRB interface configuration as VLAN 500 and VLAN 600 in VRF_3 (see Step 7 in "Configure an Edge-Routed Bridging Overlay on a Leaf Device" on page 211). Assign each IRB interface a unique IP address with a virtual gateway IP address (VGA) and a virtual gateway MAC address (the virtual gateway MAC is the same for all leaf devices).

   *Leaf 10 (IRB 100)*:

```
set interfaces irb unit 100 family inet address 10.1.0.1/24 virtual-gateway-address 10.1.0.254
set interfaces irb unit 100 family inet6 address 2001:db8::10:1:0:1/112 virtual-gateway-
address 2001:db8::10:1:0:254
set interfaces irb unit 100 family inet6 address fe80:10:1:0::1/64 virtual-gateway-address
fe80:10:1:0::254
set interfaces irb unit 100 virtual-gateway-v4-mac 00:00:5e:00:00:04
set interfaces irb unit 100 virtual-gateway-v6-mac 00:00:5e:00:00:04
```

   *Leaf 11 (IRB 200)*:

```
set interfaces irb unit 200 family inet address 10.1.1.1/24 virtual-gateway-address 10.1.1.254
set interfaces irb unit 200 family inet6 address 2001:db8::10:1:1:1/112 virtual-gateway-
address 2001:db8::10:1:1:254
set interfaces irb unit 200 family inet6 address fe80:10:1:1::1/64 virtual-gateway-address
fe80:10:1:1::254
set interfaces irb unit 200 virtual-gateway-v4-mac 00:00:5e:00:00:04
set interfaces irb unit 200 virtual-gateway-v6-mac 00:00:5e:00:00:04
```

   *Leaf 12 (IRB 300 and IRB 400)*:

```
set interfaces irb unit 300 family inet address 10.1.2.1/24 virtual-gateway-address 10.1.2.254
set interfaces irb unit 300 family inet6 address 2001:db8::10:1:2:1/112 virtual-gateway-
address 2001:db8::10:1:2:254
set interfaces irb unit 300 family inet6 address fe80:10:1:2::1/64 virtual-gateway-address
fe80:10:1:2::254
set interfaces irb unit 400 family inet address 10.1.3.1/24 virtual-gateway-address 10.1.3.254
set interfaces irb unit 400 family inet6 address 2001:db8::10:1:3:1/112 virtual-gateway-
address 2001:db8::10:1:3:254
set interfaces irb unit 400 family inet6 address fe80:10:1:3::1/64 virtual-gateway-address
fe80:10:1:3::254
set interfaces irb unit 300 virtual-gateway-v4-mac 00:00:5e:00:00:04
set interfaces irb unit 300 virtual-gateway-v6-mac 00:00:5e:00:00:04
set interfaces irb unit 400 virtual-gateway-v4-mac 00:00:5e:00:00:04
set interfaces irb unit 400 virtual-gateway-v6-mac 00:00:5e:00:00:04
```

4. Enable ping for the IRB interfaces for VLANs 100, 200, 300, and 400 on their respective leaf devices.

   *Leaf 10*:

   ```
   set interfaces irb unit 100 family inet address 10.1.0.1/24 preferred
   set interfaces irb unit 100 family inet6 address 2001:db8::10:1:0:1/112 preferred
   set interfaces irb unit 100 virtual-gateway-accept-data
   ```

   *Leaf 11*:

   ```
   set interfaces irb unit 200 family inet address 10.1.1.1/24 preferred
   set interfaces irb unit 200 family inet6 address 2001:db8::10:1:1:1/112 preferred
   set interfaces irb unit 200 virtual-gateway-accept-data
   ```

   *Leaf 12*:

   ```
   set interfaces irb unit 300 family inet address 10.1.2.1/24 preferred
   set interfaces irb unit 300 family inet6 address 2001:db8::10:1:2:1/112 preferred
   set interfaces irb unit 300 virtual-gateway-accept-data
   set interfaces irb unit 400 family inet address 10.1.3.1/24 preferred
   set interfaces irb unit 400 family inet6 address 2001:db8::10:1:3:1/112 preferred
   set interfaces irb unit 400 virtual-gateway-accept-data
   ```

5. Configure the additional VRF instance VRF_2 on all three leaf devices, similarly to how you configure the other VRF instances in "Configure an Edge-Routed Bridging Overlay on a Leaf Device" on page 211, Step 13. Assign a logical loopback interface (we use unit 2 in this case) to the new VRF instance so that the VXLAN gateway can resolve ARP requests. Include the IRB interfaces in VRF_2 for the VLANs supported on each leaf device in Figure 55 on page 229. Note that on each device, we assign a VRF route distinguisher that is unique to the device (based on the device lo0.0 address for simplicity). However, we use the same VRF route target on all the devices.

   *Leaf 10*:

   ```
   set interfaces lo0 unit 2 family inet
   set routing-instances VRF_2 instance-type vrf
   set routing-instances VRF_2 interface irb.100
   set routing-instances VRF_2 interface lo0.2
   ```

```
set routing-instances VRF_2 route-distinguisher 192.168.1.10:100
set routing-instances VRF_2 vrf-target target:62273:20000
```

*Leaf 11*:

```
set interfaces lo0 unit 2 family inet
set routing-instances VRF_2 instance-type vrf
set routing-instances VRF_2 interface irb.200
set routing-instances VRF_2 interface lo0.2
set routing-instances VRF_2 route-distinguisher 192.168.1.11:200
set routing-instances VRF_2 vrf-target target:62273:20000
```

*Leaf 12*:

```
set interfaces lo0 unit 2 family inet
set routing-instances VRF_2 instance-type vrf
set routing-instances VRF_2 interface irb.300
set routing-instances VRF_2 interface irb.400
set routing-instances VRF_2 interface lo0.2
set routing-instances VRF_2 route-distinguisher 192.168.1.12:300
set routing-instances VRF_2 vrf-target target:62273:20000
```

6. Enable EVPN Type 5 routing in VRF_2, and assign an L3 transit VNI value for Type 5 routing. We configure VNI value 16777123 here.

   Similarly to how you configure the VRF instances and Type 5 routing in the other VRF instances in "Configure an Edge-Routed Bridging Overlay on a Leaf Device" on page 211, Step 13, in this step we also:

   - Enable L3 load balancing by setting the multipath option for IPv4 and IPv6 traffic.

   - Set up at least one dummy IPv4 and IPv6 route for the new tenant VRF instance, so the device can advertise at least one EVPN Type 5 route in the VRF.

   However, we don't apply the EXPORT_HOST_ROUTES export policy for IP prefix routes from that step. For the devices to invoke symmetric Type 2 routing instead of Type 5 routing, we don't configure this Type 5 export policy in VRF_2 on the leaf devices.

   *Leaf 10, Leaf 11, and Leaf 12*:

```
set routing-instances VRF_2 protocols evpn ip-prefix-routes advertise direct-nexthop
set routing-instances VRF_2 protocols evpn ip-prefix-routes encapsulation vxlan
set routing-instances VRF_2 protocols evpn ip-prefix-routes vni 16777123
set routing-instances VRF_2 routing-options multipath
set routing-instances VRF_2 routing-options rib VRF_2.inet6.0 multipath
set routing-instances VRF_2 routing-options static route 10.10.20.30/32 discard
set routing-instances VRF_2 routing-options rib VRF_2.inet6.0 static route
2001:db8::101:1:1:1/128 discard
```

**NOTE**: ACX7100 routers and QFX5210 switches don't support asymmetric VNI values on either side of a VXLAN tunnel for a given VRF. To support EVPN Type 5 routing and symmetric IRB routing with EVPN Type 2 routes on those platforms, you must configure the same L3 VNI value for a particular VRF on each of the leaf devices. On other platforms, the L3 VNI can be different on either side of the tunnel for a given VRF. For simplicity, this step uses the same L3 VNI for VRF_2 on all leaf devices.

7. Enable symmetric Type 2 routing for VRF_2 on the leaf devices using the `irb-symmetric-routing` configuration statement at the `[edit routing-instances` *`l3-vrf-name`* `protocols evpn]` hierarchy level.

   In Step 6 of this procedure, when you configure VRF instance VRF_2, you enable Type 5 routing and assign VNI 16777123 for the Type 5 VXLAN tunnel. You use the same VNI value when you enable symmetric Type 2 routing for VRF_2.

   *Leaf 10, Leaf 11, and Leaf 12*:

```
set routing-instances VRF_2 protocols evpn irb-symmetric-routing vni 16777123
```

## Verify Symmetric IRB Routing with EVPN Type 2 Routes on a Leaf Device

In "Configure Symmetric IRB Routing with EVPN Type 2 Routes on Leaf Devices" on page 227, you enable symmetric Type 2 routing in VRF_2 on:

- Leaf 10 for VLAN 100 with VNI 10000, irb.100 = 10.1.0.1/24
- Leaf 11 for VLAN 200 with VNI 20000, irb.200 = 10.1.1.1/24
- Leaf 12 for:
  - VLAN 300 with VNI 30000, irb.300 = 10.1.2.1/24
  - VLAN 400 with VNI 40000, irb.400 = 10.1.3.1/24

In this section, on Leaf 10, we view routes toward Leaf 11 to verify that the leaf devices invoke symmetric Type 2 routing.

1. Verify that Leaf 10 adds IP host routes to the VRF_2 routing table for the remote EVPN Type 2 MAC-IP route toward Leaf 11.

```
user@leaf-10> show route 10.1.1.1

VRF_2.inet.0: 32 destinations, 48 routes (32 active, 0 holddown, 0 hidden)
@ = Routing Use Only, # = Forwarding Use Only
+ = Active Route, - = Last Active, * = Both

10.1.1.1/32    *[EVPN/7] 05:57:40
                    >  to 172.16.10.1 via ae1.0
                       to 172.16.10.5 via ae2.0
```

2. Verify that the device advertises EVPN Type 2 MAC-IP routes with the VRF_2 L3 context and the L3 transit VNI you configured for VRF_2.

   From the configuration in "Configure Symmetric IRB Routing with EVPN Type 2 Routes on Leaf Devices" on page 227:

   - The `vrf-target` of MAC-VRF-1 is `target:64512:1111`.
   - The `vrf-target` for VRF_2 is `target:62273:20000`.
   - The route distinguisher for VRF_2 on Leaf 11 is `192.168.1.11:200`.
   - The L3 transit VNI for EVPN Type 5 routing and symmetric Type 2 routing is `16777123`.

```
user@leaf-10> show route table MAC-VRF-1.evpn.0 | match 10.1.1.1
2:192.168.1.11:200::20000::00:31:46:79:e4:9a::10.1.1.1/304 MAC/IP

user@leaf-10> show route table MAC-VRF-1.evpn.0 match-prefix
2:192.168.1.11:200::20000::00:31:46:79:e4:9a::10.1.1.1 extensive

MAC-VRF-1.evpn.0: 227 destinations, 417 routes (227 active, 0 holddown, 0 hidden)
2:192.168.1.11:200::20000::00:31:46:79:e4:9a::10.1.1.1/304 MAC/IP (2 entries, 1 announced)
        *BGP    Preference: 170/-101
                Route Distinguisher: 192.168.1.11:200
                Next hop type: Indirect, Next hop index: 0
                Address: 0x74c2e14
                Next-hop reference count: 44424, key opaque handle: 0x0, non-key opaque
handle: 0x0
                Source: 192.168.0.2
                Protocol next hop: 192.168.1.11
                Indirect next hop: 0x2 no-forward INH Session ID: 0
                State: <Secondary Active Ext>
                Peer AS: 4210000001
                Age: 6:00:19    Metric2: 0
                Validation State: unverified
          .
          .
          .
                Communities: **target:62273:20000 target:64512:1111** encapsulation:vxlan(0x8)
mac-mobility:0x1:sticky (sequence 0) router-mac:00:31:46:79:e4:9a
                Import Accepted
                Route Label: **20000**
                Route Label: **16777123**
                ESI: 00:00:00:00:00:00:00:00:00:00
                Localpref: 100
                Router ID: 192.168.0.2
                Primary Routing Table: bgp.evpn.0
                Thread: junos-main
                Indirect next hops: 1
                        Protocol next hop: 192.168.1.11
                        Indirect next hop: 0x2 no-forward INH Session ID: 0
                        Indirect path forwarding next hops: 2
                                Next hop type: Router
                                Next hop: 172.16.10.1 via ae1.0
                                Session Id: 0
                                Next hop: 172.16.10.5 via ae2.0
                                Session Id: 0
                                192.168.1.11/32 Originating RIB: inet.0
                                  Node path count: 1
                                  Forwarding nexthops: 2
                                        Next hop type: Router
                                        Next hop: 172.16.10.1 via ae1.0
                                        Session Id: 0
                                        Next hop: 172.16.10.5 via ae2.0
                                        Session Id: 0
         BGP    Preference: 170/-101
                Route Distinguisher: 192.168.1.11:200
                Next hop type: Indirect, Next hop index: 0
                Address: 0x74c2e14
          .
          .
          .
```

RELATED DOCUMENTATION

*ip-prefix-routes*

*proxy-macip-advertisement*

*virtual-gateway-address*

*EVPN Overview*

*Understanding VXLANs*

*Understanding EVPN with VXLAN Data Plane Encapsulation*

*Configuring EVPN Routing Instances*

*Example: Configuring an ESI on a Logical Interface With EVPN Multihoming*

*Overview of VLAN Services for EVPN*

*Configuring Aggregated Ethernet LACP (CLI Procedure)*

*Routing Instances Overview*

*Example: Configuring VNI Route Targets Manually*

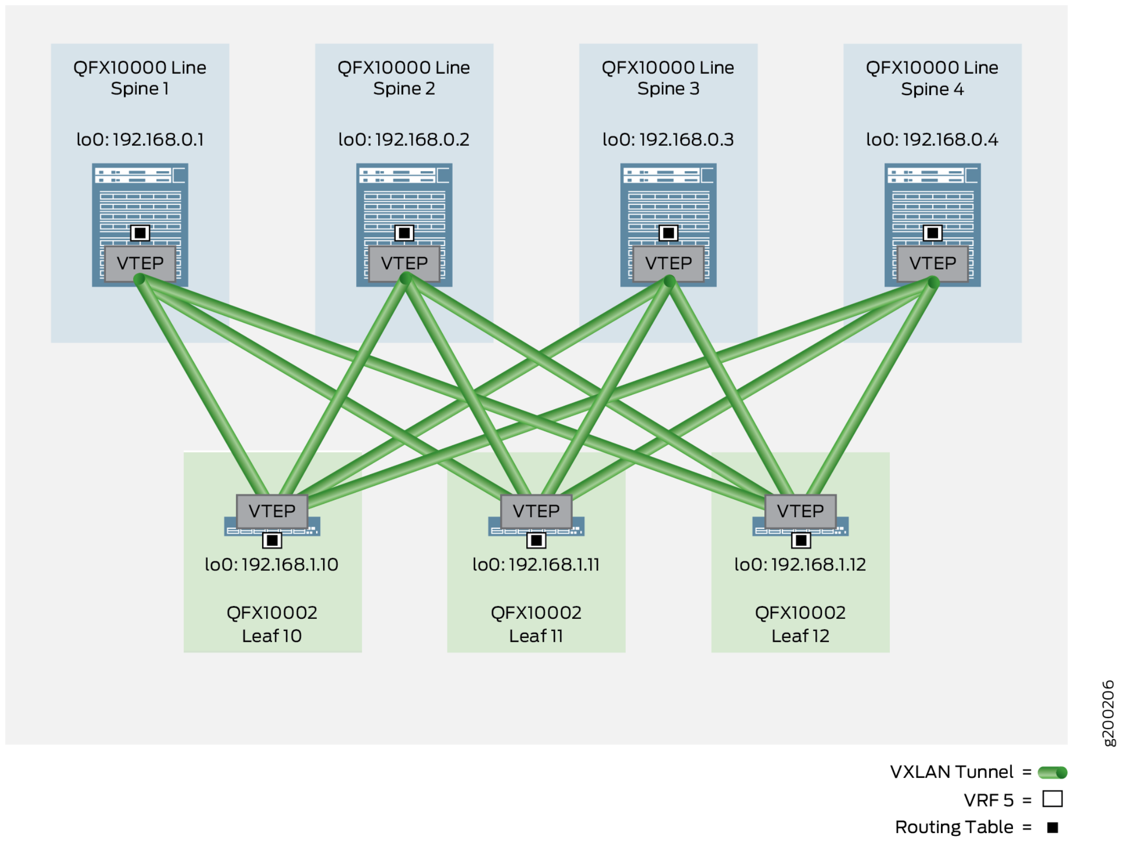
*Symmetric Integrated Routing and Bridging with EVPN Type 2 Routes in EVPN-VXLAN Fabrics*

## Routed Overlay Design and Implementation

**IN THIS SECTION**

A third overlay option for this Cloud Data Center reference design is a routed overlay, as shown in Figure 56 on page 238.

**Figure 56: Routed Overlay**



The routed overlay service supports IP-connected end systems that interconnect with leaf devices using IP (not Ethernet). The end systems can use dynamic routing protocols to exchange IP routing information with the leaf devices. The IP addresses and prefixes of the end systems are advertised as EVPN type-5 routes and imported by other EVPN/VXLAN-enabled spine and leaf devices into a corresponding VRF routing instance.

To implement a routed overlay, you must use one of the QFX10000 line of switches as the leaf devices. (The QFX5110 is planned to be verified in a future revision of this guide.)

On the spine devices, create a VRF routing instance to accept the EVPN type-5 routes. Include route distinguishers and route targets, use the `advertise direct-nexthop` option, and configure the same VNI used by the leaf devices.

To configure the leaf devices, create a policy that matches on the BGP routes received from the end systems, and send these routes into a VRF routing instance enabled for EVPN type-5 routes so they can be shared with other leaf and spine devices in the same IP VNI.

For an overview of routed overlays, see the Routed Overlay section in "Data Center Fabric Blueprint Architecture Components" on page 9.

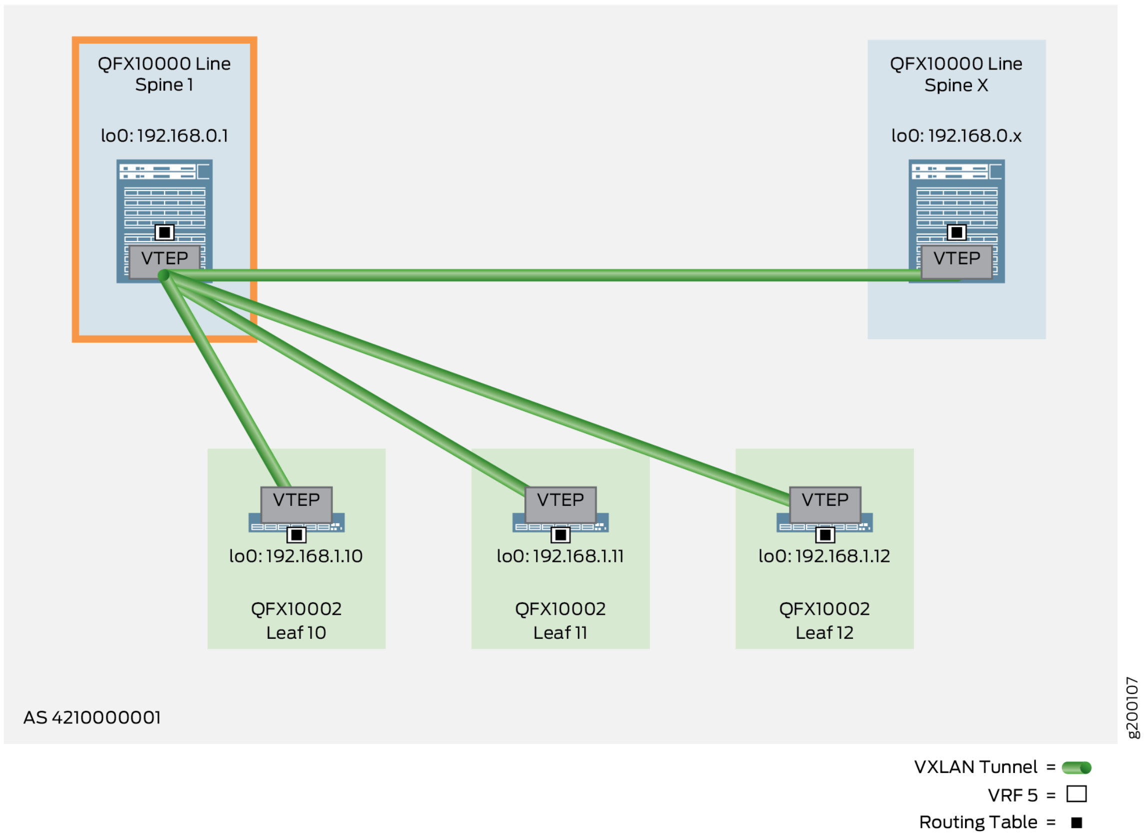
The following sections show the detailed steps of how to configure and verify the routed overlay:

## Configuring the Routed Overlay on a Spine Device

To configure the routed overlay on a spine device, perform the following:

**NOTE**: The following example shows the configuration for Spine 1, as shown in Figure 57 on page 239.

**Figure 57: Routed Overlay – Spine Devices**



1. Ensure the IP fabric underlay is in place. To see the steps required to configure an IP fabric on a spine device, see "IP Fabric Underlay Network Design and Implementation" on page 80.
2. Confirm that your IBGP overlay is up and running. To configure an IBGP overlay on your spine device, see "Configure IBGP for the Overlay" on page 96.

3. Create a VRF routing instance that enables the EVPN type-5 option (also known as IP prefix routes). Configure a route distinguisher and route target, provide load balancing for EVPN type-5 routes with the `multipath` ECMP option, enable EVPN to advertise direct next hops, specify VXLAN encapsulation, and assign the VNI.

   **NOTE**: This VRF can be used for north-south traffic flow and will be explained in a future version of this guide.

   *Spine 1*:

```
set routing-instances VRF_5 instance-type vrf
set routing-instances VRF_5 route-distinguisher 192.168.0.1:1677
set routing-instances VRF_5 vrf-target target:62273:16776000
set routing-instances VRF_5 routing-options multipath
set routing-instances VRF_5 routing-options rib VRF_5.inet6.0 multipath
set routing-instances VRF_5 protocols evpn ip-prefix-routes advertise direct-nexthop
set routing-instances VRF_5 protocols evpn ip-prefix-routes encapsulation vxlan
set routing-instances VRF_5 protocols evpn ip-prefix-routes vni 16776000
```

## Verifying the Routed Overlay on a Spine Device

To verify that the routed overlay on a spine device is working, perform the following:

1. Verify that the EVPN type-5 routes are being advertised and received for IPv4 and IPv6.

```
user@spine-1> show evpn ip-prefix-database l3-context VRF_5
L3 context: VRF_5
```

   *EVPN to IPv4 Imported Prefixes*

```
Prefix                                      Etag
172.16.104.0/30                               0
  Route distinguisher     VNI/Label  Router MAC         Nexthop/Overlay GW/ESI
  192.168.1.10:10         16776000   06:31:46:e2:f0:2a  192.168.1.10
172.16.104.4/30                               0
  Route distinguisher     VNI/Label  Router MAC         Nexthop/Overlay GW/ESI
  192.168.1.11:10       16776001   06:ac:ac:23:0c:4e  192.168.1.11
172.16.104.8/30                                 0
  Route distinguisher    VNI/Label  Router MAC         Nexthop/Overlay GW/ESI
  192.168.1.12:10       16776002   06:ac:ac:24:18:7e  192.168.1.12
192.168.3.4/32                                0
  Route distinguisher    VNI/Label  Router MAC         Nexthop/Overlay GW/ESI
  192.168.1.10:10       16776000   06:31:46:e2:f0:2a  192.168.1.10
  192.168.1.11:10       16776001   06:ac:ac:23:0c:4e  192.168.1.11
  192.168.1.12:10       16776002   06:ac:ac:24:18:7e  192.168.1.12



EVPN-->IPv6 Imported Prefixes
Prefix                                        Etag
2001:db8::172:19:4:0/126                      0
  Route distinguisher    VNI/Label  Router MAC         Nexthop/Overlay GW/ESI
  192.168.1.10:10       16776000   06:31:46:e2:f0:2a  192.168.1.10
2001:db8::172:19:4:4/126                      0
  Route distinguisher    VNI/Label  Router MAC         Nexthop/Overlay GW/ESI
  192.168.1.11:10       16776001   06:ac:ac:23:0c:4e  192.168.1.11
2001:db8::172:19:4:8/126                      0
  Route distinguisher    VNI/Label  Router MAC         Nexthop/Overlay GW/ESI
  192.168.1.12:10       16776002   06:ac:ac:24:18:7e  192.168.1.12
2001:db8::192:168:3:4/128                     0
  Route distinguisher    VNI/Label  Router MAC         Nexthop/Overlay GW/ESI
  192.168.1.10:10       16776000   06:31:46:e2:f0:2a  192.168.1.10
  192.168.1.11:10       16776001   06:ac:ac:23:0c:4e  192.168.1.11
  192.168.1.12:10       16776002   06:ac:ac:24:18:7e  192.168.1.12
```

2. Verify that the end system is multihomed to all three QFX10000 leaf devices for both IPv4 and IPv6.

```
user@spine-1> show route 192.168.3.4 table VRF_5
VRF_5.inet.0: 4 destinations, 7 routes (4 active, 0 holddown, 0 hidden)
@ = Routing Use Only, # = Forwarding Use Only
+ = Active Route, - = Last Active, * = Both

192.168.3.4/32     @[EVPN/170] 01:15:09
                     > to 172.16.10.1 via ae10.0
                    [EVPN/170] 01:19:53
                     > to 172.16.11.1 via ae11.0
                    [EVPN/170] 00:06:21
                     > to 172.16.12.1 via ae12.0
                   #[Multipath/255] 00:06:21, metric2 0
```

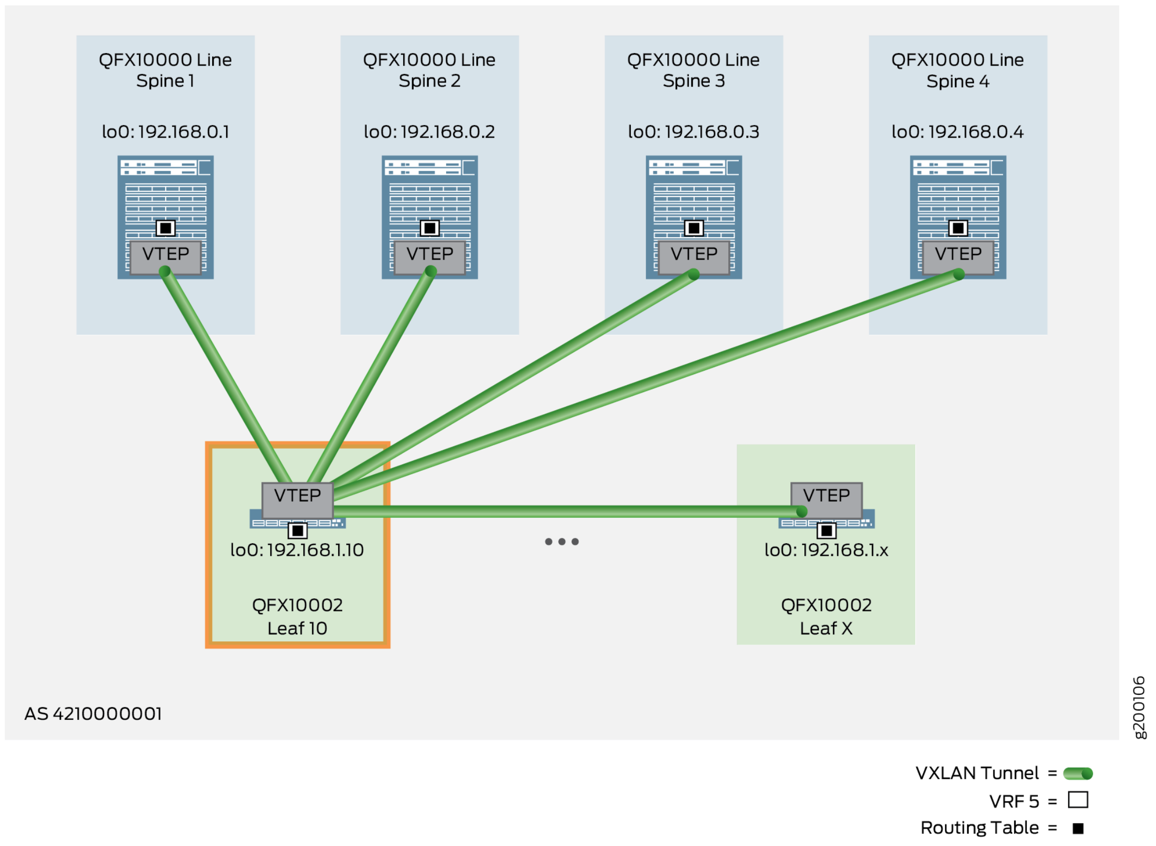
```
                       to 172.16.10.1 via ae10.0
                       to 172.16.11.1 via ae11.0
                     > to 172.16.12.1 via ae12.0
user@spine-1> show route 2001:db8::192:168:3:4 table VRF_5

VRF_5.inet6.0: 4 destinations, 7 routes (4 active, 0 holddown, 0 hidden)
@ = Routing Use Only, # = Forwarding Use Only
+ = Active Route, - = Last Active, * = Both

2001:db8::192:168:3:4/128
                   @[EVPN/170] 01:17:04
                     > to 172.16.10.1 via ae10.0
                    [EVPN/170] 01:21:48
                     > to 172.16.11.1 via ae11.0
                    [EVPN/170] 00:08:16
                     > to 172.16.12.1 via ae12.0
                   #[Multipath/255] 00:00:12, metric2 0
                       to 172.16.10.1 via ae10.0
                     > to 172.16.11.1 via ae11.0
                       to 172.16.12.1 via ae12.0
```

## Configuring the Routed Overlay on a Leaf Device

To configure the routed overlay on a leaf device, perform the following:

**NOTE**: The following example shows the configuration for Leaf 10, as shown in Figure 58 on page 243.

**Figure 58: Routed Overlay – Leaf Devices**



1. Ensure the IP fabric underlay is in place. To see the steps required to configure an IP fabric on a leaf device, see "IP Fabric Underlay Network Design and Implementation" on page 80.
2. Confirm that your IBGP overlay is up and running. To configure an IBGP overlay on your leaf device, see "Configure IBGP for the Overlay" on page 96.
3. Configure a VRF routing instance to extend EBGP peering to an end system. Specify a route target, route distinguisher, and the IP address and ASN (AS 4220000001) of the IP-connected end system to allow the leaf device and end system to become BGP neighbors. To learn how to implement IP multihoming, see "Multihoming an IP-Connected End System Design and Implementation" on page 251.

**NOTE**: Each VRF routing instance in the network should have a unique route distinguisher. In this reference design, the route distinguisher is a combination of the loopback interface IP address of the device combined with a unique identifier to signify the device. For VRF 5, the loopback interface IP address on Leaf 10 is 192.168.1.10 and the ID is 15, making the route distinguisher *192.168.1.10:15*.

*Leaf 10*:

```
set routing-instances VRF_5 instance-type vrf
set routing-instances VRF_5 interface et-0/0/15.0
set routing-instances VRF_5 protocols bgp group END_SYSTEM_1 type external
set routing-instances VRF_5 protocols bgp group END_SYSTEM_1 neighbor 172.16.104.2 peer-as
4220000001
set routing-instances VRF_5 route-distinguisher 192.168.1.10:15
set routing-instances VRF_5 vrf-target target:62273:16776000
```

4. Create a policy to match direct routes and BGP routes.

   *Leaf 10*:

```
set policy-options policy-statement ADVERTISE_EDGE_ROUTES term TERM_1 from protocol bgp
set policy-options policy-statement ADVERTISE_EDGE_ROUTES term TERM_1 then accept
set policy-options policy-statement ADVERTISE_EDGE_ROUTES term TERM_2 from protocol direct
set policy-options policy-statement ADVERTISE_EDGE_ROUTES term TERM_2 then accept
set policy-options policy-statement ADVERTISE_EDGE_ROUTES term TERM_3 then reject
```

5. Complete your configuration of the VRF routing instance by configuring the EVPN type-5 option. To implement this feature, configure EVPN to advertise direct next hops such as the IP-connected end system, specify VXLAN encapsulation, assign the VNI, and export the BGP policy that accepts the end system routes.

   *Leaf 10*:

```
set routing-instances VRF_5 protocols evpn ip-prefix-routes advertise direct-nexthop
set routing-instances VRF_5 protocols evpn ip-prefix-routes encapsulation vxlan
set routing-instances VRF_5 protocols evpn ip-prefix-routes vni 16776000
set routing-instances VRF_5 protocols evpn ip-prefix-routes export ADVERTISE_EDGE_ROUTES
```

## Verifying the Routed Overlay on a Leaf Device

**NOTE**: The operational mode command output included in the following sections assumes you have configured IP multihoming as explained in "Multihoming an IP-Connected End System Design and Implementation" on page 251.

To verify that the routed overlay on a leaf device is working, perform the following:

1. Verify that the IPv4 routes from the VRF are converted to EVPN type-5 routes are advertised to EVPN peers.

```
user@leaf-10> show evpn ip-prefix-database l3-context VRF_5

L3 context: VRF_5

```

*IPv4 to EVPN Exported Prefixes*

```
Prefix                                       EVPN route status
172.16.104.0/30                              Created
192.168.3.4/32                               Created  ## Note: this is the Lo0 interface of a
end system
```

*IPv6 to EVPN Exported Prefixes*

```
Prefix                                       EVPN route status
2001:db8::172:19:4:0/126                     Created
2001:db8::192:168:3:4/128                    Created  ## Note: this is the Lo0 interface of
an end systemEVPN to IPv4 Imported Prefixes. These are received as a
type-5 route, and converted back to IPv4.
Prefix                                       Etag
172.16.104.4/30                                0  ## From Leaf 11
  Route distinguisher    VNI/Label  Router MAC         Nexthop/Overlay GW/ESI
  192.168.1.11:10        16776001   06:ac:ac:23:0c:4e  192.168.1.11
172.16.104.8/30                                0  ## From Leaf 12
  Route distinguisher    VNI/Label  Router MAC         Nexthop/Overlay GW/ESI
  192.168.1.12:10        16776002   06:ac:ac:24:18:7e  192.168.1.12
192.168.3.4/32                               0  ## The loopback address of the end
system
  Route distinguisher    VNI/Label  Router MAC         Nexthop/Overlay GW/ESI
  192.168.1.11:10        16776001   06:ac:ac:23:0c:4e  192.168.1.11
  192.168.1.12:10        16776002   06:ac:ac:24:18:7e  192.168.1.12
```

*EVPN to IPv6 Imported Prefixes*

```
Prefix                                       Etag
2001:db8::172:19:4:4/126                     0
  Route distinguisher    VNI/Label  Router MAC         Nexthop/Overlay GW/ESI
  192.168.1.11:10       16776001   06:ac:ac:23:0c:4e  192.168.1.11
2001:db8::172:19:4:8/126                     0
  Route distinguisher    VNI/Label  Router MAC         Nexthop/Overlay GW/ESI
  192.168.1.12:10       16776002   06:ac:ac:24:18:7e  192.168.1.12
2001:db8::192:168:3:4/128                    0
  Route distinguisher    VNI/Label  Router MAC         Nexthop/Overlay GW/ESI
  192.168.1.11:10       16776001   06:ac:ac:23:0c:4e  192.168.1.11
  192.168.1.12:10       16776002   06:ac:ac:24:18:7e  192.168.1.12
```

2. View the VRF route table for IPv4, IPv6, and EVPN to verify that the end system routes and spine device routes are being exchanged.

```
user@leaf-10> show route table VRF_5
```

*This is the IPv4 section*

```
VRF_5.inet.0: 5 destinations, 7 routes (5 active, 0 holddown, 0 hidden)
@ = Routing Use Only, # = Forwarding Use Only
+ = Active Route, - = Last Active, * = Both
172.16.104.0/30      *[Direct/0] 01:33:42
                     > via et-0/0/15.0
172.16.104.1/32      *[Local/0] 01:33:42
                       Local via et-0/0/15.0
172.16.104.4/30      *[EVPN/170] 01:10:08
                       to 172.16.10.2 via ae1.0
                     > to 172.16.10.6 via ae2.0
172.16.104.8/30      *[EVPN/170] 00:01:25
                     > to 172.16.10.2 via ae1.0
                       to 172.16.10.6 via ae2.0
192.168.3.4/32     *[BGP/170] 01:33:39, localpref 100
                       AS path: 4220000001 I
                    validation-state: unverified, > to 172.16.104.2 via et-0/0/15.0
                    [EVPN/170] 01:10:08
                       to 172.16.10.2 via ae1.0
                     > to 172.16.10.6 via ae2.0
                    [EVPN/170] 00:01:25
                       to 172.16.10.2 via ae1.0
                     > to 172.16.10.6 via ae2.0
```

*This is the IPv6 section*

```
VRF_5.inet6.0: 6 destinations, 8 routes (6 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

2001:db8::172:19:4:0/126
                   *[Direct/0] 01:33:33
                    > via et-0/0/15.0
2001:db8::172:19:4:1/128
                   *[Local/0] 01:33:33
                      Local via et-0/0/15.0
2001:db8::172:19:4:4/126
                   *[EVPN/170] 01:10:08
                      to 172.16.10.2 via ae1.0
                    > to 172.16.10.6 via ae2.0
2001:db8::172:19:4:8/126
                   *[EVPN/170] 00:01:25
                    > to 172.16.10.2 via ae1.0
                      to 172.16.10.6 via ae2.0
2001:db8::192:168:3:4/128
                   *[BGP/170] 01:33:28, localpref 100
                      AS path: 4220000001 I, validation-state: unverified
                    > to 2001:db8::172:19:4:2 via et-0/0/15.0
                    [EVPN/170] 01:10:08
                      to 172.16.10.2 via ae1.0
                    > to 172.16.10.6 via ae2.0
                    [EVPN/170] 00:01:25
                      to 172.16.10.2 via ae1.0
                    > to 172.16.10.6 via ae2.0
fe80::231:4600:ae2:f06c/128
                   *[Local/0] 01:33:33
                      Local via et-0/0/15.0
```

*This is the EVPN section. EVPN routes use the following convention:*

***Type*:*Route Distinguisher*::0::*IP Address*::*Prefix*/ *NLRI Prefix***

```
VRF_5.evpn.0: 12 destinations, 36 routes (12 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

5:192.168.1.10:10::0::172.16.104.0::30/248
                   *[EVPN/170] 01:33:42
```

```
                      Indirect
5:192.168.1.10:10::0::192.168.3.4::32/248
                   *[EVPN/170] 01:33:39
                      Indirect
5:192.168.1.11:10::0::172.16.104.4::30/248
                   *[BGP/170] 01:10:08, localpref 100, from 192.168.0.1
                      AS path: I, validation-state: unverified
                      to 172.16.10.2 via ae1.0
                    > to 172.16.10.6 via ae2.0
                    [BGP/170] 01:09:54, localpref 100, from 192.168.0.2
                      AS path: I, validation-state: unverified
                      to 172.16.10.2 via ae1.0
                    > to 172.16.10.6 via ae2.0
                    [BGP/170] 01:10:08, localpref 100, from 192.168.0.3
                      AS path: I, validation-state: unverified
                      to 172.16.10.2 via ae1.0
                    > to 172.16.10.6 via ae2.0
                    [BGP/170] 01:09:39, localpref 100, from 192.168.0.4
                      AS path: I, validation-state: unverified
                      to 172.16.10.2 via ae1.0
                    > to 172.16.10.6 via ae2.0

...

5:192.168.1.12:10::0::192.168.3.4::32/248
                   *[BGP/170] 00:01:20, localpref 100, from 192.168.0.1
                      AS path: 4220000001 I, validation-state: unverified
                    > to 172.16.10.2 via ae1.0
                      to 172.16.10.6 via ae2.0
                    [BGP/170] 00:01:21, localpref 100, from 192.168.0.2
                      AS path: 4220000001 I, validation-state: unverified
                    > to 172.16.10.2 via ae1.0
                      to 172.16.10.6 via ae2.0
                    [BGP/170] 00:01:25, localpref 100, from 192.168.0.3
                      AS path: 4220000001 I, validation-state: unverified
                    > to 172.16.10.2 via ae1.0
                      to 172.16.10.6 via ae2.0
                    [BGP/170] 00:01:17, localpref 100, from 192.168.0.4
                      AS path: 4220000001 I, validation-state: unverified
                    > to 172.16.10.2 via ae1.0
                      to 172.16.10.6 via ae2.0

5:192.168.1.10:10::0::2001:db8::172:19:4:0::126/248
```

```
                    *[EVPN/170] 01:33:33
                       Indirect
5:192.168.1.10:10::0::2001:db8::192:168:3:4::128/248
                    *[EVPN/170] 01:33:28
                       Indirect
5:192.168.1.11:10::0::2001:db8::172:19:4:4::126/248
                    *[BGP/170] 01:10:08, localpref 100, from 192.168.0.1
                       AS path: I, validation-state: unverified
                     > to 172.16.10.2 via ae1.0
                       to 172.16.10.6 via ae2.0
                     [BGP/170] 01:09:54, localpref 100, from 192.168.0.2
                       AS path: I, validation-state: unverified
                     > to 172.16.10.2 via ae1.0
                       to 172.16.10.6 via ae2.0
                     [BGP/170] 01:10:08, localpref 100, from 192.168.0.3
                       AS path: I, validation-state: unverified
                     > to 172.16.10.2 via ae1.0
                       to 172.16.10.6 via ae2.0
                     [BGP/170] 01:09:39, localpref 100, from 192.168.0.4
                       AS path: I, validation-state: unverified
                     > to 172.16.10.2 via ae1.0
                       to 172.16.10.6 via ae2.0

...

5:192.168.1.12:10::0::2001:db8::192:168:3:4::128/248
                    *[BGP/170] 00:01:20, localpref 100, from 192.168.0.1
                       AS path: 4220000001 I, validation-state: unverified
                       to 172.16.10.2 via ae1.0
                     > to 172.16.10.6 via ae2.0
                     [BGP/170] 00:01:21, localpref 100, from 192.168.0.2
                       AS path: 4220000001 I, validation-state: unverified
                       to 172.16.10.2 via ae1.0
                     > to 172.16.10.6 via ae2.0
                     [BGP/170] 00:01:25, localpref 100, from 192.168.0.3
                       AS path: 4220000001 I, validation-state: unverified
                       to 172.16.10.2 via ae1.0
                     > to 172.16.10.6 via ae2.0
                     [BGP/170] 00:01:17, localpref 100, from 192.168.0.4
                       AS path: 4220000001 I, validation-state: unverified
                       to 172.16.10.2 via ae1.0
                     > to 172.16.10.6 via ae2.0
```

SEE ALSO

*Understanding EVPN Pure Type-5 Routes*

*ip-prefix-routes*

## Routed Overlay — Release History

Table 7 on page 250 provides a history of all of the features in this section and their support within this reference design.

**Table 7: Routed Overlay in the Cloud Data Center Reference Design – Release History**

| Release | Description |
|---|---|
| 19.1R2 | QFX10002-60C and QFX5120-32C switches running Junos OS Release 19.1R2 and later releases in the same release train support all features documented in this section. |
| 18.4R2 | QFX5120-48Y switches running Junos OS Release 18.4R2 and later releases in the same release train support all features documented in this section. |
| 18.1R3-S3 | QFX5110 switches running Junos OS Release 18.1R3-S3 and later releases in the same release train support all features documented in this section. |
| 17.3R3-S1 | All devices in the reference design that support Junos OS Release 17.3R3-S1 and later releases in the same release train also support all features documented in this section. |

RELATED DOCUMENTATION

*EVPN Overview*

*Understanding VXLANs*

*Understanding EVPN with VXLAN Data Plane Encapsulation*

*Configuring EVPN Routing Instances*

*Routing Instances Overview*

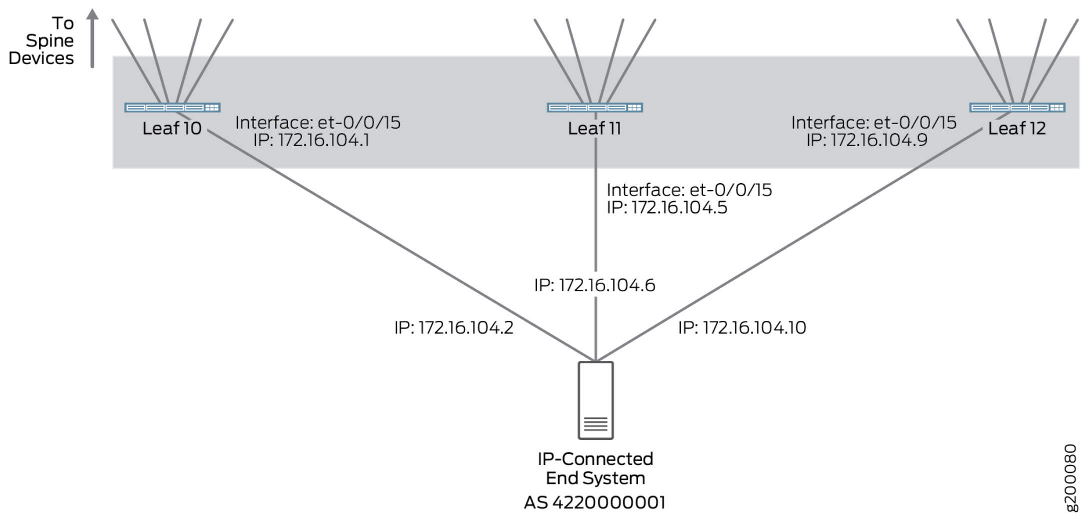
# Multihoming an IP-Connected End System Design and Implementation

**IN THIS SECTION**

For an overview of multihoming an IP-connected end system, see the Multihoming Support for IP-Connected End Systems section in "Data Center Fabric Blueprint Architecture Components" on page 9.

Figure 59 on page 251 illustrates the multihomed IP-connected end system—in this procedure, the end system is an IP-connected server—that is enabled using this procedure:

**Figure 59: Multihomed IP-Connected End System Example**



**NOTE**: This configuration uses one VLAN per leaf device access interface. See Configuring a Layer 3 Subinterface (CLI Procedure) to enable VLAN tagging if your setup requires multiple VLANs per leaf device access interface.

## Configuring the End System-Facing Interfaces on a Leaf Device

To configure the IP address of each end system-facing interface on the leaf devices:

1. Configure an IP address on each end system-facing leaf device interface, and put the interface in a VRF.

   *Leaf 10*:

   ```
   set interfaces et-0/0/15 unit 0 family inet address 172.16.104.1/30
   set routing-instances VRF_5 interface et-0/0/15.0
   ```

   **NOTE**: The `VRF_5` routing instance was configured earlier in this guide with this EBGP configuration. See "Routed Overlay Design and Implementation" on page 237.

   *Leaf 11*:

   ```
   set interfaces et-0/0/15 unit 0 family inet address 172.16.104.5/30
   set routing-instances VRF_5 interface et-0/0/15.0
   ```

   *Leaf 12*:

   ```
   set interfaces et-0/0/15 unit 0 family inet address 172.16.104.9/30
   set routing-instances VRF_5 interface et-0/0/15.0
   ```

   **NOTE**: The `VRF_5` routing instance was not explicitly configured for Leaf 11 and 12 earlier in this guide.

To configure the `VRF_5` routing instance to run on Leaf 11 and 12, repeat the procedures in "Routed Overlay Design and Implementation" on page 237 for these leaf devices. Be sure to configure a unique route distinguisher for each route to ensure that each leaf device route is treated as a unique route by the route reflectors.

2. After committing the configuration, confirm that each link is up and that the IP address is properly assigned.

   Sample output from Leaf 10 only is provided in this step.

   *Leaf 10*:

```
user@leaf10> show interfaces terse et-0/0/15
Interface               Admin Link Proto    Local                 Remote
et-0/0/15               up    up
et-0/0/15.0            up    up   inet     172.16.104.1/30
```

## Configuring EBGP Between the Leaf Device and the IP-Connected End System

EBGP—which is already used in this reference design to pass routes between spine and leaf devices in the underlay network—is also used in this reference design to pass routes between the IP-connected server and a leaf device.

**NOTE**: Other routing protocols can be used to pass routes between an IP-connected end system and the leaf device. However this is not recommended.

To configure EBGP between a leaf device and an IP-connected end system:

1. Configure EBGP in a VRF routing instance.

   *Leaf 10*:

```
set routing-instances VRF_5 protocols bgp group END_SYSTEM_1 type external
set routing-instances VRF_5 protocols bgp group END_SYSTEM_1 neighbor 172.16.104.2 peer-as
4220000001
```

2. Repeat this procedure on all leaf device interfaces that are multihomed to the IP-connected end system.

*Leaf 11*:

```
set routing-instances VRF_5 protocols bgp group END_SYSTEM_1 type external
set routing-instances VRF_5 protocols bgp group END_SYSTEM_1 neighbor 172.16.104.6 peer-as
4220000001
```

*Leaf 12*:

```
set routing-instances VRF_5 protocols bgp group END_SYSTEM_1 type external
set routing-instances VRF_5 protocols bgp group END_SYSTEM_1 neighbor 172.16.104.10 peer-as
4220000001
```

3. After committing the configurations, confirm that BGP is operational.

   A sample of this verification procedure on Leaf 10 is provided below.

   *Leaf 10*:

```
user@leaf10> show bgp summary instance VRF_5
Groups: 1 Peers: 2 Down peers: 0
Table          Tot Paths  Act Paths Suppressed    History Damp State    Pending
VRF_5.inet.0    1          1          0          0          0          0
VRF_5.mdt.0     0          0          0          0          0          0
VRF_5.evpn.0   32          8          0          0          0          0

Peer                     AS      InPkt     OutPkt    OutQ   Flaps Last Up/Dwn State|#Active/
Received/Accepted/Damped...
172.16.104.2        422000001     28         24       0       1      10:37     Establ
VRF_5.inet.0: 1/1/1/0
```

## Multihoming an IP-Connected End System—Release History

Table 8 on page 255 provides a history of all of the features in this section and their support within this reference design.

**Table 8: Multihoming IP-Connected End Systems Release History**

| Release | Description |
|---|---|
| 19.1R2 | QFX10002-60C and QFX5120-32C switches running Junos OS Release 19.1R2 and later releases in the same release train support all features documented in this section. |
| 17.3R3-S1 | All devices in the reference design that support Junos OS Release 17.3R3-S1 and later releases in the same release train also support all features documented in this section. |

## RELATED DOCUMENTATION

*EVPN Multihoming Overview*

Routed Overlay Design and Implementation | **237**

Edge-Routed Bridging Overlay Design and Implementation | **206**

# Five-Stage IP Fabric Design and Implementation

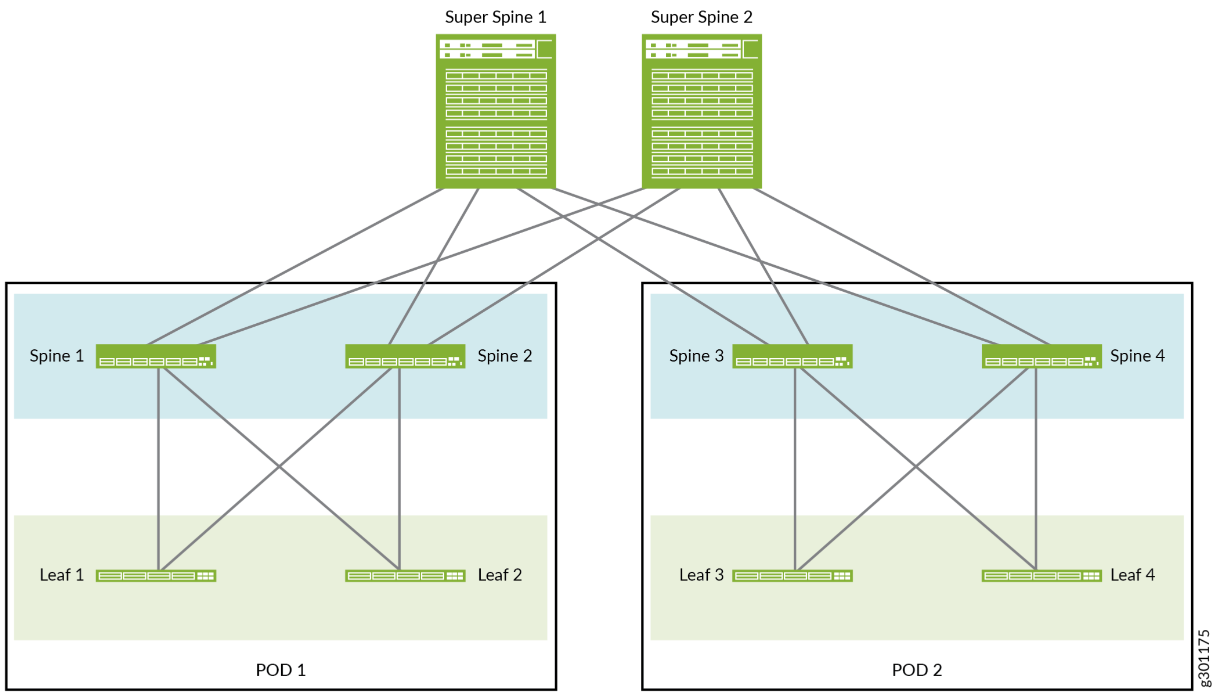
**IN THIS SECTION**

To enable you to scale your existing EVPN-VXLAN network in a data center, Juniper Networks supports a 5-stage IP fabric. Although a 5-stage IP fabric is actually comprised of 3 tiers of networking devices, the term *5-stage* refers to the number of network devices that traffic sent from one host to another must traverse to reach its destination.

Juniper Networks supports a 5-stage IP fabric in an inter-point of delivery (POD) connectivity use case within a data center. This use case assumes that your EVPN-VXLAN network already includes tiers of spine and leaf devices in two PODs. To enable connectivity between the two PODs, you add a tier of super spine devices. To determine which Juniper Networks devices you can use as a super spine device, see the "Data Center EVPN-VXLAN Fabric Reference Designs—Supported Hardware Summary" on page 51 table.

Figure 60 on page 256 shows the 5-stage IP fabric that we use in this reference design.

**Figure 60: Sample 5-Stage IP Fabric**



As shown in Figure 60 on page 256, each super spine device is connected to each spine device in each POD.

We support the following network overlay type combinations in each POD:

- The EVPN-VXLAN fabric in both PODs has a centrally routed bridging overlay.
- The EVPN-VXLAN fabric in both PODs has an edge-routed bridging overlay.
- The EVPN-VXLAN fabric in one POD has a centrally routed bridging overlay, and the fabric in the other POD has an edge-routed bridging overlay.

Juniper Network's 5-stage IP fabric supports RFC 7938, *Use of BGP for Routing in Large-Scale Data Centers*. However, where appropriate, we use terminology that more effectively describes our use case.

Note the following about the 5-stage IP fabric reference design:

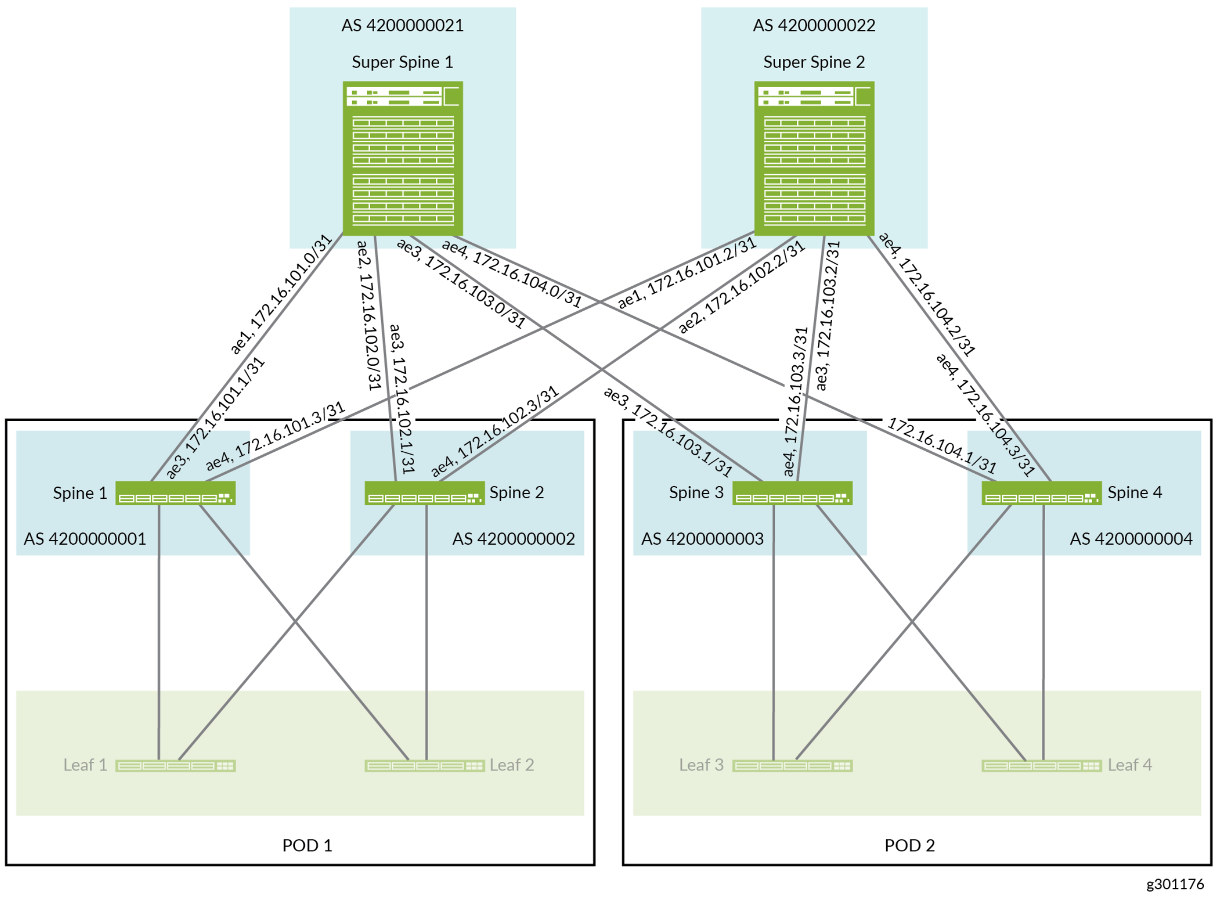
- This reference design assumes that the tiers of spine and leaf devices in the two PODs already exist and are up and running. As a result, except when describing how to configure the advertisement of EVPN type-5 routes, this topic provides the configuration for the super spine devices only. For information about configuring the spine and leaf devices in the two PODs, see the following:
    - "IP Fabric Underlay Network Design and Implementation" on page 80
    - "Configure IBGP for the Overlay" on page 96
    - "Centrally-Routed Bridging Overlay Design and Implementation" on page 142
    - "Edge-Routed Bridging Overlay Design and Implementation" on page 206
    - "Multicast IGMP Snooping and PIM Design and Implementation" on page 486
    - "Multicast Optimization Design and Implementation" on page 495
- The reference design integrates Super Spines 1 and 2 into existing IP fabric underlay and EVPN overlay networks.
- The super spine devices have the following functions:
    - They act as IP transit devices only.
    - They serve as route reflectors for Spines 1 through 4.
- When configuring the routing protocol in the EVPN overlay network, you can use either IBGP or EBGP. Typically, you use IBGP if your data center uses the same autonomous system (AS) number throughout and EBGP if your data center uses different AS numbers throughout. This reference design uses the IBGP configuration option. For information about the EBGP configuration option, see Over-the-Top Data Center Interconnect in an EVPN Network.
- After you integrate Super Spines 1 and 2 into existing IP fabric underlay and EVPN overlay networks and verify the configuration, the super spine devices will handle the communication between PODs 1 and 2 by advertising EVPN type-2 routes. This method will work if your PODs use the same IP address subnet scheme. However, if servers connected to the leaf devices in each POD are in different subnets, you must configure the devices that handle inter-subnet routing in the PODs to advertise EVPN type-5 routes. For more information, see How to Enable the Advertisement of EVPN Type-5 Routes on the Routing Devices in the PODs later in this topic.

## How to Integrate the Super Spine Devices into the IP Fabric Underlay Network

This section shows you how to configure the super spine devices so that they can communicate with the spine devices, which are already configured as part of an existing IP fabric underlay network.

For details about the interfaces and autonomous systems (ASs) in the IP fabric underlay network, see Figure 61 on page 258.

**Figure 61: Integrating Super Spine Devices into an Existing IP Fabric Underlay Network**



1. Configure the interfaces that connect the super spine devices to Spines 1 through 4.

   For the connection to each spine device, we create an aggregated Ethernet interface that currently includes a single link. We use this approach in case you need to increase the throughput to each spine device at a later time.

   For interface details for the super spine devices, see Figure 61 on page 258.

**Super Spine 1**

```
set interfaces et-0/0/1 ether-options 802.3ad ae1
set interfaces ae1 unit 0 family inet address 172.16.101.0/31
set interfaces et-0/0/2 ether-options 802.3ad ae2
set interfaces ae2 unit 0 family inet address 172.16.102.0/31
set interfaces et-0/0/3 ether-options 802.3ad ae3
set interfaces ae3 unit 0 family inet address 172.16.103.0/31
set interfaces et-0/0/4 ether-options 802.3ad ae4
set interfaces ae4 unit 0 family inet address 172.16.104.0/31
```

**Super Spine 2**

```
set interfaces et-0/0/1 ether-options 802.3ad ae1
set interfaces ae1 unit 0 family inet address 172.16.101.2/31
set interfaces et-0/0/2 ether-options 802.3ad ae2
set interfaces ae2 unit 0 family inet address 172.16.102.2/31
set interfaces et-0/0/3 ether-options 802.3ad ae3
set interfaces ae3 unit 0 family inet address 172.16.103.2/31
set interfaces et-0/0/4 ether-options 802.3ad ae4
set interfaces ae4 unit 0 family inet address 172.16.104.2/31
```

2. Specify an IP address for loopback interface lo0.0.

   We use the loopback address for each super spine device when setting up an export routing policy later in this procedure.

   **Super Spine 1**

   ```
   set interfaces lo0 unit 0 family inet address 192.168.2.1/32
   ```

   **Super Spine 2**

   ```
   set interfaces lo0 unit 0 family inet address 192.168.2.2/32
   ```

3. Configure the router ID.

   We use the router ID for each super spine device when setting up the route reflector cluster in the EVPN overlay network.

**Super Spine 1**

```
set routing-options router-id 192.168.2.1
```

**Super Spine 2**

```
set routing-options router-id 192.168.2.2
```

4. Create a BGP peer group named `underlay-bgp`, and enable EBGP as the routing protocol in the underlay network.

   **Super Spines 1 and 2**

   ```
   set protocols bgp group underlay-bgp type external
   ```

5. Configure the AS number.

   In this reference design, each device is assigned a unique AS number in the underlay network. For the AS numbers of the super spine devices, see Figure 61 on page 258.

   The AS number for EBGP in the underlay network is configured at the BGP peer group level using the `local-as` statement because the system AS number setting is used for MP-IBGP signaling in the EVPN overlay network.

   **Super Spine 1**

   ```
   set protocols bgp group underlay-bgp local-as 4200000021
   ```

   **Super Spine 2**

   ```
   set protocols bgp group underlay-bgp local-as 4200000022
   ```

6. Set up a BGP peer relationship with Spines 1 through 4.

   To establish the peer relationship, on each super spine device, configure each spine device as a neighbor by specifying the spine device's IP address and AS number. For the IP addresses and AS numbers of the spine devices, see Figure 61 on page 258.

   **Super Spine 1**

   ```
   set protocols bgp group underlay-bgp neighbor 172.16.101.1 peer-as 4200000001
   set protocols bgp group underlay-bgp neighbor 172.16.102.1 peer-as 4200000002
   ```

```
set protocols bgp group underlay-bgp neighbor 172.16.103.1 peer-as 4200000003
set protocols bgp group underlay-bgp neighbor 172.16.104.1 peer-as 4200000004
```

**Super Spine 2**

```
set protocols bgp group underlay-bgp neighbor 172.16.101.3 peer-as 4200000001
set protocols bgp group underlay-bgp neighbor 172.16.102.3 peer-as 4200000002
set protocols bgp group underlay-bgp neighbor 172.16.103.3 peer-as 4200000003
set protocols bgp group underlay-bgp neighbor 172.16.104.3 peer-as 4200000004
```

7. Configure an export routing policy that advertises the IP address of loopback interface lo0.0 on the super spine devices to the EBGP peering devices (Spines 1 through 4). This policy rejects all other advertisements.

   **Super Spines 1 and 2**

```
set policy-options policy-statement underlay-clos-export term loopback from interface lo0.0
set policy-options policy-statement underlay-clos-export term loopback then accept
set policy-options policy-statement underlay-clos-export term def then reject
set protocols bgp group underlay-bgp export underlay-clos-export
```

8. Enable multipath with the `multiple-as` option, which enables load balancing between EBGP peers in different ASs.

   EBGP, by default, selects one best path for each prefix and installs that route in the forwarding table. When BGP multipath is enabled, all equal-cost paths to a given destination are installed into the forwarding table.

   **Super Spines 1 and 2**

```
set protocols bgp group underlay-bgp multipath multiple-as
```

9. Enable Bidirectional Forwarding Detection (BFD) for all BGP sessions to enable the rapid detection of failures and reconvergence.

   **Super Spines 1 and 2**

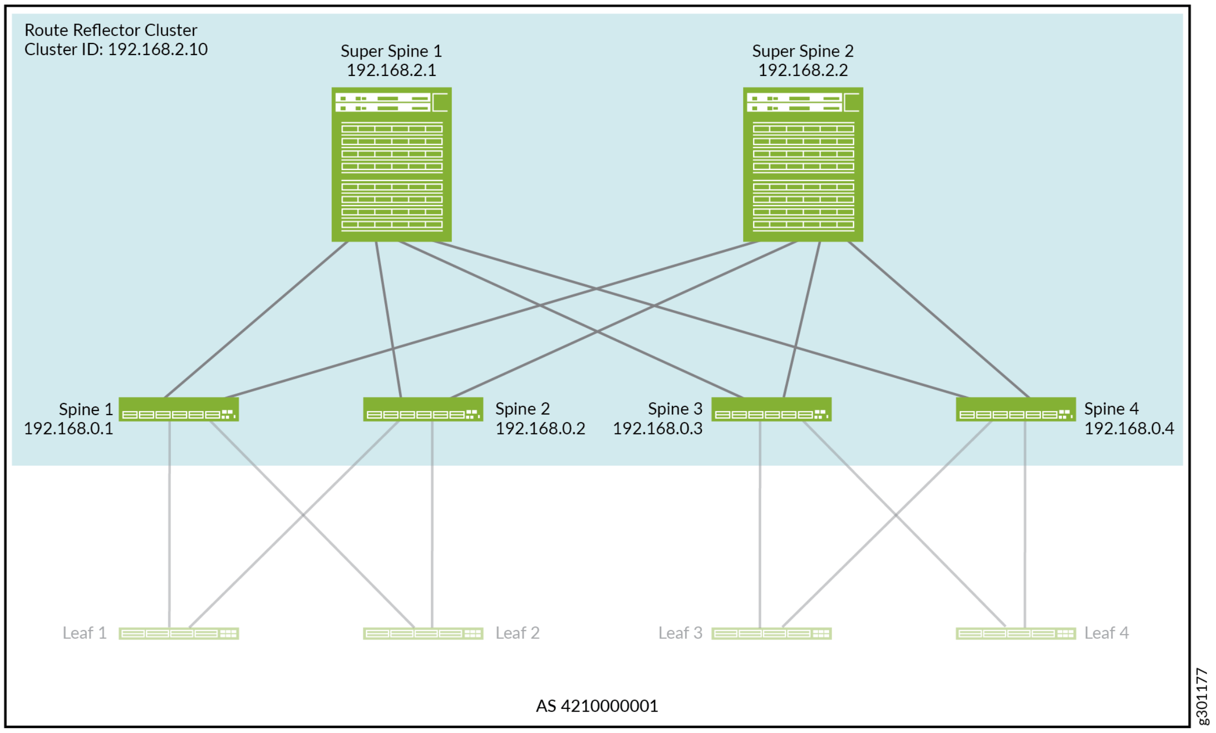
```
set protocols bgp group underlay-bgp bfd-liveness-detection minimum-interval 1000
set protocols bgp group underlay-bgp bfd-liveness-detection multiplier 3
set protocols bgp group underlay-bgp bfd-liveness-detection session-mode automatic
```

## How to Integrate the Super Spine Devices into the EVPN Overlay Network

This section explains how to integrate the super spine devices into the EVPN overlay network. In this control-plane driven overlay, we establish a signalling path between all devices within a single AS using IBGP with Multiprotocol BGP (MP-IBGP).

In this IBGP overlay, the super spine devices act as a route reflector cluster, and the spine devices are route reflector clients. For details about the route reflector cluster ID and BGP neighbor IP addresses in the EVPN overlay network, see Figure 62 on page 262.

**Figure 62: Integrating Super Spine Devices into Existing EVPN Overlay Network**



1. Configure an AS number for the IBGP overlay.

   All devices participating in this overlay (Super Spines 1 and 2, Spines 1 through 4, Leafs 1 through 4) must use the same AS number. In this example, the AS number is private AS 4210000001.

   **Super Spines 1 and 2**

   ```
   set routing-options autonomous-system 4210000001
   ```

2. Configure IBGP using EVPN signaling to peer with Spines 1 through 4. Also, form the route reflector cluster (cluster ID 192.168.2.10), and configure equal cost multipath (ECMP) for BGP. Enable path maximum transmission unit (MTU) discovery to dynamically determine the MTU size on the network path between the source and the destination, with the goal of avoiding IP fragmentation.

   For details about the route reflector cluster ID and BGP neighbor IP addresses for super spine and spine devices, see Figure 62 on page 262.

   **Super Spine 1**

   ```
   set protocols bgp group overlay-bgp type internal
   set protocols bgp group overlay-bgp local-address 192.168.2.1
   set protocols bgp group overlay-bgp mtu-discovery
   set protocols bgp group overlay-bgp family evpn signaling
   set protocols bgp group overlay-bgp cluster 192.168.2.10
   set protocols bgp group overlay-bgp multipath
   set protocols bgp group overlay-bgp neighbor 192.168.0.1
   set protocols bgp group overlay-bgp neighbor 192.168.0.2
   set protocols bgp group overlay-bgp neighbor 192.168.0.3
   set protocols bgp group overlay-bgp neighbor 192.168.0.4
   ```

   **Super Spine 2**

   ```
   set protocols bgp group overlay-bgp type internal
   set protocols bgp group overlay-bgp local-address 192.168.2.2
   set protocols bgp group overlay-bgp mtu-discovery
   set protocols bgp group overlay-bgp family evpn signaling
   set protocols bgp group overlay-bgp cluster 192.168.2.10
   set protocols bgp group overlay-bgp multipath
   set protocols bgp group overlay-bgp neighbor 192.168.0.1
   set protocols bgp group overlay-bgp neighbor 192.168.0.2
   set protocols bgp group overlay-bgp neighbor 192.168.0.3
   set protocols bgp group overlay-bgp neighbor 192.168.0.4
   ```

   **NOTE**: This reference design does not include the configuration of BGP peering between Super Spines 1 and 2. However, if you want to set up this peering to complete the full mesh peering topology, you can optionally do so by creating another BGP group and specifying the configuration in that group. For example:

   **Super Spine 1**

```
set protocols bgp group overlay-bgp2 type internal
set protocols bgp group overlay-bgp2 local-address 192.168.2.1
set protocols bgp group overlay-bgp2 family evpn signaling
set protocols bgp group overlay-bgp2 neighbor 192.168.2.2
```

**Super Spine 2**

```
set protocols bgp group overlay-bgp2 type internal
set protocols bgp group overlay-bgp2 local-address 192.168.2.2
set protocols bgp group overlay-bgp2 family evpn signaling
set protocols bgp group overlay-bgp2 neighbor 192.168.2.1
```

3. Enable BFD for all BGP sessions to enable rapid detection of failures and reconvergence.

   **Super Spines 1 and 2**

```
set protocols bgp group overlay-bgp bfd-liveness-detection minimum-interval 1000
set protocols bgp group overlay-bgp bfd-liveness-detection multiplier 3
set protocols bgp group overlay-bgp bfd-liveness-detection session-mode automatic
```

## How to Verify That the Super Spine Devices Are Integrated Into the Underlay and Overlay Networks

This section explains how you can verify that the super spine devices are properly integrated into the IP fabric underlay and EVPN overlay networks.

After you successfully complete this verification, the super spine devices will handle communication between PODs 1 and 2 by advertising EVPN type-2 routes. This method will work if your PODs use the same IP address subnet scheme. However, if each POD uses a different IP address subnet scheme, you must additionally configure the devices that handle inter-subnet routing in the PODs to advertise EVPN type-5 routes. For more information, see How to Enable the Advertisement of EVPN Type-5 Routes on the Routing Devices in the PODs later in this topic.

1. Verify that the aggregated Ethernet interfaces are enabled, that the physical links are up, and that packets are being transmitted if traffic has been sent.

   The output below provides this verification for aggregated Ethernet interface ae1 on Super Spine 1.

```
user@super-spine-1> show interfaces ae1
Physical interface: ae1, Enabled, Physical link is Up
```

```
  Interface index: 129, SNMP ifIndex: 544
  Link-level type: Ethernet, MTU: 9192, Speed: 80Gbps, BPDU Error: None, Ethernet-Switching
Error: None, MAC-REWRITE Error: None, Loopback: Disabled,
  Source filtering: Disabled, Flow control: Disabled, Minimum links needed: 1, Minimum
bandwidth needed: 1bps
  Device flags   : Present Running
  Interface flags: SNMP-Traps Internal: 0x4000
  Current address: 80:ac:ac:24:21:98, Hardware address: 80:ac:ac:24:21:98
  Last flapped   : 2020-07-30 13:09:31 PDT (3d 05:01 ago)
  Input rate     : 42963216 bps (30206 pps)
  Output rate    : 107152 bps (76 pps)

  Logical interface ae1.0 (Index 544) (SNMP ifIndex 564)
    Flags: Up SNMP-Traps 0x4004000 Encapsulation: ENET2
    Statistics        Packets        pps         Bytes          bps
    Bundle:
        Input :    7423834047      30126 1155962320326     37535088
        Output:     149534343         82   17315939427        83824
    Adaptive Statistics:
        Adaptive Adjusts:          0
        Adaptive Scans  :          0
        Adaptive Updates:          0
    Protocol inet, MTU: 9000
    Max nh cache: 75000, New hold nh limit: 75000, Curr nh cnt: 1, Curr new hold cnt: 0, NH
drop cnt: 0
      Flags: Sendbcast-pkt-to-re, Is-Primary, User-MTU
      Addresses, Flags: Is-Preferred Is-Primary
        Destination: 172.16.101.0/31, Local: 172.16.101.0
```

2. Verify that the BGP is up and running.

   The output below verifies that EBGP and IBGP peer relationships with Spines 1 through 4 are established and that traffic paths are active.

```
user@super-spine-1> show bgp summary
Threading mode: BGP I/O
Groups: 2 Peers: 8 Down peers: 0
Table          Tot Paths  Act Paths Suppressed    History Damp State    Pending
bgp.evpn.0
                  219394     210148          0          0          0          0
inet.0
                      55         27          0          0          0          0
Peer                     AS      InPkt     OutPkt    OutQ   Flaps Last Up/Dwn State|#Active/
```

```
Received/Accepted/Damped...
172.16.101.1     4200000001       9452      10053       0       2  3d 5:01:58 Establ
  inet.0: 6/14/14/0
172.16.102.1     4200000002       9462      10061       0       3  3d 4:58:50 Establ
  inet.0: 7/14/14/0
172.16.103.1     4200000003       9244       9828       0       5  3d 3:14:54 Establ
  inet.0: 7/14/14/0
172.16.104.1     4200000004       9457      10057       0       1  3d 4:58:35 Establ
  inet.0: 7/13/13/0
192.168.0.1      4210000001      29707     436404       0       2  3d 5:01:49 Establ
  bgp.evpn.0: 16897/16897/16897/0
192.168.0.2      4210000001     946949     237127       0       3  3d 4:58:51 Establ
  bgp.evpn.0: 50844/55440/55440/0
192.168.0.3      4210000001      40107     350304       0       7  3d 3:13:54 Establ
  bgp.evpn.0: 50723/55373/55373/0
192.168.0.4      4210000001      50670     274946       0       1  3d 4:58:32 Establ
  bgp.evpn.0: 91684/91684/91684/0
```

3. Verify that BFD is working.

   The output below verifies that BGP sessions between Super Spine 1 and Spines 1 through 4 are established and in the Up state.

```
user@super-spine-1> show bfd session
                                                  Detect   Transmit
Address                  State     Interface      Time     Interval  Multiplier
172.16.101.1             Up        ae1.0          3.000     1.000        3
172.16.102.1             Up        ae2.0          3.000     1.000        3
172.16.103.1             Up        ae3.0          3.000     1.000        3
172.16.104.1             Up        ae4.0          3.000     1.000        3
192.168.0.1              Up                       3.000     1.000        3
192.168.0.2              Up                       3.000     1.000        3
192.168.0.3              Up                       3.000     1.000        3
192.168.0.4              Up                       3.000     1.000        3
```

## How to Enable the Advertisement of EVPN Type-5 Routes on the Routing Devices in the PODs

After you complete the tasks in the following sections, the super spine devices will handle communication between PODs 1 and 2 by advertising EVPN type-2 routes.

- How to Integrate the Super Spine Devices into the IP Fabric Underlay Network
- How to Integrate the Super Spine Devices into the EVPN Overlay Network
- How to Verify the Integration of the Super Spine Devices Into the Underlay and Overlay Networks

If servers connected to the leaf devices in both PODs are in the same subnet, you can skip the task in this section. However, if servers in each POD are in different subnets, you must further configure the devices that handle inter-subnet routing in the PODs to advertise EVPN type-5 routes as described in this section. This type of route is also known as an *IP prefix route*.

In this EVPN type-5 reference design, the EVPN-VXLAN fabric in both PODs has a centrally routed bridging overlay. In this type of overlay, the spine devices handle inter-subnet routing. Therefore, this section explains how to enable the advertisement of EVPN type-5 routes on the spine devices in PODs 1 and 2.

To enable the advertisement of EVPN type-5 routes, you set up a tenant routing instance named VRF-1 on each spine device. In the routing instance, you specify which host IP addresses and prefixes that you want a spine device to advertise as EVPN type-5 routes with a VXLAN network identifier (VNI) of 500001. A spine device will advertise the EVPN type-5 routes to the other spine and leaf devices within the same POD. The spine device will also advertise the EVPN type-5 routes to the super spine devices, which will in turn advertise the routes to the spine devices in the other POD. All spine devices on which you have configured VRF-1 will import the EVPN type-5 routes into their VRF-1 routing table.

After you enable the advertisement of EVPN type-5 routes, the super spine devices will handle communication between PODs 1 and 2 by advertising EVPN type-5 routes.

Figure 63 on page 268 shows the EVPN type-5 configuration details for the inter-POD use case.

Figure 63: Advertisement of EVPN-Type-5 Routes Between PODs 1 and 2

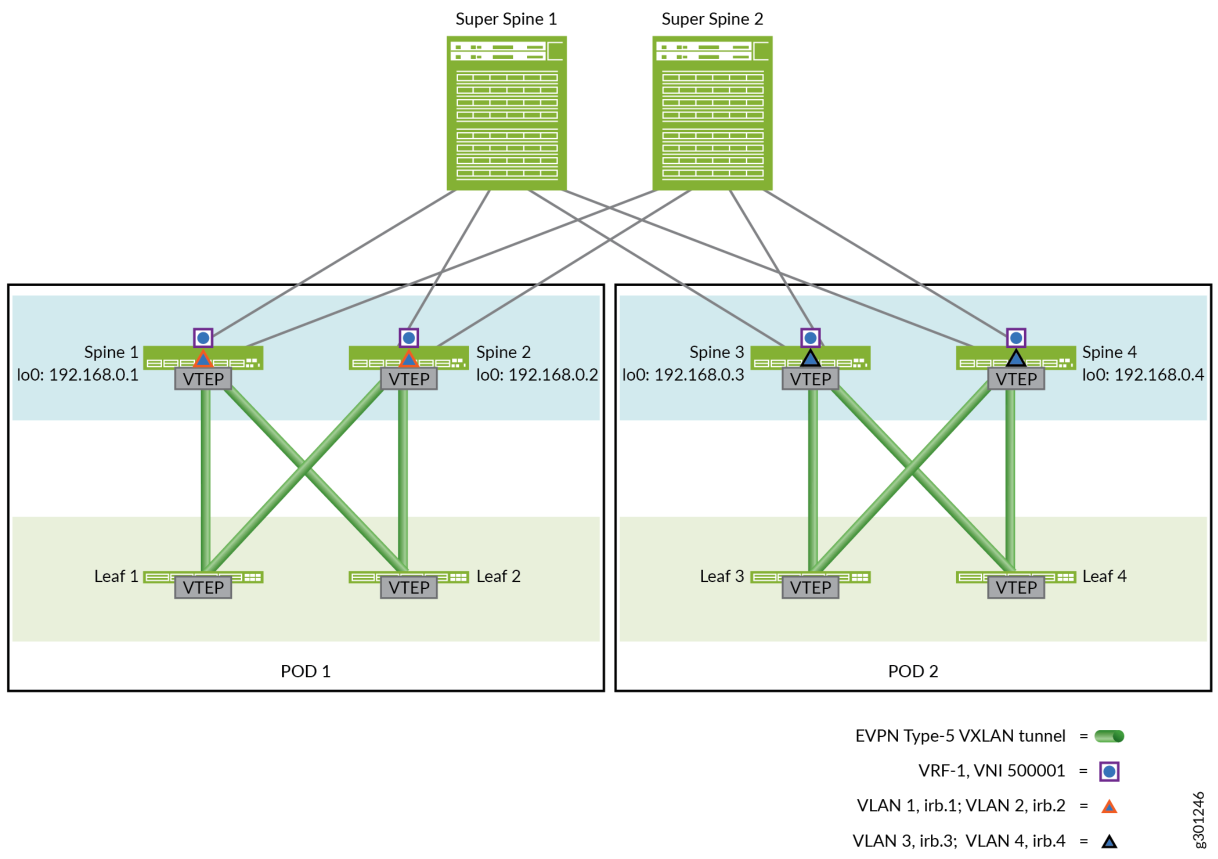


Table 9 on page 268 outlines the VLAN ID to IRB interface mappings for this reference design.

**Table 9: VLAN ID to IRB Interface Mappings**

| VLAN Names | VLAN IDs | IRB Interface |
|---|---|---|
| **Spines 1 and 2 in POD 1** | | |
| VLAN BD-1 | 1 | irb.1 |
| VLAN BD-2 | 2 | irb.2 |
| **Spines 3 and 4 in POD 2** | | |

**Table 9: VLAN ID to IRB Interface Mappings *(Continued)***

| VLAN Names | VLAN IDs | IRB Interface |
|---|---|---|
| VLAN BD-3 | 3 | irb.3 |
| VLAN BD-4 | 4 | irb.4 |

To set up the advertisement of EVPN type-5 routes:

1. Create loopback interface lo0.1, and specify that is in the IPv4 address family.

   For example:

   **Spine 1**

   ```
   set interfaces lo0 unit 1 family inet
   ```

2. Configure a routing instance of type `vrf` named VRF-1. In this routing instance, include loopback interface lo0.1 so that the spine device, which acts as a VXLAN gateway, can resolve ARP requests and the IRB interfaces that correspond to each spine device (see Table 9 on page 268). Set a route distinguisher and VRF targets for the routing instance. Configure load balancing for EVPN type-5 routes with the multipath ECMP option.

   For example:

   **Spine 1**

   ```
   set routing-instances VRF-1 instance-type vrf
   set routing-instances VRF-1 interface lo0.1
   set routing-instances VRF-1 interface irb.1
   set routing-instances VRF-1 interface irb.2
   set routing-instances VRF-1 route-distinguisher 192.168.0.1:1
   set routing-instances VRF-1 vrf-target import target:200:1
   set routing-instances VRF-1 vrf-target export target:100:1
   set routing-instances VRF-1 routing-options rib VRF-1.inet6.0 multipath
   set routing-instances VRF-1 routing-options multipath
   ```

   **Spine 2**

   ```
   set routing-instances VRF-1 instance-type vrf
   set routing-instances VRF-1 interface lo0.1
   ```

```
set routing-instances VRF-1 interface irb.1
set routing-instances VRF-1 interface irb.2
set routing-instances VRF-1 route-distinguisher 192.168.0.2:1
set routing-instances VRF-1 vrf-target import target:200:1
set routing-instances VRF-1 vrf-target export target:100:1
set routing-instances VRF-1 routing-options rib VRF-1.inet6.0 multipath
set routing-instances VRF-1 routing-options multipath
```

**Spine 3**

```
set routing-instances VRF-1 instance-type vrf
set routing-instances VRF-1 interface lo0.1
set routing-instances VRF-1 interface irb.3
set routing-instances VRF-1 interface irb.4
set routing-instances VRF-1 route-distinguisher 192.168.0.3:1
set routing-instances VRF-1 vrf-target import target:100:1
set routing-instances VRF-1 vrf-target export target:200:1
set routing-instances VRF-1 routing-options rib VRF-1.inet6.0 multipath
set routing-instances VRF-1 routing-options multipath
```

**Spine 4**

```
set routing-instances VRF-1 instance-type vrf
set routing-instances VRF-1 interface lo0.1
set routing-instances VRF-1 interface irb.3
set routing-instances VRF-1 interface irb.4
set routing-instances VRF-1 route-distinguisher 192.168.0.4:1
set routing-instances VRF-1 vrf-target import target:100:1
set routing-instances VRF-1 vrf-target export target:200:1
set routing-instances VRF-1 routing-options rib VRF-1.inet6.0 multipath
set routing-instances VRF-1 routing-options multipath
```

3. Enable EVPN to advertise direct next hops, specify VXLAN encapsulation, and assign VNI 500001 to the EVPN type-5 routes.

   For the configuration of Spines 1 through 4, use VNI 500001 in this configuration.

   For example:

**Spine 1**

```
set routing-instances VRF-1 protocols evpn ip-prefix-routes advertise direct-nexthop
set routing-instances VRF-1 protocols evpn ip-prefix-routes encapsulation vxlan
set routing-instances VRF-1 protocols evpn ip-prefix-routes vni 500001
```

4. Define an EVPN type-5 export policy named ExportHostRoutes for tenant routing instance VRF-1.

   For example, the following configuration establishes that VRF-1 advertises all host IPv4 and IPv6 addresses and prefixes learned by EVPN and from networks directly connected to Spine 1.

   **Spine 1**

```
set policy-options policy-statement ExportHostRoutes term 1 from protocol evpn
set policy-options policy-statement ExportHostRoutes term 1 from route-filter 0.0.0.0/0
prefix-length-range /32-/32
set policy-options policy-statement ExportHostRoutes term 1 then accept
set policy-options policy-statement ExportHostRoutes term 2 from family inet6
set policy-options policy-statement ExportHostRoutes term 2 from protocol evpn
set policy-options policy-statement ExportHostRoutes term 2 from route-filter 0::0/0 prefix-
length-range /128-/128
set policy-options policy-statement ExportHostRoutes term 2 then accept
set policy-options policy-statement ExportHostRoutes term 3 from protocol direct
set policy-options policy-statement ExportHostRoutes term 3 then accept
```

5. Apply the export policy named ExportHostRoutes to VRF-1.

   For example:

   **Spine 1**

```
set routing-instances VRF-1 protocols evpn ip-prefix-routes export ExportHostRoutes
```

6. In this reference design, QFX5120-32C switches act as spine devices. For these switches and all other QFX5*XXX* switches that act as spine devices in a centrally routed bridging overlay, you must perform the following additional configuration to properly implement EVPN pure type-5 routing.

   **Spines 1 through 4**

```
set routing-options forwarding-table chained-composite-next-hop ingress evpn
set forwarding-options vxlan-routing next-hop 32768
set forwarding-options vxlan-routing interface-num 8192
set forwarding-options vxlan-routing overlay-ecmp
```

## How to Verify the Advertisement of EVPN Type-5 Routes on the Routing Devices in the PODs

To verify that the spine devices in this reference design are properly advertising EVPN type-5 routes:

1. View the VRF route table to verify that the end system routes and spine device routes are being exchanged.

   The following snippet of output shows IPv4 routes only.

   **Spine 1**

```
user@Spine-1> show route table VRF-1
VRF-1.inet.0: 53 destinations, 93 routes (53 active, 0 holddown, 0 hidden)
@ = Routing Use Only, # = Forwarding Use Only
+ = Active Route, - = Last Active, * = Both

10.0.1.0/24        *[Direct/0] 1d 00:19:49
                    >  via irb.1
10.0.1.1/32        *[EVPN/7] 00:00:56
                    >  via irb.1
10.0.1.241/32      *[Local/0] 1d 00:19:49
                       Local via irb.1
10.0.1.254/32      *[Local/0] 1d 00:19:49
                       Local via irb.1
10.0.2.0/24        *[Direct/0] 1d 00:19:49
                    >  via irb.2
10.0.2.1/32        *[EVPN/7] 00:00:56
                    >  via irb.2
10.0.2.241/32      *[Local/0] 1d 00:19:49
                       Local via irb.2
10.0.2.254/32      *[Local/0] 1d 00:19:49
                       Local via irb.2
10.0.3.0/24        @[EVPN/170] 1d 00:17:54
                       to 172.16.101.0 via  ae3.0
                    >  to 172.16.101.2 via  ae4.0
                    [EVPN/170] 20:53:20
                       to 172.16.101.0 via  ae3.0
                    >  to 172.16.101.2 via  ae4.0
                   #[Multipath/255] 20:53:20, metric2 0
                       to 172.16.101.0 via  ae3.0
                    >  to 172.16.101.2 via  ae4.0
                       to 172.16.101.0 via  ae3.0
                    >  to 172.16.101.2 via  ae4.0
10.0.3.1/32        @[EVPN/170] 00:00:26
                       to 172.16.101.0 via  ae3.0
                    >  to 172.16.101.2 via  ae4.0
                    [EVPN/170] 00:00:26
                       to 172.16.101.0 via  ae3.0
                    >  to 172.16.101.2 via  ae4.0
                   #[Multipath/255] 00:00:26, metric2 0
                       to 172.16.101.0 via  ae3.0
                    >  to 172.16.101.2 via  ae4.0
                       to 172.16.101.0 via  ae3.0
                    >  to 172.16.101.2 via  ae4.0
10.0.4.0/24        @[EVPN/170] 1d 00:17:54
                       to 172.16.101.0 via  ae3.0
                    >  to 172.16.101.2 via  ae4.0
                    [EVPN/170] 20:53:20
                       to 172.16.101.0 via  ae3.0
                    >  to 172.16.101.2 via  ae4.0
                   #[Multipath/255] 20:53:20, metric2 0
                       to 172.16.101.0 via  ae3.0
                    >  to 172.16.101.2 via  ae4.0
                       to 172.16.101.0 via  ae3.0
                    >  to 172.16.101.2 via  ae4.0
10.0.4.1/32        @[EVPN/170] 00:00:26
                       to 172.16.101.0 via  ae3.0
                    >  to 172.16.101.2 via  ae4.0
                    [EVPN/170] 00:00:26
                       to 172.16.101.0 via  ae3.0
                    >  to 172.16.101.2 via  ae4.0
                   #[Multipath/255] 00:00:26, metric2 0
                       to 172.16.101.0 via  ae3.0
                    >  to 172.16.101.2 via  ae4.0
                       to 172.16.101.0 via  ae3.0
                    >  to 172.16.101.2 via  ae4.0
...
```

### Spine 3

```
user@Spine-3> show route table VRF-1
10.0.1.0/24        @[EVPN/170] 1d 00:17:54
                       to 172.16.103.0 via  ae3.0
                    >  to 172.16.103.2 via  ae4.0
```

```
                     [EVPN/170] 20:53:20
                        to 172.16.103.0 via  ae3.0
                     >  to 172.16.103.2 via  ae4.0
                    #[Multipath/255] 20:53:20, metric2 0
                        to 172.16.103.0 via  ae3.0
                     >  to 172.16.103.2 via  ae4.0
                        to 172.16.103.0 via  ae3.0
                     >  to 172.16.103.2 via  ae4.0
10.0.1.1/32         @[EVPN/170] 00:00:26
                        to 172.16.103.0 via  ae3.0
                     >  to 172.16.103.2 via  ae4.0
                     [EVPN/170] 00:00:26
                        to 172.16.103.0 via  ae3.0
                     >  to 172.16.103.2 via  ae4.0
                    #[Multipath/255] 00:00:26, metric2 0
                        to 172.16.103.0 via  ae3.0
                     >  to 172.16.103.2 via  ae4.0
                        to 172.16.103.0 via  ae3.0
                     >  to 172.16.103.2 via  ae4.0
10.0.2.0/24         @[EVPN/170] 1d 00:17:54
                        to 172.16.103.0 via  ae3.0
                     >  to 172.16.103.2 via  ae4.0
                     [EVPN/170] 20:53:20
                        to 172.16.103.0 via  ae3.0
                     >  to 172.16.103.2 via  ae4.0
                    #[Multipath/255] 20:53:20, metric2 0
                        to 172.16.103.0 via  ae3.0
                     >  to 172.16.103.2 via  ae4.0
                        to 172.16.103.0 via  ae3.0
                     >  to 172.16.103.2 via  ae4.0
10.0.2.1/32         @[EVPN/170] 00:00:26
                        to 172.16.103.0 via  ae3.0
                     >  to 172.16.103.2 via  ae4.0
                     [EVPN/170] 00:00:26
                        to 172.16.103.0 via  ae3.0
                     >  to 172.16.103.2 via  ae4.0
                    #[Multipath/255] 00:00:26, metric2 0
                        to 172.16.103.0 via  ae3.0
                     >  to 172.16.103.2 via  ae4.0
                        to 172.16.103.0 via  ae3.0
                     >  to 172.16.103.2 via  ae4.0
10.0.3.0/24         *[Direct/0] 1d 00:19:49
                     >  via irb.3
```

```
10.0.3.1/32        *[EVPN/7] 00:00:56
                    >  via irb.3
10.0.3.241/32      *[Local/0] 1d 00:19:49
                       Local via irb.3
10.0.3.254/32      *[Local/0] 1d 00:19:49
                       Local via irb.3
10.0.4.0/24        *[Direct/0] 1d 00:19:49
                    >  via irb.4
10.0.4.1/32        *[EVPN/7] 00:00:56
                    >  via irb.4
10.0.4.241/32      *[Local/0] 1d 00:19:49
                       Local via irb.4
10.0.4.254/32      *[Local/0] 1d 00:19:49
                       Local via irb.4
...
```

2. Verify that EVPN type-5 IPv4 and IPv6 routes are exported and imported into the VRF-1 routing instance.

   **Spine 1**

```
user@Spine-1> show evpn ip-prefix-database l3-context VRF-1
L3 context: VRF-1

IPv4->EVPN Exported Prefixes
Prefix                                       EVPN route status
10.0.1.0/24                                  Created
10.0.1.1/32                                  Created
10.0.2.0/24                                  Created
10.0.2.1/32                                  Created

IPv6->EVPN Exported Prefixes
Prefix                                       EVPN route status
2001:db8::10.0.1:0/112                       Created
2001:db8::10.0.1:1/128                       Created
2001:db8::10.0.2:0/112                       Created
2001:db8::10.0.2:1/128                       Created

EVPN->IPv4 Imported Prefixes
Prefix                                       Etag
10.0.3.0/24                                  0
  Route distinguisher    VNI/Label  Router MAC         Nexthop/Overlay GW/ESI
  192.168.0.3:1          500001     00:00:5e:00:53:f2  192.168.0.3
  192.168.0.4:1          500001     00:00:5e:00:53:d0  192.168.0.4
10.0.3.1/32                                 0
  Route distinguisher    VNI/Label  Router MAC         Nexthop/Overlay GW/ESI
  192.168.0.3:1          500001     00:00:5e:00:53:f2  192.168.0.3
  192.168.0.4:1          500001     00:00:5e:00:53:d0  192.168.0.4
10.0.4.0/24                                 0
  Route distinguisher    VNI/Label  Router MAC         Nexthop/Overlay GW/ESI
  192.168.0.3:1          500001     00:00:5e:00:53:f2  192.168.0.3
  192.168.0.4:1          500001     00:00:5e:00:53:d0  192.168.0.4
10.0.4.1/32                                 0
  Route distinguisher    VNI/Label  Router MAC         Nexthop/Overlay GW/ESI
  192.168.0.3:1          500001     00:00:5e:00:53:f2  192.168.0.3
  192.168.0.4:1          500001     00:00:5e:00:53:d0  192.168.0.4



EVPN->IPv6 Imported Prefixes
Prefix                                      Etag
2001:db8::10:0:3:0/112                      0
  Route distinguisher    VNI/Label  Router MAC         Nexthop/Overlay GW/ESI
  192.168.0.3:1          500001     00:00:5e:00:53:f2  192.168.0.3
  192.168.0.4:1          500001     00:00:5e:00:53:d0  192.168.0.4
2001:db8::10:0:3:1/128                      0
  Route distinguisher    VNI/Label  Router MAC         Nexthop/Overlay GW/ESI
  192.168.0.3:1          500001     00:00:5e:00:53:f2  192.168.0.3
  192.168.0.4:1          500001     00:00:5e:00:53:d0  192.168.0.4
2001:db8::10:0:4:0/112                      0
  Route distinguisher    VNI/Label  Router MAC         Nexthop/Overlay GW/ESI
  192.168.0.3:1          500001     00:00:5e:00:53:f2  192.168.0.3
  192.168.0.4:1          500001     00:00:5e:00:53:d0  192.168.0.4
2001:db8::10:0:4:1/128                      0
  Route distinguisher    VNI/Label  Router MAC         Nexthop/Overlay GW/ESI
  192.168.0.3:1          500001     00:00:5e:00:53:f2  192.168.0.3
  192.168.0.4:1          500001     00:00:5e:00:53:d0  192.168.0.4
```

### Spine 3

```
user@Spine-3> show evpn ip-prefix-database l3-context VRF-1
L3 context: VRF-1

IPv4->EVPN Exported Prefixes
Prefix                                      EVPN route status
10.0.3.0/24                                 Created
```

```
10.0.3.1/32                                 Created
10.0.4.0/24                                 Created
10.0.4.1/32                                 Created

IPv6->EVPN Exported Prefixes
Prefix                                      EVPN route status
2001:db8::10.0.3:0/112                      Created
2001:db8::10.0.3:1/128                      Created
2001:db8::10.0.4:0/112                      Created
2001:db8::10.0.4:1/128                      Created

EVPN->IPv4 Imported Prefixes
Prefix                                      Etag
10.0.1.0/24                                 0
  Route distinguisher    VNI/Label  Router MAC         Nexthop/Overlay GW/ESI
  192.168.0.1:1          500001     00:00:5e:00:53:38  192.168.0.1
  192.168.0.2:1          500001     00:00:5e:00:53:29  192.168.0.2
10.0.1.1/32                                0
  Route distinguisher    VNI/Label  Router MAC         Nexthop/Overlay GW/ESI
  192.168.0.1:1          500001     00:00:5e:00:53:38  192.168.0.1
  192.168.0.2:1          500001     00:00:5e:00:53:29  192.168.0.2
10.0.2.0/24                                 0
  Route distinguisher    VNI/Label  Router MAC         Nexthop/Overlay GW/ESI
  192.168.0.1:1          500001     00:00:5e:00:53:38  192.168.0.1
  192.168.0.2:1          500001     00:00:5e:00:53:29  192.168.0.2
10.0.2.1/32                                 0
  Route distinguisher    VNI/Label  Router MAC         Nexthop/Overlay GW/ESI
  192.168.0.1:1          500001     00:00:5e:00:53:38  192.168.0.1
  192.168.0.2:1          500001     00:00:5e:00:53:29  192.168.0.2



EVPN->IPv6 Imported Prefixes
Prefix                                      Etag
2001:db8::10:0:1:0/112                      0
  Route distinguisher    VNI/Label  Router MAC         Nexthop/Overlay GW/ESI
  192.168.0.1:1          500001     00:00:5e:00:53:38  192.168.0.1
  192.168.0.2:1          500001     00:00:5e:00:53:29  192.168.0.2
2001:db8::10:0:1:1/128                      0
  Route distinguisher    VNI/Label  Router MAC         Nexthop/Overlay GW/ESI
  192.168.0.1:1          500001     00:00:5e:00:53:38  192.168.0.1
  192.168.0.2:1          500001     00:00:5e:00:53:29  192.168.0.2
2001:db8::10:0:2:0/112                      0
  Route distinguisher    VNI/Label  Router MAC         Nexthop/Overlay GW/ESI
  192.168.0.1:1         500001     00:00:5e:00:53:38  192.168.0.1
  192.168.0.2:1         500001     00:00:5e:00:53:29  192.168.0.2
2001:db8::10:0:2:1/128                      0
  Route distinguisher    VNI/Label  Router MAC         Nexthop/Overlay GW/ESI
  192.168.0.1:1         500001     00:00:5e:00:53:38  192.168.0.1
  192.168.0.2:1         500001     00:00:5e:00:53:29  192.168.0.2
```

3. Verify the EVPN type-5 route encapsulation details. The following output shows the details for specified prefixes.

**Spine 1**

```
user@Spine-1> show route table VRF-1 10.0.4.1 extensive
VRF-1.inet.0: 53 destinations, 93 routes (53 active, 0 holddown, 0 hidden)
10.0.4.1/32 (3 entries, 1 announced)
        State: CalcForwarding
TSI:
KRT in-kernel 10.0.4.1/32 -> {list:composite(99398), composite(129244)}
        @EVPN   Preference: 170/-101
                Next hop type: Indirect, Next hop index: 0
                Address: 0x16197b18
                Next-hop reference count: 31
                Next hop type: Router, Next hop index: 0
                Next hop: 172.16.101.0 via  ae3.0
                Session Id: 0x0
                Next hop: 172.16.101.2 via  ae4.0, selected
                Session Id: 0x0
                Protocol next hop: 192.168.0.4
                Composite next hop: 0x1b8ed840 99398 INH Session ID: 0x349
                  VXLAN tunnel rewrite:
                    MTU: 0, Flags: 0x0
                    Encap table ID: 0, Decap table ID: 1508
                    Encap VNI: 500001, Decap VNI: 500001
                    Source VTEP: 192.168.0.1, Destination VTEP: 192.168.0.4
                    SMAC: 00:00:5e:00:53:38, DMAC: 00:00:5e:00:53:f2
                Indirect next hop: 0x15bc4284 2101077 INH Session ID: 0x349
                State: Active Int Ext
                Age: 6:49       Metric2: 0
                Validation State: unverified
                Task: VRF-1-EVPN-L3-context
                AS path: I  (Originator)
                Cluster list:  192.168.2.10
                Originator ID: 192.168.0.4
                Communities: target:200:1 encapsulation:vxlan(0x8) router-
mac:00:00:5e:00:53:f2
                Composite next hops: 1
                        Protocol next hop: 192.168.0.4
                        Composite next hop: 0x1b8ed840 99398 INH Session ID: 0x349
                          VXLAN tunnel rewrite:
                            MTU: 0, Flags: 0x0
                            Encap table ID: 0, Decap table ID: 1508
                            Encap VNI: 500001, Decap VNI: 500001
                            Source VTEP: 192.168.0.1, Destination VTEP: 192.168.0.4
                            SMAC: 00:00:5e:00:53:38, DMAC: 00:00:5e:00:53:f2
                        Indirect next hop: 0x15bc4284 2101077 INH Session ID: 0x349
                        Indirect path forwarding next hops: 2
                                Next hop type: Router
                                Next hop: 172.16.101.0 via  ae3.0
                                Session Id: 0x0
                                Next hop: 172.16.101.2 via  ae4.0
                                Session Id: 0x0
                                192.168.0.4/32 Originating RIB: inet.0
                                  Node path count: 1
                                  Forwarding nexthops: 2
                                        Next hop type: Router
                                        Next hop: 172.16.101.0 via  ae3.0
                                        Session Id: 0x0
                                        Next hop: 172.16.101.2 via  ae4.0
                                        Session Id: 0x0
         EVPN   Preference: 170/-101
                Next hop type: Indirect, Next hop index: 0
                Address: 0x2755af1c
                Next-hop reference count: 31
                Next hop type: Router, Next hop index: 0
                Next hop: 172.16.101.0 via  ae3.0
                Session Id: 0x0
                Next hop: 172.16.101.2 via  ae4.0, selected
                Session Id: 0x0
                Protocol next hop: 192.168.0.3
                Composite next hop: 0x2a627e20 129244 INH Session ID: 0x84e
                  VXLAN tunnel rewrite:
                    MTU: 0, Flags: 0x0
                    Encap table ID: 0, Decap table ID: 1508
                    Encap VNI: 500001, Decap VNI: 500001
                    Source VTEP: 192.168.0.2, Destination VTEP: 192.168.0.3
                    SMAC: 00:00:5e:00:53:38, DMAC: 00:00:5e:00:53:d0
                Indirect next hop: 0x15bb6c04 2105498 INH Session ID: 0x84e
                State: Int Ext
                Inactive reason: Nexthop address
                Age: 6:49       Metric2: 0
                Validation State: unverified
                Task: VRF-1-EVPN-L3-context
                AS path: I  (Originator)
                Cluster list:  192.168.2.10
                Originator ID: 192.168.0.3
                Communities: target:200:1 encapsulation:vxlan(0x8) router-
mac:00:00:5e:00:53:d0
                Composite next hops: 1
                        Protocol next hop: 192.168.0.3
                        Composite next hop: 0x2a627e20 129244 INH Session ID: 0x84e
                          VXLAN tunnel rewrite:
                            MTU: 0, Flags: 0x0
                            Encap table ID: 0, Decap table ID: 1508
                            Encap VNI: 500001, Decap VNI: 500001
                            Source VTEP: 192.168.0.2, Destination VTEP: 192.168.0.3
                            SMAC: 00:00:5e:00:53:38, DMAC: 00:00:5e:00:53:d0
                        Indirect next hop: 0x15bb6c04 2105498 INH Session ID: 0x84e
                        Indirect path forwarding next hops: 2
                                Next hop type: Router
                                Next hop: 172.16.101.0 via ae3.0
                                Session Id: 0x0
                                Next hop: 172.16.101.2 via ae4.0
                                Session Id: 0x0
                                192.168.0.3/32 Originating RIB: inet.0
                                  Node path count: 1
                                  Forwarding nexthops: 2
                                        Next hop type: Router
                                        Next hop: 172.16.101.0 via ae3.0
                                        Session Id: 0x0
                                        Next hop: 172.16.101.2 via ae4.0
                                        Session Id: 0x0
        #Multipath Preference: 255
                Next hop type: Indirect, Next hop index: 0
                Address: 0xe3aa170
                Next-hop reference count: 19
                Next hop type: Router, Next hop index: 0
                Next hop: 172.16.101.0 via ae3.0
                Session Id: 0x0
                Next hop: 172.16.101.2 via ae4.0
                Session Id: 0x0
                Next hop type: Router, Next hop index: 0
                Next hop: 172.16.101.0 via ae3.0
                Session Id: 0x0
                Next hop: 172.16.101.2 via ae4.0, selected
                Session Id: 0x0
                Protocol next hop: 192.168.0.4
                Composite next hop: 0x1b8ed840 99398 INH Session ID: 0x349
                  VXLAN tunnel rewrite:
                    MTU: 0, Flags: 0x0
                    Encap table ID: 0, Decap table ID: 1508
                    Encap VNI: 500001, Decap VNI: 500001
                    Source VTEP: 192.168.0.2, Destination VTEP: 192.168.0.4
                    SMAC: 00:00:5e:00:53:38, DMAC: 00:00:5e:00:53:f2
                Indirect next hop: 0x15bc4284 2101077 INH Session ID: 0x349
                Protocol next hop: 192.168.0.3
                Composite next hop: 0x2a627e20 129244 INH Session ID: 0x84e
                  VXLAN tunnel rewrite:
                    MTU: 0, Flags: 0x0
                    Encap table ID: 0, Decap table ID: 1508
                    Encap VNI: 500001, Decap VNI: 500001
                    Source VTEP: 192.168.0.2, Destination VTEP: 192.168.0.3
                    SMAC: 00:00:5e:00:53:38, DMAC: 00:00:5e:00:53:d0
                Indirect next hop: 0x15bb6c04 2105498 INH Session ID: 0x84e
                State: ForwardingOnly Int Ext
                Inactive reason: Forwarding use only
                Age: 6:49       Metric2: 0
                Validation State: unverified
                Task: RT
                Announcement bits (1): 2-KRT
                AS path: I  (Originator)
                Cluster list:  192.168.2.10
                Originator ID: 192.168.0.4
                Communities: target:200:1 encapsulation:vxlan(0x8) router-
mac:00:00:5e:00:53:f2
```

### Spine 2

```
user@Spine-3> **show route table VRF-1 10.0.1.1 extensive**
VRF-1.inet.0: 53 destinations, 93 routes (53 active, 0 holddown, 0 hidden)
10.0.1.1/32 (3 entries, 1 announced)
        State: CalcForwarding
```

```
TSI:
KRT in-kernel 10.0.1.1/32 -> {list:composite(99398), composite(129244)}
        @EVPN   Preference: 170/-101
                Next hop type: Indirect, Next hop index: 0
                Address: 0x16197b18
                Next-hop reference count: 31
                Next hop type: Router, Next hop index: 0
                Next hop: 172.16.103.0 via  ae3.0
                Session Id: 0x0
                Next hop: 172.16.103.2 via  ae4.0, selected
                Session Id: 0x0
                Protocol next hop: 192.168.0.1
                Composite next hop: 0x1b8ed840 99398 INH Session ID: 0x349
                  VXLAN tunnel rewrite:
                    MTU: 0, Flags: 0x0
                    Encap table ID: 0, Decap table ID: 1508
                    Encap VNI: 500001, Decap VNI: 500001
                    Source VTEP: 192.168.0.3, Destination VTEP: 192.168.0.1
                    SMAC: 00:00:5e:00:53:f2, DMAC: 00:00:5e:00:53:38
                Indirect next hop: 0x15bc4284 2101077 INH Session ID: 0x349
                State: Active Int Ext
                Age: 6:49       Metric2: 0
                Validation State: unverified
                Task: VRF-1-EVPN-L3-context
                AS path: I  (Originator)
                Cluster list:  192.168.2.10
                Originator ID: 192.168.0.1
                Communities: target:100:1 encapsulation:vxlan(0x8) router-
mac:00:00:5e:00:53:38
                Composite next hops: 1
                        Protocol next hop: 192.168.0.1
                        Composite next hop: 0x1b8ed840 99398 INH Session ID: 0x349
                          VXLAN tunnel rewrite:
                            MTU: 0, Flags: 0x0
                            Encap table ID: 0, Decap table ID: 1508
                            Encap VNI: 500001, Decap VNI: 500001
                            Source VTEP: 192.168.0.3, Destination VTEP: 192.168.0.1
                            SMAC: 00:00:5e:00:53:f2, DMAC: 00:00:5e:00:53:38
                        Indirect next hop: 0x15bc4284 2101077 INH Session ID: 0x349
                        Indirect path forwarding next hops: 2
                                Next hop type: Router
                                Next hop: 172.16.103.0 via  ae3.0
                                Session Id: 0x0
                                Next hop: 172.16.103.2 via  ae4.0
                                Session Id: 0x0
                                192.168.0.1/32 Originating RIB: inet.0
                                  Node path count: 1
                                  Forwarding nexthops: 2
                                        Next hop type: Router
                                        Next hop: 172.16.103.0 via  ae3.0
                                        Session Id: 0x0
                                        Next hop: 172.16.103.2 via  ae4.0
                                        Session Id: 0x0
         EVPN   Preference: 170/-101
                Next hop type: Indirect, Next hop index: 0
                Address: 0x2755af1c
                Next-hop reference count: 31
                Next hop type: Router, Next hop index: 0
                Next hop: 172.16.103.0 via  ae3.0
                Session Id: 0x0
                Next hop: 172.16.103.2 via  ae4.0, selected
                Session Id: 0x0
                Protocol next hop: 192.168.0.2
                Composite next hop: 0x2a627e20 129244 INH Session ID: 0x84e
                  VXLAN tunnel rewrite:
                    MTU: 0, Flags: 0x0
                    Encap table ID: 0, Decap table ID: 1508
                    Encap VNI: 500001, Decap VNI: 500001
                    Source VTEP: 192.168.0.3, Destination VTEP: 192.168.0.2
                    SMAC: 00:00:5e:00:53:f2, DMAC: 00:00:5e:00:53:29
                Indirect next hop: 0x15bb6c04 2105498 INH Session ID: 0x84e
                State: Int Ext
                Inactive reason: Nexthop address
                Age: 6:49       Metric2: 0
                Validation State: unverified
                Task: VRF-1-EVPN-L3-context
                AS path: I  (Originator)
                Cluster list:  192.168.2.10
                Originator ID: 192.168.0.2
                Communities: target:100:1 encapsulation:vxlan(0x8) router-
mac:00:00:5e:00:53:29
                Composite next hops: 1
                        Protocol next hop: 192.168.0.2
                        Composite next hop: 0x2a627e20 129244 INH Session ID: 0x84e
                          VXLAN tunnel rewrite:
                            MTU: 0, Flags: 0x0
                            Encap table ID: 0, Decap table ID: 1508
                            Encap VNI: 500001, Decap VNI: 500001
                            Source VTEP: 192.168.0.3, Destination VTEP: 192.168.0.2
                            SMAC: 00:00:5e:00:53:f2, DMAC: 00:00:5e:00:53:29
                        Indirect next hop: 0x15bb6c04 2105498 INH Session ID: 0x84e
                        Indirect path forwarding next hops: 2
                                Next hop type: Router
                                Next hop: 172.16.103.0 via ae3.0
                                Session Id: 0x0
                                Next hop: 172.16.103.2 via ae4.0
                                Session Id: 0x0
                                192.168.0.3/32 Originating RIB: inet.0
                                  Node path count: 1
                                  Forwarding nexthops: 2
                                        Next hop type: Router
                                        Next hop: 172.16.103.0 via ae3.0
                                        Session Id: 0x0
                                        Next hop: 172.16.103.2 via ae4.0
                                        Session Id: 0x0
        #Multipath Preference: 255
                Next hop type: Indirect, Next hop index: 0
                Address: 0xe3aa170
                Next-hop reference count: 19
                Next hop type: Router, Next hop index: 0
                Next hop: 172.16.103.0 via ae3.0
                Session Id: 0x0
                Next hop: 172.16.103.2 via ae4.0
                Session Id: 0x0
                Next hop type: Router, Next hop index: 0
                Next hop: 172.16.103.0 via ae3.0
                Session Id: 0x0
                Next hop: 172.16.103.2 via ae4.0, selected
                Session Id: 0x0
                Protocol next hop: 192.168.0.1
                Composite next hop: 0x1b8ed840 99398 INH Session ID: 0x349
                  VXLAN tunnel rewrite:
                    MTU: 0, Flags: 0x0
                    Encap table ID: 0, Decap table ID: 1508
                    Encap VNI: 500001, Decap VNI: 500001
                    Source VTEP: 192.168.0.3, Destination VTEP: 192.168.0.1
                    SMAC: 00:00:5e:00:53:f2, DMAC: 00:00:5e:00:53:38
                Indirect next hop: 0x15bc4284 2101077 INH Session ID: 0x349
                Protocol next hop: 192.168.0.2
```

```
                Composite next hop: 0x2a627e20 129244 INH Session ID: 0x84e
                  VXLAN tunnel rewrite:
                    MTU: 0, Flags: 0x0
                    Encap table ID: 0, Decap table ID: 1508
                    Encap VNI: 500001, Decap VNI: 500001
                    Source VTEP: 192.168.0.3, Destination VTEP: 192.168.0.2
                    SMAC: 00:00:5e:00:53:f2, DMAC: 00:00:5e:00:53:29
                Indirect next hop: 0x15bb6c04 2105498 INH Session ID: 0x84e
                State: ForwardingOnly Int Ext
                Inactive reason: Forwarding use only
                Age: 6:49       Metric2: 0
                Validation State: unverified
                Task: RT
                Announcement bits (1): 2-KRT
                AS path: I  (Originator)
                Cluster list:  192.168.2.10
                Originator ID: 192.168.0.2
                Communities: target:100:1 encapsulation:vxlan(0x8) router-
mac:00:00:5e:00:53:29
```

# Collapsed Spine Fabric Design and Implementation

**IN THIS SECTION**

In collapsed spine fabrics, core EVPN-VXLAN overlay functions are collapsed only onto a spine layer. There is no leaf layer; the spine devices can interface directly to existing top-of-rack (ToR) switches in the access layer that might not support EVPN-VXLAN.

TOR switches can be multihomed to more than one spine device for access layer resiliency, which the spine devices manage using EVPN multihoming (also called ESI-LAG) the same way as the leaf devices do in other EVPN-VXLAN reference architectures. (See "Multihoming an Ethernet-Connected End System Design and Implementation" on page 199 for details.)

The spine devices also assume any border device roles for connectivity outside the data center.

Some common elements in collapsed spine architecture use cases include:

- Collapsed spine fabric with spine devices connected *back-to-back*:

  In this model, the spine devices are connected with point-to-point links. The spine devices establish BGP peering in the underlay and overlay over those links using their loopback addresses. See Figure 64 on page 287.

  Alternatively, the collapsed spine core devices can be integrated with a route reflector cluster in a super spine layer, which is explained later (our reference architecture).

- Data center locations connected with Data Center Interconnect (DCI):

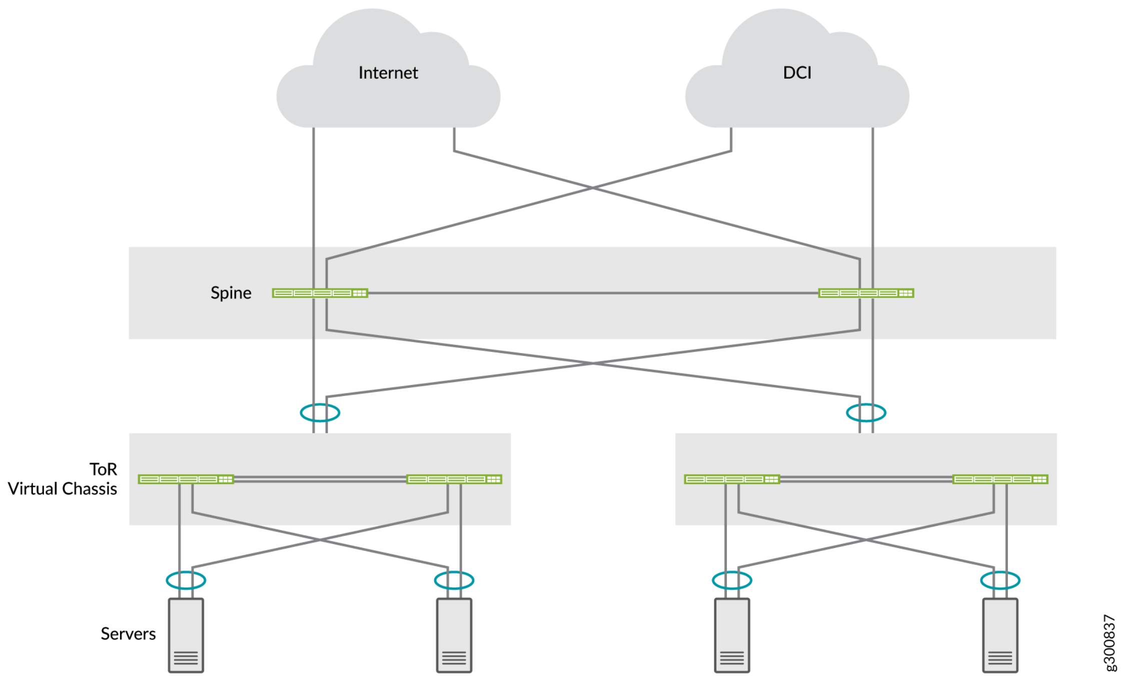
  The spine devices can perform border gateway functions to establish EVPN peering between data centers, including Layer 2 stretch and Layer 3 connectivity, as Figure 64 on page 287 shows.

- Standalone switches or Virtual Chassis in the access layer:

  The ToR layer can contain standalone switches or Virtual Chassis multihomed to the collapsed spine devices. With Virtual Chassis, you can establish redundant links in the ESI-LAGs between the spine devices and different Virtual Chassis member switches to increase resiliency. See Figure 65 on page 287.

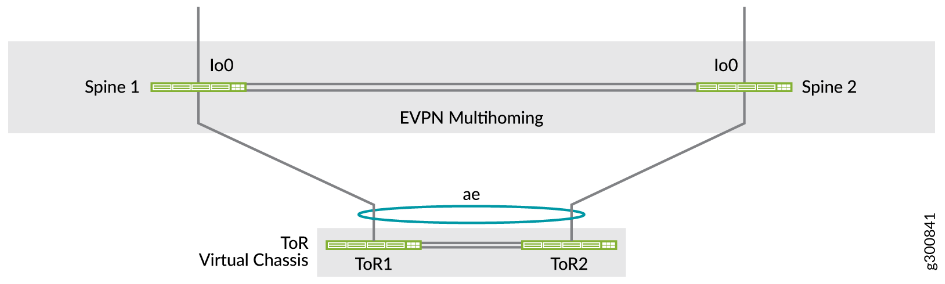
Figure 64 on page 287 shows a logical view of a collapsed spine data center with border connectivity, DCI between data centers, and Virtual Chassis in the ToR layer multihomed to the spine devices.

**Figure 64: Collapsed Spine Data Center With Multihomed Virtual Chassis TOR Devices and Data Center Interconnect**



[Figure 65 on page 287](#) illustrates Virtual Chassis in the ToR layer multihomed to a back-to-back collapsed spine layer, where the spine devices link to different Virtual Chassis member switches to improve ESI-LAG resiliency.

**Figure 65: Collapsed Spine Design With Back-to-Back Spine Devices and Multihomed Virtual Chassis in ToR Layer**



Refer to [Collapsed Spine with EVPN Multihoming](#), a network configuration example that describes a common collapsed spine use case with back-to-back spine devices. In that example, the ToR devices are Virtual Chassis that are multihomed to the collapsed spine devices. The example includes how to configure additional security services using an SRX chassis cluster to protect inter-tenant traffic, with inter-data center traffic also routed through the SRX cluster as a DCI solution.

Another collapsed spine fabric model interconnects the spine devices through an IP transit layer route reflector cluster that you integrate with the collapsed spine core underlay and overlay networks. Our reference architecture uses this model and is described in the following sections.

## Overview of Collapsed Spine Reference Architecture

Our reference architecture presents a use case for a collapsed spine data center fabric comprising two inter-point of delivery (POD) modules. The PODs and collapsed spine devices in the PODs are interconnected by a super spine IP transit layer configured as a route reflector cluster. See Figure 66 on page 289. This architecture is similar to a five-stage IP fabric design (see "Five-Stage IP Fabric Design and Implementation" on page 255), but with only the super spine, spine, and access layers. You configure the collapsed spine fabric to integrate the route reflector cluster devices into the IP fabric underlay and EVPN overlay in a similar way.

**Figure 66: Collapsed Spine Fabric Integrated With a Route Reflector Cluster**

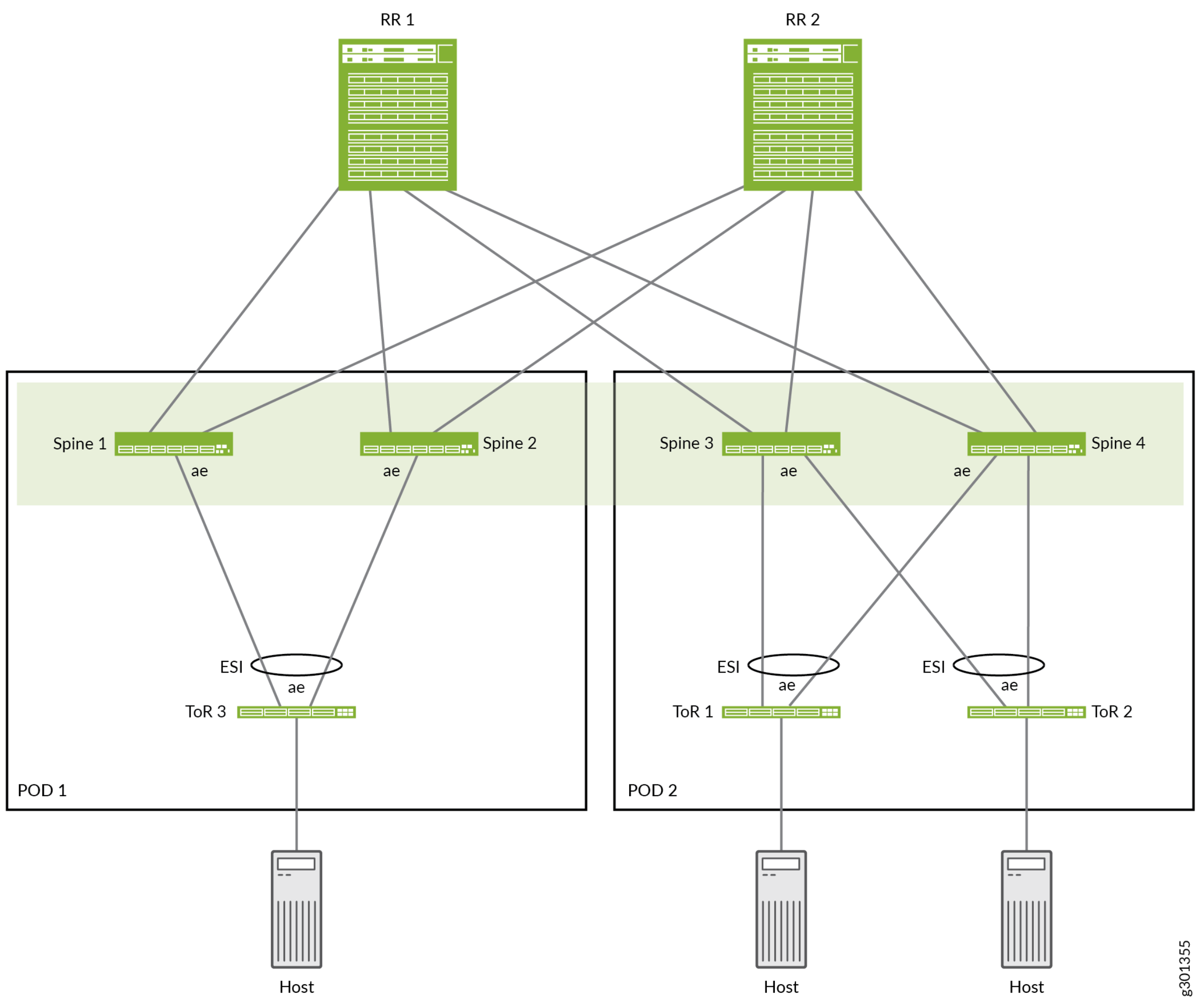


Figure 66 on page 289 shows an example of the collapsed spine reference design, which includes the following elements:

- POD 1: ToR 3 multihomed to Spine 1 and Spine 2
- POD 2: ToR 1 and ToR 2 multihomed to Spine 3 and Spine 4
- Route reflector cluster: RR 1 and RR 2 interconnecting Spine devices 1 through 4

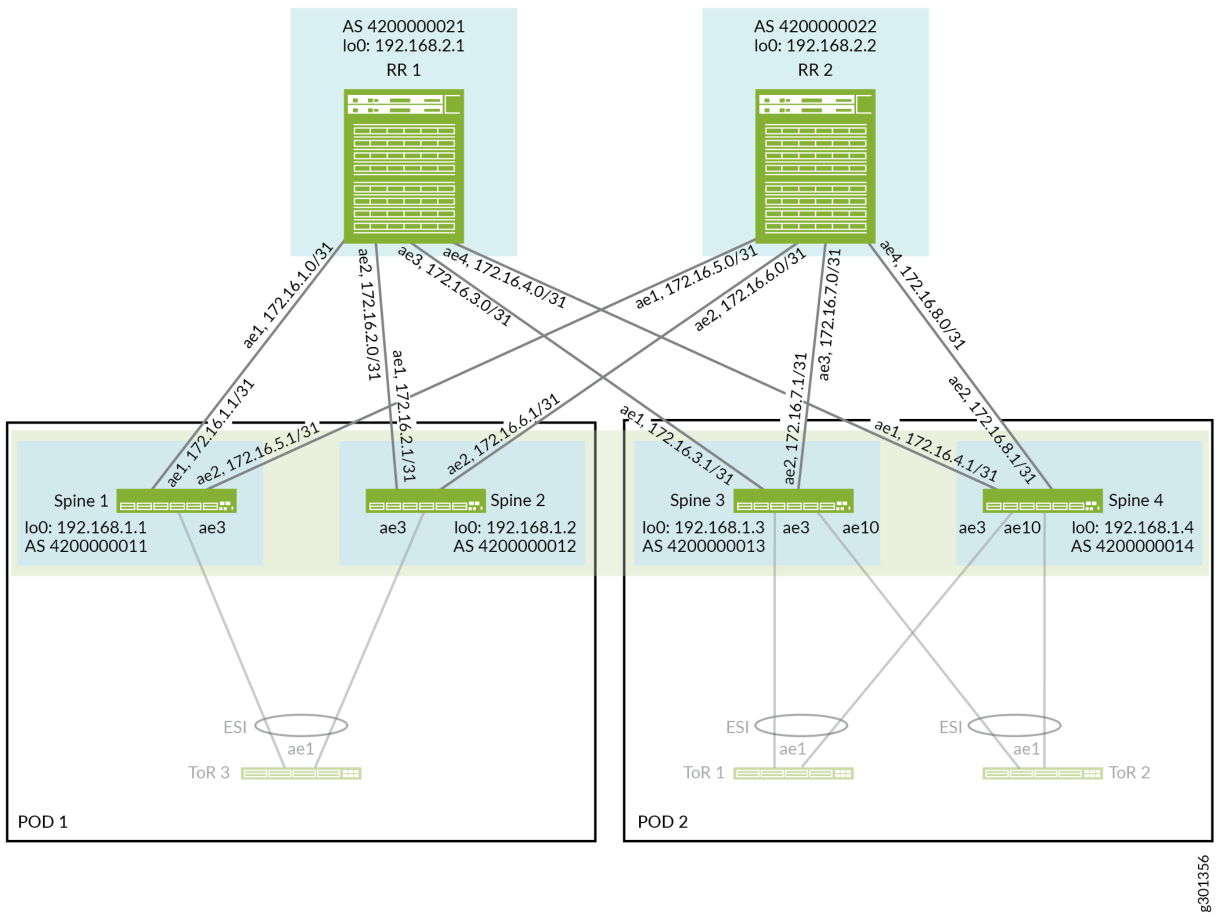
The four spine devices make up the collapsed spine EVPN fabric core, with Layer 2 stretch and Layer 3 routing between the spine devices in the two PODs. The spine devices in each POD use ESI-LAGs to the multihomed ToR switches in the same POD.

## Configure the Collapsed Spine IP Fabric Underlay Integrated With the Route Reflector Layer

This section describes how to configure the interconnecting links and the IP fabric underlay on the spine and route reflector devices.

Figure 67 on page 290 shows the collapsed spine and route reflector devices connected by aggregated Ethernet interface links.

**Figure 67: Collapsed Spine Reference Architecture Underlay Integrated With Route Reflector Cluster**



To configure the underlay:

1. Before you configure the interfaces connecting the route reflector and spine devices in the fabric, on each of those devices you must set the number of aggregated Ethernet interfaces you might need on the device. The device assigns unique MAC addresses to each aggregated Ethernet interface you configure.

Configure the number of aggregated Ethernet interfaces on RR 1, RR 2, Spine 1, Spine 2, Spine 3, and Spine 4 :

```
set chassis aggregated-devices ethernet device-count 20
```

2. Configure the aggregated Ethernet interfaces on the route reflector and spine devices that form the collapsed spine fabric as shown in Figure 67 on page 290.

   For redundancy, this reference design uses two physical interfaces in each aggregated Ethernet link between the route reflector and spine devices. The route reflector devices link to the four spine devices using aggregated Ethernet interfaces `ae1` through `ae4`. Each spine device uses aggregated Ethernet interfaces `ae1` (to RR 1) and `ae2` (to RR 2).

   Also, we configure a higher MTU (9192) on the physical interfaces to account for VXLAN encapsulation.

   *RR 1:*

```
set interfaces et-0/0/46 ether-options 802.3ad ae1
set interfaces et-0/0/62 ether-options 802.3ad ae1

set interfaces et-0/0/9 ether-options 802.3ad ae2
set interfaces et-0/0/10 ether-options 802.3ad ae2

set interfaces et-0/0/49 ether-options 802.3ad ae3
set interfaces et-0/0/58 ether-options 802.3ad ae3

set interfaces xe-0/0/34:2 ether-options 802.3ad ae4
set interfaces xe-0/0/34:3 ether-options 802.3ad ae4

set interfaces ae1 mtu 9192
set interfaces ae1 aggregated-ether-options minimum-links 1
set interfaces ae1 aggregated-ether-options lacp active
set interfaces ae1 aggregated-ether-options lacp periodic fast
set interfaces ae1 unit 0 family inet address 172.16.1.0/31

set interfaces ae2 mtu 9192
set interfaces ae2 aggregated-ether-options minimum-links 1
set interfaces ae2 aggregated-ether-options lacp active
set interfaces ae2 aggregated-ether-options lacp periodic fast
set interfaces ae2 unit 0 family inet address 172.16.2.0/31

set interfaces ae3 mtu 9192
set interfaces ae3 aggregated-ether-options minimum-links 1
set interfaces ae3 aggregated-ether-options lacp active
set interfaces ae3 aggregated-ether-options lacp periodic fast
set interfaces ae3 unit 0 family inet address 172.16.3.0/31

set interfaces ae4 mtu 9192
set interfaces ae4 aggregated-ether-options minimum-links 1
set interfaces ae4 aggregated-ether-options lacp active
set interfaces ae4 aggregated-ether-options lacp periodic fast
set interfaces ae4 unit 0 family inet address 172.16.4.0/31
```

*RR 2:*

```
set interfaces et-0/0/18 ether-options 802.3ad ae1
set interfaces et-0/0/35 ether-options 802.3ad ae1

set interfaces et-0/0/13 ether-options 802.3ad ae2
set interfaces et-0/0/14 ether-options 802.3ad ae2

set interfaces et-0/0/22 ether-options 802.3ad ae3
set interfaces et-0/0/23 ether-options 802.3ad ae3

set interfaces et-0/0/19 ether-options 802.3ad ae4
set interfaces et-0/0/20 ether-options 802.3ad ae4

set interfaces ae1 mtu 9192
set interfaces ae1 aggregated-ether-options minimum-links 1
set interfaces ae1 aggregated-ether-options lacp active
set interfaces ae1 aggregated-ether-options lacp periodic fast
set interfaces ae1 unit 0 family inet address 172.16.5.0/31

set interfaces ae2 mtu 9192
set interfaces ae2 aggregated-ether-options minimum-links 1
set interfaces ae2 aggregated-ether-options lacp active
set interfaces ae2 aggregated-ether-options lacp periodic fast
set interfaces ae2 unit 0 family inet address 172.16.6.0/31

set interfaces ae3 mtu 9192
set interfaces ae3 aggregated-ether-options minimum-links 1
set interfaces ae3 aggregated-ether-options lacp active
set interfaces ae3 aggregated-ether-options lacp periodic fast
set interfaces ae3 unit 0 family inet address 172.16.7.0/31
set interfaces ae4 mtu 9192
set interfaces ae4 aggregated-ether-options minimum-links 1
set interfaces ae4 aggregated-ether-options lacp active
set interfaces ae4 aggregated-ether-options lacp periodic fast
set interfaces ae4 unit 0 family inet address 172.16.8.0/31
```

*Spine 1:*

```
set interfaces et-0/0/1 ether-options 802.3ad ae1
set interfaces et-0/0/2 ether-options 802.3ad ae1
set interfaces et-0/0/14 ether-options 802.3ad ae2
set interfaces et-0/0/27 ether-options 802.3ad ae2

set interfaces ae1 mtu 9192
set interfaces ae1 aggregated-ether-options minimum-links 1
set interfaces ae1 aggregated-ether-options lacp active
set interfaces ae1 aggregated-ether-options lacp periodic fast
set interfaces ae1 unit 0 family inet address 172.16.1.1/31

set interfaces ae2 mtu 9192
set interfaces ae2 aggregated-ether-options minimum-links 1
set interfaces ae2 aggregated-ether-options lacp active
set interfaces ae2 aggregated-ether-options lacp periodic fast
set interfaces ae2 unit 0 family inet address 172.16.5.1/31
```

*Spine 2:*

```
set interfaces et-0/0/1 ether-options 802.3ad ae1
set interfaces et-0/0/2 ether-options 802.3ad ae1

set interfaces et-0/0/14 ether-options 802.3ad ae2
set interfaces et-0/0/15 ether-options 802.3ad ae2

set interfaces ae1 mtu 9192
set interfaces ae1 aggregated-ether-options minimum-links 1
set interfaces ae1 aggregated-ether-options lacp active
set interfaces ae1 aggregated-ether-options lacp periodic fast
set interfaces ae1 unit 0 family inet address 172.16.2.1/31

set interfaces ae2 mtu 9192
set interfaces ae2 aggregated-ether-options minimum-links 1
set interfaces ae2 aggregated-ether-options lacp active
set interfaces ae2 aggregated-ether-options lacp periodic fast
set interfaces ae2 unit 0 family inet address 172.16.6.1/31
```

*Spine 3:*

```
set interfaces et-0/0/0 ether-options 802.3ad ae1
set interfaces et-0/0/1 ether-options 802.3ad ae1

set interfaces et-0/0/7 ether-options 802.3ad ae2
set interfaces et-0/0/8 ether-options 802.3ad ae2

set interfaces ae1 mtu 9192
set interfaces ae1 aggregated-ether-options minimum-links 1
set interfaces ae1 aggregated-ether-options lacp active
set interfaces ae1 aggregated-ether-options lacp periodic fast
set interfaces ae1 unit 0 family inet address 172.16.3.1/31

set interfaces ae2 mtu 9192
set interfaces ae2 aggregated-ether-options minimum-links 1
set interfaces ae2 aggregated-ether-options lacp active
set interfaces ae2 aggregated-ether-options lacp periodic fast
set interfaces ae2 unit 0 family inet address 172.16.7.1/31
```

*Spine 4:*

```
set interfaces xe-0/0/3:2 ether-options 802.3ad ae1
set interfaces xe-0/0/3:3 ether-options 802.3ad ae1

set interfaces et-0/0/19 ether-options 802.3ad ae2
set interfaces et-0/0/20 ether-options 802.3ad ae2

set interfaces ae1 mtu 9192
set interfaces ae1 aggregated-ether-options minimum-links 1
set interfaces ae1 aggregated-ether-options lacp active
set interfaces ae1 aggregated-ether-options lacp periodic fast
set interfaces ae1 unit 0 family inet address 172.16.4.1/31

set interfaces ae2 mtu 9192
set interfaces ae2 aggregated-ether-options minimum-links 1
set interfaces ae2 aggregated-ether-options lacp active
set interfaces ae2 aggregated-ether-options lacp periodic fast
set interfaces ae2 unit 0 family inet address 172.16.8.1/31
```

3. Configure IP addresses for the loopback interfaces and the router id for each route reflector and spine device, as shown in Figure 67 on page 290.

```
set interfaces lo0 unit 0 family inet address addr/32
set routing-options router-id addr
```

4. On the route reflector and spine devices, configure the EBGP IP fabric underlay. The underlay configuration is similar to other spine and leaf reference architecture designs in "IP Fabric Underlay Network Design and Implementation" on page 80. However, in the underlay in this reference design, the collapsed spine fabric is integrated with the route reflector devices for IP transit functions between the spine devices within and across the PODs.

   The underlay configuration includes the following:

   - Define an export routing policy (`underlay-clos-export`) that advertises the IP address of the loopback interface to EBGP peering devices. This export routing policy is used to make the IP address of the loopback interface of each device reachable by all devices in the IP fabric (all route reflector and spine devices).

   - Define a local AS number on each device.

   - On the route reflector devices: Identify the four spine devices as the EBGP neighbors by their aggregated Ethernet link IP addresses and local AS numbers.

     On the spine devices: Identify the two route reflector devices as the EBGP neighbors by their aggregated Ethernet link IP addresses and local AS numbers.

   - Turn on BGP peer state transition logging.

   *RR 1:*

```
set protocols bgp group underlay-bgp type external

set policy-options policy-statement underlay-clos-export term loopback from interface lo0.0
set policy-options policy-statement underlay-clos-export term loopback then accept
set protocols bgp group underlay-bgp export underlay-clos-export

set protocols bgp group underlay-bgp local-as 4200000021
set protocols bgp group underlay-bgp multipath multiple-as

set protocols bgp group underlay-bgp neighbor 172.16.1.1 peer-as 4200000011
set protocols bgp group underlay-bgp neighbor 172.16.2.1 peer-as 4200000012
set protocols bgp group underlay-bgp neighbor 172.16.3.1 peer-as 4200000013
set protocols bgp group underlay-bgp neighbor 172.16.4.1 peer-as 4200000014
set protocols bgp log-updown
```

*RR 2:*

```
set protocols bgp group underlay-bgp type external

set policy-options policy-statement underlay-clos-export term loopback from interface lo0.0
set policy-options policy-statement underlay-clos-export term loopback then accept
set protocols bgp group underlay-bgp export underlay-clos-export

set protocols bgp group underlay-bgp local-as 4200000022
set protocols bgp group underlay-bgp multipath multiple-as

set protocols bgp group underlay-bgp neighbor 172.16.5.1 peer-as 4200000011
set protocols bgp group underlay-bgp neighbor 172.16.6.1 peer-as 4200000012
set protocols bgp group underlay-bgp neighbor 172.16.7.1 peer-as 4200000013
set protocols bgp group underlay-bgp neighbor 172.16.8.1 peer-as 4200000014
set protocols bgp log-updown
```

*Spine 1:*

```
set protocols bgp group underlay-bgp type external

set policy-options policy-statement underlay-clos-export term loopback from interface lo0.0
set policy-options policy-statement underlay-clos-export term loopback then accept
set protocols bgp group underlay-bgp export underlay-clos-export

set protocols bgp group underlay-bgp local-as 4200000011
set protocols bgp group underlay-bgp multipath multiple-as

set protocols bgp group underlay-bgp neighbor 172.16.1.0 peer-as 4200000021
set protocols bgp group underlay-bgp neighbor 172.16.5.0 peer-as 4200000022

set protocols bgp log-updown
```

*Spine 2:*

```
set protocols bgp group underlay-bgp type external

set policy-options policy-statement underlay-clos-export term loopback from interface lo0.0
set policy-options policy-statement underlay-clos-export term loopback then accept
set protocols bgp group underlay-bgp export underlay-clos-export

set protocols bgp group underlay-bgp local-as 4200000012
set protocols bgp group underlay-bgp multipath multiple-as

set protocols bgp group underlay-bgp neighbor 172.16.2.0 peer-as 4200000021
set protocols bgp group underlay-bgp neighbor 172.16.6.0 peer-as 4200000022

set protocols bgp log-updown
```

*Spine 3:*

```
set protocols bgp group underlay-bgp type external

set policy-options policy-statement underlay-clos-export term loopback from interface lo0.0
set policy-options policy-statement underlay-clos-export term loopback then accept
set protocols bgp group underlay-bgp export underlay-clos-export

set protocols bgp group underlay-bgp local-as 4200000013
set protocols bgp group underlay-bgp multipath multiple-as

set protocols bgp group underlay-bgp neighbor 172.16.3.0 peer-as 4200000021
set protocols bgp group underlay-bgp neighbor 172.16.7.0 peer-as 4200000022

set protocols bgp log-updown
```

*Spine 4:*

```
set protocols bgp group underlay-bgp type external

set policy-options policy-statement underlay-clos-export term loopback from interface lo0.0
set policy-options policy-statement underlay-clos-export term loopback then accept
set protocols bgp group underlay-bgp export underlay-clos-export

set protocols bgp group underlay-bgp local-as 4200000014
set protocols bgp group underlay-bgp multipath multiple-as

set protocols bgp group underlay-bgp neighbor 172.16.4.0 peer-as 4200000021
set protocols bgp group underlay-bgp neighbor 172.16.8.0 peer-as 4200000022

set protocols bgp log-updown
```

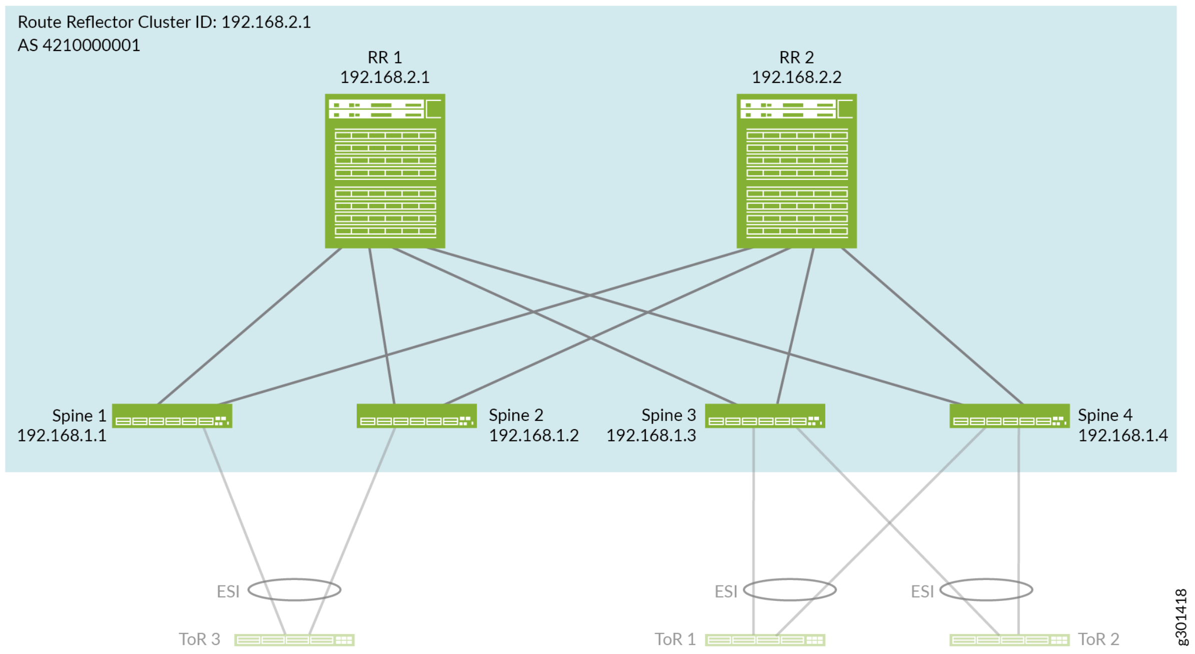
## Configure the Collapsed Spine EVPN-VXLAN Overlay Integrated With the Route Reflector Layer

In this design, the overlay is similar to other EVPN-VXLAN data center spine and leaf reference architectures, but doesn't include a leaf layer. Only the spine devices (integrated with the route reflector cluster) do intra-VLAN and inter-VLAN routing in the fabric. We configure IBGP with Multiprotocol BGP (MP-IBGP) with a single autonomous system (AS) number on the spine devices to establish a signalling path between them by way of the route reflector cluster devices as follows:

- The route reflector cluster devices peer with the spine devices in both PODs for IP transit.
- The spine devices peer with the route reflector devices.

See Figure 68 on page 299, which illustrates the spine and route reflector cluster devices and BGP neighbor IP addresses we configure in the EVPN overlay network.

**Figure 68: Collapsed Spine Reference Architecture Overlay Integrated With Route Reflector Cluster**



The overlay configuration is the same on both of the route reflector devices except for the device's local address (the loopback address). The route reflector devices peer with all of the spine devices.

The overlay configuration is the same on each of the spine devices except for the device's local address (the loopback address). All of the spine devices peer with the route reflector cluster devices.

We configure EVPN with VXLAN encapsulation and virtual tunnel endpoint (VTEP) interfaces only on the spine devices in the collapsed spine fabric.

To configure the overlay:

1. Configure an AS number for the IBGP overlay on all spine and route reflector devices:

```
set routing-options autonomous-system 4210000001
```

2. Configure IBGP with EVPN signaling on the route reflector devices to peer with the collapsed spine devices, identified as IBGP neighbors by their device loopback addresses as illustrated in Figure 68 on page 299.

   In this step, you also:

   - Define RR 1 and RR 2 as a route reflector cluster (with cluster ID 192.168.2.1).

- Enable path maximum transmission unit (MTU) discovery to dynamically determine the MTU size on the network path between the source and the destination, which can help avoid IP fragmentation.
- Set up Bidirectional Forwarding Detection (BFD) for detecting IBGP neighbor failures.
- Set the `vpn-apply-export` option to ensure that both the VRF and BGP group or neighbor export policies in the BGP configuration are applied (in that order) before the device advertises routes in the VPN routing tables to the other route reflector or spine devices. (See *Distributing VPN Routes* for more information.)

*RR 1:*

```
set protocols bgp group overlay-with-rr type internal
set protocols bgp group overlay-with-rr local-address 192.168.2.1

set protocols bgp group overlay-with-rr family evpn signaling

set protocols bgp group overlay-with-rr cluster 192.168.2.1
set protocols bgp group overlay-with-rr multipath
set protocols bgp group overlay-with-rr mtu-discovery

set protocols bgp group overlay-with-rr neighbor 192.168.1.1
set protocols bgp group overlay-with-rr neighbor 192.168.1.2
set protocols bgp group overlay-with-rr neighbor 192.168.1.3
set protocols bgp group overlay-with-rr neighbor 192.168.1.4

set protocols bgp group overlay-with-rr bfd-liveness-detection minimum-interval 1000
set protocols bgp group overlay-with-rr bfd-liveness-detection multiplier 3
set protocols bgp group overlay-with-rr bfd-liveness-detection session-mode automatic

set protocols bgp group overlay-with-rr vpn-apply-export
```

*RR 2:*

```
set protocols bgp group overlay-with-rr type internal
set protocols bgp group overlay-with-rr local-address 192.168.2.2

set protocols bgp group overlay-with-rr family evpn signaling

set protocols bgp group overlay-with-rr cluster 192.168.2.1
set protocols bgp group overlay-with-rr multipath
set protocols bgp group overlay-with-rr mtu-discovery

set protocols bgp group overlay-with-rr neighbor 192.168.1.1
set protocols bgp group overlay-with-rr neighbor 192.168.1.2
set protocols bgp group overlay-with-rr neighbor 192.168.1.3
set protocols bgp group overlay-with-rr neighbor 192.168.1.4

set protocols bgp group overlay-with-rr bfd-liveness-detection minimum-interval 1000
set protocols bgp group overlay-with-rr bfd-liveness-detection multiplier 3
set protocols bgp group overlay-with-rr bfd-liveness-detection session-mode automatic

set protocols bgp group overlay-with-rr vpn-apply-export
```

3. Configure IBGP with EVPN on the collapsed spine devices to peer with the route reflector devices, which are identified as IBGP neighbors by their device loopback addresses shown in Figure 68 on page 299. The configuration is the same on all spine devices except you substitute the spine device's loopback IP address for the `local-address` *`device-loopback-addr`* value.

   In this step you also:

   - Enable path maximum transmission unit (MTU) discovery to dynamically determine the MTU size on the network path between the source and the destination, which can help avoid IP fragmentation.

   - Set up BFD for detecting IBGP neighbor failures.

   - Set the `vpn-apply-export` option to ensure that both the VRF and BGP group or neighbor export policies in the BGP configuration are applied (in that order) before the device advertises routes in the VPN routing tables to the other route reflector or spine devices. (See *Distributing VPN Routes* for more information.)

   *All spine devices:*

```
set protocols bgp group overlay-with-rr type internal
set protocols bgp group overlay-with-rr local-address device-loopback-addr

set protocols bgp group overlay-with-rr family evpn signaling

set protocols bgp group overlay-with-rr multipath
set protocols bgp group overlay-with-rr mtu-discovery

set protocols bgp group overlay-with-rr neighbor 192.168.2.1
set protocols bgp group overlay-with-rr neighbor 192.168.2.2
set protocols bgp group overlay-with-rr bfd-liveness-detection minimum-interval 1000
set protocols bgp group overlay-with-rr bfd-liveness-detection multiplier 3
set protocols bgp group overlay-with-rr bfd-liveness-detection session-mode automatic

set protocols bgp group overlay-with-rr vpn-apply-export
```

4. Ensure LLDP is enabled on all interfaces except the management interface (`em0`) on the route reflector cluster and spine devices.

   *All route reflector and spine devices:*

   ```
   set protocols lldp interface all
   set protocols lldp interface em0 disable
   ```

5. Configure EVPN with VXLAN encapsulation in the overlay on the spine devices. The configuration is the same on all spine devices in the collapsed spine fabric.

   In this step:

   - Specify and apply a policy for per-packet load balancing for ECMP in the forwarding table.
   - Configure these EVPN options at the [edit protocols evpn] hierarchy level along with setting VXLAN encapsulation:
     - `default-gateway no-gateway-community`: Advertise the virtual gateway and IRB MAC addresses to the EVPN peer devices so that Ethernet-only edge devices can learn these MAC addresses. You configure `no-gateway-community` in a collapsed spine fabric if the spines use:
       - Anycast IP as the gateway with a common anycast MAC address, or
       - virtual-gateway-address with a VRRP-based MAC address (00:00:5e:00:01:*XX*) (see Step 5 in "Configure EVPN Multihoming and Virtual Networks on the Spine Devices for the ToR Switches" on page 304.
     - `extended-vni-list all` option: Allow all configured VXLAN network identifiers (VNIs) to be part of this EVPN-VXLAN BGP domain. We configure VLANs and VLAN to VNI mappings in a later section.
     - `remote-ip-host-routes`: Enable virtual machine traffic optimization (VMTO). (See "Ingress Virtual Machine Traffic Optimization for EVPN " on page 42 for more information.)

   *All spine devices:*

   ```
   set policy-options policy-statement per-packet-load-balance term 1 then load-balance per-
   packet
   set routing-options forwarding-table export per-packet-load-balance
   ```

```
set protocols evpn encapsulation vxlan
set protocols evpn default-gateway no-gateway-community
set protocols evpn extended-vni-list all
set protocols evpn remote-ip-host-routes
```

6. Configure VTEP, route target, and virtual routing and forwarding (VRF) switch options on the spine devices.

   The configuration is the same on all spine devices except on each device you substitute the device's loopback IP address for the `route-distinguisher` value. This value defines a unique route distinguisher for routes generated by each device.

   The VTEP source interface in the EVPN instance should also match the IBGP local peer address, which is likewise the device loopback IP address.

   *Spine 1:*

```
set switch-options vtep-source-interface lo0.0
set switch-options route-distinguisher 192.168.1.1:3333
set switch-options vrf-target target:10458:0
set switch-options vrf-target auto
```

   *Spine 2:*

```
set switch-options vtep-source-interface lo0.0
set switch-options route-distinguisher 192.168.1.2:3333
set switch-options vrf-target target:10458:0
set switch-options vrf-target auto
```

   *Spine 3:*

```
set switch-options vtep-source-interface lo0.0
set switch-options route-distinguisher 192.168.1.3:3333
set switch-options vrf-target target:10458:0
set switch-options vrf-target auto
```

   *Spine 4:*

```
set switch-options vtep-source-interface lo0.0
set switch-options route-distinguisher 192.168.1.4:3333
set switch-options vrf-target target:10458:0
set switch-options vrf-target auto
```

7. (Required on PTX10000 Series routers only) Enable tunnel termination globally (in other words, on all interfaces) on the device:

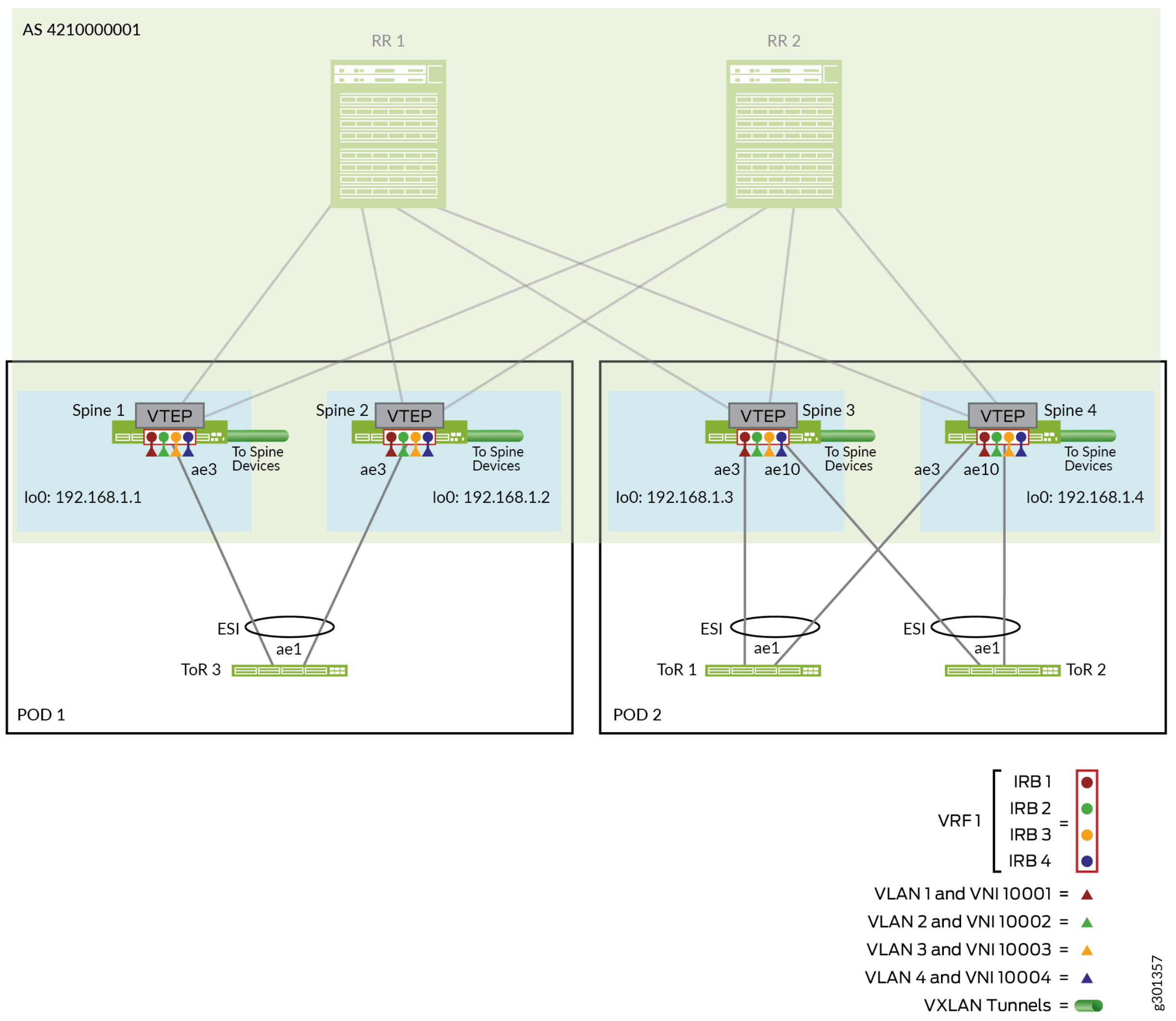
```
set forwarding-options tunnel-termination
```

## Configure EVPN Multihoming and Virtual Networks on the Spine Devices for the ToR Switches

This collapsed spine reference design implements EVPN multihoming as described in "Multihoming an Ethernet-Connected End System Design and Implementation" on page 199, except because the leaf layer functions are collapsed into the spine layer, you configure the ESI-LAGs on the spine devices. You also configure VLANs and Layer 2 and Layer 3 routing functions on the spine devices in a similar way as you would on the leaf devices in an edge-routed bridging (ERB) overlay design. The core collapsed spine configuration implements a Layer 2 stretch by setting the same VLANs (and VLAN-to-VNI mappings) on all of the spine devices in both PODs. EVPN Type 2 routes enable communication between endpoints within and across the PODs.

Figure 69 on page 305 shows the collapsed spine devices in each POD connected with aggregated Ethernet interface links to the multihomed ToR switches in the POD.

**Figure 69: Collapsed Spine Fabric With Multihomed ToR Switches**



For brevity, this section illustrates one aggregated Ethernet link between each spine and each ToR device, with one interface configured on each aggregated Ethernet link from the spine devices to the ToR devices in the POD.

This section covers configuration details only for the spine and ToR devices in POD 2. You can apply a similar configuration with applicable device parameters and interfaces to the spine and ToR devices in POD 1.

The ToR devices include two interfaces in their aggregated Ethernet links, one to each spine device in the POD that form the ESI-LAG for multihoming.

The configuration includes steps to:

- Configure the interfaces.
- Set up the ESI-LAGs for EVPN multihoming.
- Configure Layer 2 and Layer 3 gateway functions, including defining VLANs, the associated IRB interfaces for inter-VLAN routing, and corresponding VLAN-to-VNI mappings.

1. Configure the interfaces and aggregated Ethernet links on the spines (Spine 3 and Spine 4) to the multihomed ToR switches (ToR 1 and ToR 2) in POD 2.

   *Spine 3:*

   ```
   set interfaces xe-0/0/22:0 hold-time up 450000
   set interfaces xe-0/0/22:0 hold-time down 450000
   set interfaces xe-0/0/22:0 ether-options 802.3ad ae3

   set interfaces xe-0/0/23:0 hold-time up 450000
   set interfaces xe-0/0/23:0 hold-time down 450000
   set interfaces xe-0/0/23:0 ether-options 802.3ad ae10
   ```

   *Spine 4:*

   ```
   set interfaces xe-0/0/4:2 hold-time up 450000
   set interfaces xe-0/0/4:2 hold-time down 450000
   set interfaces xe-0/0/4:2 ether-options 802.3ad ae3

   set interfaces xe-0/0/6:1 hold-time up 450000
   set interfaces xe-0/0/6:1 hold-time down 450000
   set interfaces xe-0/0/6:1 ether-options 802.3ad ae10
   ```

2. Configure the ESI-LAGs for EVPN multihoming on the spine devices for the multihomed ToR switches in POD 2. This design uses the same aggregated Ethernet interfaces on the spine devices to the ToR switches, so you use the same configuration on both devices.

   In this reference design, `ae3` connects to ToR 1 and `ae10` connects to ToR 2.

   *Spine 3 and Spine 4:*

   ```
   set interfaces ae3 esi 00:00:00:ff:00:02:00:01:00:03
   set interfaces ae3 esi all-active
   set interfaces ae3 aggregated-ether-options lacp active
   set interfaces ae3 aggregated-ether-options lacp periodic fast
   set interfaces ae3 aggregated-ether-options lacp system-id 00:00:00:99:99:01
   set interfaces ae3 aggregated-ether-options lacp hold-time up 300
   ```

```
set interfaces ae10 esi 00:00:00:ff:00:01:00:01:00:0a
set interfaces ae10 esi all-active
set interfaces ae10 aggregated-ether-options lacp active
set interfaces ae10 aggregated-ether-options lacp periodic fast
set interfaces ae10 aggregated-ether-options lacp system-id 00:00:00:99:99:01
set interfaces ae10 aggregated-ether-options lacp hold-time up 300
```

3. Configure VLANs on the spine devices in POD 2 with `ae3` and `ae10` as VLAN members.

   *Spine 3 and Spine 4:*

```
set interfaces ae3 native-vlan-id 4094
set interfaces ae3 unit 0 family ethernet-switching interface-mode trunk
set interfaces ae3 unit 0 family ethernet-switching vlan members VLAN-1
set interfaces ae3 unit 0 family ethernet-switching vlan members VLAN-2
set interfaces ae3 unit 0 family ethernet-switching vlan members VLAN-3
set interfaces ae3 unit 0 family ethernet-switching vlan members VLAN-4

set interfaces ae10 native-vlan-id 4094
set interfaces ae10 unit 0 family ethernet-switching interface-mode trunk
set interfaces ae10 unit 0 family ethernet-switching vlan members VLAN-1
set interfaces ae10 unit 0 family ethernet-switching vlan members VLAN-2
set interfaces ae10 unit 0 family ethernet-switching vlan members VLAN-3
set interfaces ae10 unit 0 family ethernet-switching vlan members VLAN-4
```

4. Map the VLANs to VNIs for the VXLAN tunnels and associate an IRB interface with each one.

   *Spine 3 and Spine 4:*

```
set vlans VLAN-1 vlan-id 1
set vlans VLAN-1 l3-interface irb.1
set vlans VLAN-1 vxlan vni 100001
set vlans VLAN-2 vlan-id 2
set vlans VLAN-2 l3-interface irb.2
set vlans VLAN-2 vxlan vni 100002
set vlans VLAN-3 vlan-id 3
set vlans VLAN-3 l3-interface irb.3
set vlans VLAN-3 vxlan vni 100003
set vlans VLAN-4 vlan-id 4
set vlans VLAN-4 l3-interface irb.4
set vlans VLAN-4 vxlan vni 100004
```

5. Configure the IRB interfaces for the VLANs (VNIs) on the spine devices in POD 2 with IPv4 and IPv6 dual stack addresses for both the IRB IP address and virtual gateway IP address.

   *Spine 3:*

```
set interfaces irb unit 1 virtual-gateway-accept-data
set interfaces irb unit 1 family inet address 10.0.1.243/24 preferred
set interfaces irb unit 1 family inet address 10.0.1.243/24 virtual-gateway-address 10.0.1.254
set interfaces irb unit 1 family inet6 nd6-stale-time 3600
set interfaces irb unit 1 family inet6 address 2001:db8::10:0:1:243/112 preferred
set interfaces irb unit 1 family inet6 address 2001:db8::10:0:1:243/112 virtual-gateway-
address 2001:db8::10:0:1:254
set interfaces irb unit 1 virtual-gateway-v4-mac 00:00:5e:00:00:04
set interfaces irb unit 1 virtual-gateway-v6-mac 00:00:5e:00:00:04

set interfaces irb unit 2 virtual-gateway-accept-data
set interfaces irb unit 2 family inet address 10.0.2.243/24 preferred
set interfaces irb unit 2 family inet address 10.0.2.243/24 virtual-gateway-address 10.0.2.254
set interfaces irb unit 2 family inet6 nd6-stale-time 3600
set interfaces irb unit 2 family inet6 address 2001:db8::10:0:2:243/112 preferred
set interfaces irb unit 2 family inet6 address 2001:db8::10:0:2:243/112 virtual-gateway-
address 2001:db8::10:0:2:254
set interfaces irb unit 2 virtual-gateway-v4-mac 00:00:5e:00:00:04
set interfaces irb unit 2 virtual-gateway-v6-mac 00:00:5e:00:00:04

set interfaces irb unit 3 virtual-gateway-accept-data
set interfaces irb unit 3 family inet address 10.0.3.243/24 preferred
set interfaces irb unit 3 family inet address 10.0.3.243/24 virtual-gateway-address 10.0.3.254
set interfaces irb unit 3 family inet6 nd6-stale-time 3600
set interfaces irb unit 3 family inet6 address 2001:db8::10:0:3:243/112 preferred
set interfaces irb unit 3 family inet6 address 2001:db8::10:0:3:243/112 virtual-gateway-
address 2001:db8::10:0:3:254
set interfaces irb unit 3 virtual-gateway-v4-mac 00:00:5e:00:00:04
set interfaces irb unit 3 virtual-gateway-v6-mac 00:00:5e:00:00:04

set interfaces irb unit 4 virtual-gateway-accept-data
set interfaces irb unit 4 family inet address 10.0.4.243/24 preferred
set interfaces irb unit 4 family inet address 10.0.4.243/24 virtual-gateway-address 10.0.4.254
set interfaces irb unit 4 family inet6 nd6-stale-time 3600
set interfaces irb unit 4 family inet6 address 2001:db8::10:0:4:243/112 preferred
set interfaces irb unit 4 family inet6 address 2001:db8::10:0:4:243/112 virtual-gateway-
address 2001:db8::10:0:4:254
```

```
set interfaces irb unit 4 virtual-gateway-v4-mac 00:00:5e:00:00:04
set interfaces irb unit 4 virtual-gateway-v6-mac 00:00:5e:00:00:04
```

*Spine 4:*

```
set interfaces irb unit 1 virtual-gateway-accept-data
set interfaces irb unit 1 family inet address 10.0.1.244/24 preferred
set interfaces irb unit 1 family inet address 10.0.1.244/24 virtual-gateway-address 10.0.1.254
set interfaces irb unit 1 family inet6 nd6-stale-time 3600
set interfaces irb unit 1 family inet6 address 2001:db8::10:0:1:244/112 preferred
set interfaces irb unit 1 family inet6 address 2001:db8::10:0:1:244/112 virtual-gateway-
address 2001:db8::10:0:1:254
set interfaces irb unit 1 virtual-gateway-v4-mac 00:00:5e:00:00:04
set interfaces irb unit 1 virtual-gateway-v6-mac 00:00:5e:00:00:04

set interfaces irb unit 2 virtual-gateway-accept-data
set interfaces irb unit 2 family inet address 10.0.2.244/24 preferred
set interfaces irb unit 2 family inet address 10.0.2.244/24 virtual-gateway-address 10.0.2.254
set interfaces irb unit 2 family inet6 nd6-stale-time 3600
set interfaces irb unit 2 family inet6 address 2001:db8::10:0:2:244/112 preferred
set interfaces irb unit 2 family inet6 address 2001:db8::10:0:2:244/112 virtual-gateway-
address 2001:db8::10:0:2:254
set interfaces irb unit 2 virtual-gateway-v4-mac 00:00:5e:00:00:04
set interfaces irb unit 2 virtual-gateway-v6-mac 00:00:5e:00:00:04

set interfaces irb unit 3 virtual-gateway-accept-data
set interfaces irb unit 3 family inet address 10.0.3.244/24 preferred
set interfaces irb unit 3 family inet address 10.0.3.244/24 virtual-gateway-address 10.0.3.254
set interfaces irb unit 3 family inet6 nd6-stale-time 3600
set interfaces irb unit 3 family inet6 address 2001:db8::10:0:3:244/112 preferred
set interfaces irb unit 3 family inet6 address 2001:db8::10:0:3:244/112 virtual-gateway-
address 2001:db8::10:0:3:254
set interfaces irb unit 3 virtual-gateway-v4-mac 00:00:5e:00:00:04
set interfaces irb unit 3 virtual-gateway-v6-mac 00:00:5e:00:00:04

set interfaces irb unit 4 virtual-gateway-accept-data
set interfaces irb unit 4 family inet address 10.0.4.244/24 preferred
set interfaces irb unit 4 family inet address 10.0.4.244/24 virtual-gateway-address 10.0.4.254
set interfaces irb unit 4 family inet6 nd6-stale-time 3600
set interfaces irb unit 4 family inet6 address 2001:db8::10:0:4:244/112 preferred
set interfaces irb unit 4 family inet6 address 2001:db8::10:0:4:244/112 virtual-gateway-
address 2001:db8::10:0:4:254
```

```
set interfaces irb unit 4 virtual-gateway-v4-mac 00:00:5e:00:00:04
set interfaces irb unit 4 virtual-gateway-v6-mac 00:00:5e:00:00:04
```

6. Define the VRF routing instance and corresponding IRB interfaces for EVPN Type 2 routes on each spine device in POD 2 for the configured VLANs (VNIs).

   *Spine 3:*

```
set interfaces lo0 unit 1 family inet

set routing-instances VRF-T2-1 instance-type vrf
set routing-instances VRF-T2-1 interface lo0.1
set routing-instances VRF-T2-1 interface irb.1
set routing-instances VRF-T2-1 interface irb.2
set routing-instances VRF-T2-1 interface irb.3
set routing-instances VRF-T2-1 interface irb.4
set routing-instances VRF-T2-1 route-distinguisher 192.168.1.3:1
set routing-instances VRF-T2-1 vrf-target target:100:1
```

   *Spine 4:*

```
set interfaces lo0 unit 1 family inet

set routing-instances VRF-T2-1 instance-type vrf
set routing-instances VRF-T2-1 interface lo0.1
set routing-instances VRF-T2-1 interface irb.1
set routing-instances VRF-T2-1 interface irb.2
set routing-instances VRF-T2-1 interface irb.3
set routing-instances VRF-T2-1 interface irb.4
set routing-instances VRF-T2-1 route-distinguisher 192.168.1.4:1
set routing-instances VRF-T2-1 vrf-target target:100:1
```

7. Configure the interfaces and aggregated Ethernet links on the multihomed ToR switches (ToR 1 and ToR 2) to the spine devices (Spine 3 and Spine 4) in POD 2. In this step, you:

   - Set the number of aggregated Ethernet interfaces on the switch that you might need (we set 20 here as an example).
   - Configure aggregated Ethernet link `ae1` on each ToR switch to the spine devices in POD 2.
   - Configure LLDP on the interfaces.

*ToR 1:*

```
set chassis aggregated-devices ethernet device-count 20

set interfaces xe-0/0/26 ether-options 802.3ad ae1
set interfaces xe-0/0/27 ether-options 802.3ad ae1

set interfaces ae1 aggregated-ether-options minimum-links 1
set interfaces ae1 aggregated-ether-options lacp active
set interfaces ae1 aggregated-ether-options lacp periodic fast

set protocols lldp interface all
set protocols lldp interface em0 disable
```

*ToR 2:*

```
set chassis aggregated-devices ethernet device-count 20

set interfaces xe-0/0/1 ether-options 802.3ad ae1
set interfaces xe-0/0/27 ether-options 802.3ad ae1

set interfaces ae1 aggregated-ether-options minimum-links 1
set interfaces ae1 aggregated-ether-options lacp active
set interfaces ae1 aggregated-ether-options lacp periodic fast

set protocols lldp interface all
set protocols lldp interface em0 disable
```

8. Configure the VLANs on the ToR switches in POD 2. These match the VLANs you configured in Step 3 on the spine devices in POD 2.

   *ToR 1 and ToR 2:*

```
set vlans VLAN-1 vlan-id 1
set vlans VLAN-2 vlan-id 2
set vlans VLAN-3 vlan-id 3
set vlans VLAN-4 vlan-id 4

set interfaces ae1 native-vlan-id 4094
set interfaces ae1 unit 0 family ethernet-switching interface-mode trunk
set interfaces ae1 unit 0 family ethernet-switching vlan members VLAN-1
set interfaces ae1 unit 0 family ethernet-switching vlan members VLAN-2
set interfaces ae1 unit 0 family ethernet-switching vlan members VLAN-3
set interfaces ae1 unit 0 family ethernet-switching vlan members VLAN-4
```

## Verify Collapsed Spine Fabric Connectivity With Route Reflector Cluster and ToR Devices

This section shows CLI commands you can use to verify connectivity between the collapsed spine devices and the route reflector cluster, and between the collapsed spine devices and the ToR devices.

For brevity, this section includes verifying connectivity on the spine devices using only Spine 3 and Spine 4 in POD 2. You can use the same commands on the spine devices (Spine 1 and Spine 2) in POD 1.

1. Verify connectivity on the aggregated Ethernet links on the route reflector devices toward the four collapsed spine devices. On each route reflector device, `ae`*X* connects to Spine *X*).

   *RR 1:*

```
user@rr-1> show lacp interfaces
Aggregated interface: ae1
    LACP state:       Role   Exp   Def  Dist  Col  Syn  Aggr  Timeout  Activity
      et-0/0/46      Actor    No    No   Yes  Yes  Yes   Yes     Fast    Active
      et-0/0/46    Partner    No    No   Yes  Yes  Yes   Yes     Fast    Active
      et-0/0/62      Actor    No    No   Yes  Yes  Yes   Yes     Fast    Active
      et-0/0/62    Partner    No    No   Yes  Yes  Yes   Yes     Fast    Active
    LACP protocol:        Receive State  Transmit State          Mux State
      et-0/0/46                 Current   Fast periodic Collecting distributing
      et-0/0/62                 Current   Fast periodic Collecting distributing


Aggregated interface: ae2
    LACP state:       Role   Exp   Def  Dist  Col  Syn  Aggr  Timeout  Activity
      et-0/0/9       Actor    No    No   Yes  Yes  Yes   Yes     Fast    Active
      et-0/0/9     Partner    No    No   Yes  Yes  Yes   Yes     Fast    Active
      et-0/0/10      Actor    No    No   Yes  Yes  Yes   Yes     Fast    Active
      et-0/0/10    Partner    No    No   Yes  Yes  Yes   Yes     Fast    Active
    LACP protocol:        Receive State  Transmit State          Mux State
      et-0/0/9                  Current   Fast periodic Collecting distributing
      et-0/0/10                 Current   Fast periodic Collecting distributing
```

```
Aggregated interface: ae3
    LACP state:       Role   Exp   Def  Dist  Col  Syn  Aggr  Timeout  Activity
      et-0/0/49      Actor    No    No   Yes  Yes  Yes   Yes     Fast    Active
      et-0/0/49    Partner    No    No   Yes  Yes  Yes   Yes     Fast    Active
      et-0/0/58      Actor    No    No   Yes  Yes  Yes   Yes     Fast    Active
      et-0/0/58    Partner    No    No   Yes  Yes  Yes   Yes     Fast    Active
    LACP protocol:        Receive State  Transmit State          Mux State
      et-0/0/49                 Current   Fast periodic Collecting distributing
      et-0/0/58                 Current   Fast periodic Collecting distributing


Aggregated interface: ae4
    LACP state:       Role   Exp   Def  Dist  Col  Syn  Aggr  Timeout  Activity
      xe-0/0/34:2    Actor    No    No   Yes  Yes  Yes   Yes     Fast    Active
      xe-0/0/34:2  Partner    No    No   Yes  Yes  Yes   Yes     Fast    Active
      xe-0/0/34:3    Actor    No    No   Yes  Yes  Yes   Yes     Fast    Active
      xe-0/0/34:3  Partner    No    No   Yes  Yes  Yes   Yes     Fast    Active
    LACP protocol:        Receive State  Transmit State          Mux State
      xe-0/0/34:2               Current   Fast periodic Collecting distributing
      xe-0/0/34:3               Current   Fast periodic Collecting distributing
```

*RR 2:*

```
user@rr-2> show lacp interfaces
Aggregated interface: ae1
    LACP state:       Role   Exp   Def  Dist  Col  Syn  Aggr  Timeout  Activity
      et-0/0/18      Actor    No    No   Yes  Yes  Yes   Yes     Fast    Active
      et-0/0/18    Partner    No    No   Yes  Yes  Yes   Yes     Fast    Active
      et-0/0/35      Actor    No    No   Yes  Yes  Yes   Yes     Fast    Active
      et-0/0/35    Partner    No    No   Yes  Yes  Yes   Yes     Fast    Active
    LACP protocol:        Receive State  Transmit State          Mux State
      et-0/0/18                 Current   Fast periodic Collecting distributing
      et-0/0/35                 Current   Fast periodic Collecting distributing


Aggregated interface: ae2
    LACP state:       Role   Exp   Def  Dist  Col  Syn  Aggr  Timeout  Activity
      et-0/0/13      Actor    No    No   Yes  Yes  Yes   Yes     Fast    Active
      et-0/0/13    Partner    No    No   Yes  Yes  Yes   Yes     Fast    Active
      et-0/0/14      Actor    No    No   Yes  Yes  Yes   Yes     Fast    Active
      et-0/0/14    Partner    No    No   Yes  Yes  Yes   Yes     Fast    Active
    LACP protocol:        Receive State  Transmit State          Mux State
```

```
      et-0/0/13                 Current   Fast periodic Collecting distributing
      et-0/0/14                 Current   Fast periodic Collecting distributing



Aggregated interface: ae3
    LACP state:       Role   Exp   Def  Dist  Col  Syn  Aggr  Timeout  Activity
      et-0/0/22      Actor    No    No   Yes  Yes  Yes   Yes     Fast    Active
      et-0/0/22    Partner    No    No   Yes  Yes  Yes   Yes     Fast    Active
      et-0/0/23      Actor    No    No   Yes  Yes  Yes   Yes     Fast    Active
      et-0/0/23    Partner    No    No   Yes  Yes  Yes   Yes     Fast    Active
    LACP protocol:        Receive State  Transmit State          Mux State
      et-0/0/22                 Current   Fast periodic Collecting distributing
      et-0/0/23                 Current   Fast periodic Collecting distributing



Aggregated interface: ae4
    LACP state:       Role   Exp   Def  Dist  Col  Syn  Aggr  Timeout  Activity
      et-0/0/19      Actor    No    No   Yes  Yes  Yes   Yes     Fast    Active
      et-0/0/19    Partner    No    No   Yes  Yes  Yes   Yes     Fast    Active
      et-0/0/20      Actor    No    No   Yes  Yes  Yes   Yes     Fast    Active
      et-0/0/20    Partner    No    No   Yes  Yes  Yes   Yes     Fast    Active
    LACP protocol:        Receive State  Transmit State          Mux State
      et-0/0/19                 Current   Fast periodic Collecting distributing
      et-0/0/20                 Current   Fast periodic Collecting distributing
```

2. Verify connectivity on the aggregated Ethernet links on the spine devices in POD 2 (Spine 3 and Spine 4) toward the route reflector devices. Links `ae1` and `ae2` connect to route reflector devices RR 1 and RR 2, respectively, on both Spine 3 and Spine 4.

   *Spine 3:*

```
user@spine-3> show lacp interfaces ae1
Aggregated interface: ae1
    LACP state:       Role   Exp   Def  Dist  Col  Syn  Aggr  Timeout  Activity
      et-0/0/0       Actor    No    No   Yes  Yes  Yes   Yes     Fast    Active
      et-0/0/0     Partner    No    No   Yes  Yes  Yes   Yes     Fast    Active
      et-0/0/1       Actor    No    No   Yes  Yes  Yes   Yes     Fast    Active
      et-0/0/1     Partner    No    No   Yes  Yes  Yes   Yes     Fast    Active
    LACP protocol:        Receive State  Transmit State          Mux State
      et-0/0/0                  Current   Fast periodic Collecting distributing
      et-0/0/1                  Current   Fast periodic Collecting distributing
```

```
user@spine-3> show lacp interfaces ae2
Aggregated interface: ae2
    LACP state:       Role   Exp   Def  Dist  Col  Syn  Aggr  Timeout  Activity
      et-0/0/7       Actor    No    No   Yes  Yes  Yes   Yes     Fast    Active
      et-0/0/7     Partner    No    No   Yes  Yes  Yes   Yes     Fast    Active
      et-0/0/8       Actor    No    No   Yes  Yes  Yes   Yes     Fast    Active
      et-0/0/8     Partner    No    No   Yes  Yes  Yes   Yes     Fast    Active
    LACP protocol:        Receive State  Transmit State          Mux State
      et-0/0/7                  Current   Fast periodic Collecting distributing
      et-0/0/8                  Current   Fast periodic Collecting distributing
```

*Spine 4:*

```
user@spine-4> show lacp interfaces ae1
Aggregated interface: ae1
    LACP state:       Role   Exp   Def  Dist  Col  Syn  Aggr  Timeout  Activity
      xe-0/0/3:2     Actor    No    No   Yes  Yes  Yes   Yes     Fast    Active
      xe-0/0/3:2   Partner    No    No   Yes  Yes  Yes   Yes     Fast    Active
      xe-0/0/3:3     Actor    No    No   Yes  Yes  Yes   Yes     Fast    Active
      xe-0/0/3:3   Partner    No    No   Yes  Yes  Yes   Yes     Fast    Active
    LACP protocol:        Receive State  Transmit State          Mux State
      xe-0/0/3:2                Current   Fast periodic Collecting distributing
      xe-0/0/3:3                Current   Fast periodic Collecting distributing


user@spine-4> show lacp interfaces ae2
Aggregated interface: ae2
    LACP state:       Role   Exp   Def  Dist  Col  Syn  Aggr  Timeout  Activity
      et-0/0/19      Actor    No    No   Yes  Yes  Yes   Yes     Fast    Active
      et-0/0/19    Partner    No    No   Yes  Yes  Yes   Yes     Fast    Active
      et-0/0/20      Actor    No    No   Yes  Yes  Yes   Yes     Fast    Active
      et-0/0/20    Partner    No    No   Yes  Yes  Yes   Yes     Fast    Active
    LACP protocol:        Receive State  Transmit State          Mux State
      et-0/0/19                 Current   Fast periodic Collecting distributing
      et-0/0/20                 Current   Fast periodic Collecting distributing
```

3. Verify connectivity on the aggregated Ethernet links on the spine devices in POD 2 (Spine 3 and Spine 4) toward the multihomed ToR switches. Links `ae3` and `ae10` connect to ToR 1 and ToR 2, respectively, on both Spine 3 and Spine 4, so this command line filters the output to find link states starting with `ae3`. The output is truncated to show status only for the relevant links.

*Spine 3:*

```
user@spine-3> **show lacp interfaces | find ae3**
Aggregated interface: ae3
    LACP state:       Role   Exp   Def  Dist  Col  Syn  Aggr  Timeout  Activity
      xe-0/0/22:0    Actor    No    No   Yes  Yes  Yes   Yes     Fast    Active
      xe-0/0/22:0  Partner    No    No   Yes  Yes  Yes   Yes     Fast    Active
    LACP protocol:        Receive State  Transmit State          Mux State
      xe-0/0/22:0               Current   Fast periodic Collecting distributing

    LACP hold-timer:  Up, Enabled, Interval: 300 sec
                        Status         Re-Start Cnt   TTE(sec)    Hold Start
    xe-0/0/22:0         Not-Running    NA             NA          NA

...

Aggregated interface: ae10
    LACP state:       Role   Exp   Def  Dist  Col  Syn  Aggr  Timeout  Activity
      xe-0/0/23:0    Actor    No    No   Yes  Yes  Yes   Yes     Fast    Active
      xe-0/0/23:0  Partner    No    No   Yes  Yes  Yes   Yes     Fast    Active
    LACP protocol:        Receive State  Transmit State          Mux State
      xe-0/0/23:0               Current   Fast periodic Collecting distributing

    LACP hold-timer:  Up, Enabled, Interval: 300 sec
                        Status         Re-Start Cnt   TTE(sec)    Hold Start
    xe-0/0/23:0         Not-Running    NA             NA          NA
...
```

*Spine 4:*

```
user@spine-3> **show lacp interfaces | find ae3**
Aggregated interface: ae3
    LACP state:       Role   Exp   Def  Dist  Col  Syn  Aggr  Timeout  Activity
      xe-0/0/4:2     Actor    No    No   Yes  Yes  Yes   Yes     Fast    Active
      xe-0/0/4:2   Partner    No    No   Yes  Yes  Yes   Yes     Fast    Active
    LACP protocol:        Receive State  Transmit State          Mux State
      xe-0/0/4:2                Current   Fast periodic Collecting distributing

    LACP hold-timer:  Up, Enabled, Interval: 300 sec
                        Status         Re-Start Cnt   TTE(sec)    Hold Start
    xe-0/0/4:2          Not-Running    NA             NA          NA
```

```
...

Aggregated interface: ae10
    LACP state:       Role   Exp   Def  Dist  Col  Syn  Aggr  Timeout  Activity
      xe-0/0/6:1     Actor    No    No   Yes  Yes  Yes   Yes     Fast    Active
      xe-0/0/6:1   Partner    No    No   Yes  Yes  Yes   Yes     Fast    Active
    LACP protocol:        Receive State  Transmit State          Mux State
      xe-0/0/6:1                Current   Fast periodic Collecting distributing

    LACP hold-timer:  Up, Enabled, Interval: 300 sec
                        Status         Re-Start Cnt   TTE(sec)     Hold Start
    xe-0/0/6:1          Not-Running    NA             NA           NA

...
```

4. Verify that the spine devices in POD 2 (Spine 3 and Spine 4) detect the route reflector devices and the ToR switches in POD 2 as LLDP neighbors. For the spine to ToR links, this verifies that the ESI member links have been established to the multihomed ToR switches.

   This sample command output is filtered and truncated to show only the relevant aggregated Ethernet links. Comment lines show the columns for the values displayed in the resulting output. See Figure 67 on page 290 again, which shows that both spine switches in POD 2 use `ae1` and `ae2` to link to the route reflector devices, `ae3` to link to ToR1, and `ae10` to link to ToR 2.

   *Spine 3:*

```
user@spine-3> show lldp neighbors | grep ae
#Local Interface   Parent Interface   Chassis Id          Port info          System Name
et-0/0/0           ae1                54:4b:8c:cd:e4:38   et-0/0/58          rr-1
et-0/0/1           ae1                54:4b:8c:cd:e4:38   et-0/0/49          rr-1
et-0/0/7           ae2                c0:bf:a7:ca:53:c0   et-0/0/22          rr-2
et-0/0/8           ae2                c0:bf:a7:ca:53:c0   et-0/0/23          rr-2
et-0/0/22:0        ae3                10:0e:7e:b0:a1:40   xe-0/0/26          tor-1
...
xe-0/0/23:0        ae10               20:d8:0b:14:72:00   xe-0/0/1           tor-2
...
```

*Spine 4:*

```
user@spine-3> show lldp neighbors | grep ae
#Local Interface   Parent Interface    Chassis Id          Port info          System Name
xe-0/0/3:2         ae1                 54:4b:8c:cd:e4:38   xe-0/0/34:2        rr-1
xe-0/0/3:3         ae1                 54:4b:8c:cd:e4:38   xe-0/0/34:3        rr-1
et-0/0/19          ae2                 c0:bf:a7:ca:53:c0   et-0/0/19          rr-2
et-0/0/20          ae2                 c0:bf:a7:ca:53:c0   et-0/0/20          rr-2

xe-0/0/4:2         ae3                 10:0e:7e:b0:a1:40   xe-0/0/27          tor-1
...
xe-0/0/6:1         ae10                20:d8:0b:14:72:00   xe-0/0/27          tor-2


...
```

## Verify Collapsed Spine Fabric BGP Underlay and EVPN-VXLAN Overlay Configuration

This section shows CLI commands you can use to verify the underlay and overlay are working for the collapsed spine devices integrated with the route reflector cluste. Refer to Figure 67 on page 290 and Figure 68 on page 299 again for the configured underlay and overlay parameters.

For brevity, this section includes verifying connectivity on the spine devices using only Spine 3 and Spine 4 in POD 2. You can use the same commands on the spine devices (Spine 1 and Spine 2) in POD 1.

1. Verify on the route reflector devices that the EBGP and IBGP peering is established and traffic paths with the four spine devices are active. This sample command output is filtered to show only the relevant status lines showing the established peering. Comment lines show the columns for the values displayed in the resulting output.

   *RR 1:*

```
user@rr-1> show bgp summary | match Estab
# Peer            AS               InPkt     OutPkt    OutQ   Flaps Last Up/Dwn State|#Active/
Received/Damped...
# underlay BGP peerings
172.16.1.1        4200000011        3758       3767       0       0  1d 4:38:36 Establ
172.16.2.1        4200000012         129        131       0       5       56:59 Establ
172.16.3.1        4200000013        3802       3773       0       0  1d 4:41:03 Establ
172.16.4.1        4200000014        3791       3762       0       0  1d 4:36:06 Establ
```

```
...
# overlay BGP peerings
192.168.1.1      4210000001     980683    4088207       0       0  1d 4:38:32 Establ
192.168.1.2      4210000001      27145     154826       0       5       56:58 Establ
192.168.1.3      4210000001    2696563    2953756       0       0  1d 4:41:02 Establ
192.168.1.4      4210000001    2640667    3000173       0       0  1d 4:36:04 Establ
...
```

*RR 2:*

```
user@rr-2> show bgp summary | match Estab
# Peer           AS                InPkt     OutPkt    OutQ   Flaps Last Up/Dwn State|#Active/
Received/Damped...
# underlay BGP peerings
172.16.5.1       4200000011       3748       3763       0       0  1d 4:37:57 Establ
172.16.6.1       4200000012        131        131       0       5       56:16 Establ
172.16.7.1       4200000013       3796       3765       0       0  1d 4:39:01 Establ
172.16.8.1       4200000014       3788       3756       0       0  1d 4:35:27 Establ
...
# overlay BGP peerings
192.168.1.1      4210000001     980619    4085507       0       0  1d 4:37:55 Establ
192.168.1.2      4210000001      27074     154082       0       5       56:14 Establ
192.168.1.3      4210000001    2695621    2952494       0       0  1d 4:38:59 Establ
192.168.1.4      4210000001    2640070    2998889       0       0  1d 4:35:25 Establ
...
```

2. Verify on the spine devices in POD 2 that the underlay EBGP and overlay IBGP peerings are established. This sample command output is filtered to show only the relevant status lines showing the established peering. Comment lines show the columns for the values displayed in the resulting output.

   *Spine 3:*

```
user@spine-3> show bgp summary | match Estab
# Peer           AS                InPkt     OutPkt    OutQ   Flaps Last Up/Dwn  State|#Active/
Received/Damped...
172.16.3.0       4200000021       3761       3788       0       1  1d 4:35:08  Establ
172.16.7.0       4200000022       3755       3783       0       1  1d 4:33:44  Establ
...
192.168.2.1      4210000001    2942193    2685492       0       1  1d 4:35:06  Establ
192.168.2.2      4210000001    2941362    2685039       0       1  1d 4:33:43  Establ
```

*Spine 4:*

```
user@spine-4> show bgp summary | match Estab
# Peer           AS               InPkt     OutPkt    OutQ   Flaps Last Up/Dwn  State|#Active/
Received/Damped...
172.16.4.0       4200000021        3746       3773       0       0  1d 4:28:12 Establ
172.16.8.0       4200000022        3742       3771       0       0  1d 4:28:12 Establ
...
192.168.2.1      4210000001     2986192    2627487       0       0  1d 4:28:10 Establ
192.168.2.2      4210000001     2985323    2627487       0       0  1d 4:28:10 Establ
```

3. Verify the endpoint destination IP addresses for the remote VTEP interfaces, which are the loopback addresses of the other three spine devices in POD 1 and POD 2 of this collapsed spine topology. We include sample output for Spine 3 in POD 2 here; results are similar on the other spine devices.

   *Spine 3:*

```
user@spine-3> show interfaces vtep | match Remote
    VXLAN Endpoint Type: Remote, VXLAN Endpoint Address: 192.168.1.4, L2 Routing Instance:
default-switch, L3 Routing Instance: default
    VXLAN Endpoint Type: Remote, VXLAN Endpoint Address: 192.168.1.1, L2 Routing Instance:
default-switch, L3 Routing Instance: default
    VXLAN Endpoint Type: Remote, VXLAN Endpoint Address: 192.168.1.2, L2 Routing Instance:
default-switch, L3 Routing Instance: default
```

4. Verify the ESI-LAGs on the spine devices toward the ToR switches. We include sample output here for Spine 3 in POD 2 here; results are similar on the other spine devices.

   *Spine 3:*

```
user@spine-3> show evpn instance extensive
Instance: __default_evpn__
  Route Distinguisher: 192.168.1.3:0
  Number of bridge domains: 0
  Number of neighbors: 1
    Address               MAC    MAC+IP        AD        IM        ES Leaf-label Remote-DCI-
Peer
    192.168.1.4             0         0         0         0         2

Instance: default-switch
  Route Distinguisher: 192.168.1.3:3333
  Encapsulation type: VXLAN
  Duplicate MAC detection threshold: 5
  Duplicate MAC detection window: 180
  MAC database status                     Local  Remote
    MAC advertisements:                       5       9
    MAC+IP advertisements:                   21      21
    Default gateway MAC advertisements:       8       0
  Number of local interfaces: 3 (3 up)
    Interface name  ESI                            Mode             Status     AC-Role
    .local..5       00:00:00:00:00:00:00:00:00:00  single-homed     Up         Root
    ae10.0          00:00:00:ff:00:01:00:01:00:0a  all-active       Up         Root
    ae3.0           00:00:00:ff:00:02:00:01:00:03  all-active       Up         Root
  Number of IRB interfaces: 4 (4 up)
    Interface name  VLAN   VNI    Status  L3 context
    irb.1                  100001  Up     VRF-T2-1
    irb.2                  100002  Up     VRF-T2-1
    irb.3                  100003  Up     VRF-T2-1
    irb.4                  100004  Up     VRF-T2-1
  Number of protect interfaces: 0
  Number of bridge domains: 4
    VLAN  Domain-ID Intfs/up   IRB-intf  Mode            MAC-sync IM-label  v4-SG-sync IM-
core-NH v6-SG-sync IM-core-NH Trans-ID
    1     100001       2  2    irb.1     Extended        Enabled  100001
Disabled              Disabled              100001
    2     100002       2  2    irb.2     Extended        Enabled  100002
Disabled              Disabled              100002
    3     100003       2  2    irb.3     Extended        Enabled  100003
Disabled              Disabled              100003
    4     100004       2  2    irb.4     Extended        Enabled  100004
Disabled              Disabled              100004
  Number of neighbors: 1
    Address               MAC    MAC+IP        AD        IM        ES Leaf-label Remote-DCI-
Peer
    192.168.1.4             9        21         8         4         0
  Number of ethernet segments: 6
    ESI: 00:00:00:ff:00:01:00:01:00:0a
      Status: Resolved by IFL ae10.0
      Local interface: ae10.0, Status: Up/Forwarding
      Number of remote PEs connected: 1
        Remote-PE        MAC-label  Aliasing-label  Mode
        192.168.1.4      0          0               all-active
      DF Election Algorithm: MOD based
      Designated forwarder: 192.168.1.4
      Backup forwarder: 192.168.1.3
      Last designated forwarder update: Apr 09 13:13:20
    ESI: 00:00:00:ff:00:02:00:01:00:03
      Status: Resolved by IFL ae3.0
      Local interface: ae3.0, Status: Up/Forwarding
      Number of remote PEs connected: 1
        Remote-PE        MAC-label  Aliasing-label  Mode
        192.168.1.4      100001     0               all-active
      DF Election Algorithm: MOD based
      Designated forwarder: 192.168.1.4
      Backup forwarder: 192.168.1.3
      Last designated forwarder update: Apr 09 13:13:20
    ESI: 05:fa:ef:80:81:00:01:86:a1:00
      Local interface: irb.1, Status: Up/Forwarding
      Number of remote PEs connected: 1
        Remote-PE        MAC-label  Aliasing-label  Mode
        192.168.1.4      100001     0               all-active
    ESI: 05:fa:ef:80:81:00:01:86:a2:00
      Local interface: irb.2, Status: Up/Forwarding
      Number of remote PEs connected: 1
        Remote-PE        MAC-label  Aliasing-label  Mode
        192.168.1.4      100002     0               all-active
    ESI: 05:fa:ef:80:81:00:01:86:a3:00
      Local interface: irb.3, Status: Up/Forwarding
      Number of remote PEs connected: 1
        Remote-PE        MAC-label  Aliasing-label  Mode
        192.168.1.4      100003     0               all-active
    ESI: 05:fa:ef:80:81:00:01:86:a4:00
      Local interface: irb.4, Status: Up/Forwarding
      Number of remote PEs connected: 1
        Remote-PE        MAC-label  Aliasing-label  Mode
        192.168.1.4      100004     0               all-active
  Router-ID: 192.168.1.3
  SMET Forwarding: Disabled
```

## RELATED DOCUMENTATION

Data Center Fabric Reference Design Overview and Validated Topology | 49

Edge-Routed Bridging Overlay Design and Implementation | 206

Multihoming an IP-Connected End System Design and Implementation | 251

# EVPN Type 2 and Type 5 Route Coexistence Implementation

IN THIS SECTION

By default, EVPN-VXLAN devices import and advertise EVPN Type 2 routes (MAC with IP advertisement routes) for ESI MAC address control plane learning. You can also enable the devices to import and advertise EVPN Type 5 routes (IP prefix routes) in a virtual routing and forwarding (VRF) instance using the `ip-prefix-routes` statement at the `[edit routing-instances name protocols evpn]` hierarchy level. With Type 5 routes enabled, the device will learn how to reach an IP host address from both a Type 2 route (the IP portion) and from a Type 5 route for the same prefix.

When the device learns either type of route for the same destination, it stores both as separate routes, so Type 2 and Type 5 routes might coexist for a VRF. You can configure the VRF with different import and export VRF route targets to prevent importing local Type 5 routes back into the device (essentially preventing coexistence entirely).

However, in some cases, you might want to accept coexisting routes for local or remote destinations. You can also apply routing policies to prevent the device from importing Type 2 or Type 5 host routes for particular prefixes or destinations.

## Type 2 and Type 5 Route Coexistence in a Virtual Routing and Forwarding Instance

IN THIS SECTION

In a large data center with an edge-routed bridging (ERB) overlay fabric, Type 2 and Type 5 route coexistence can strain the device's Packet Forwarding Engine (PFE) next-hop resources. Also, the device performs better in certain cases if it can prefer one type of route over the other. As a result, we provide a Type 2 and Type 5 route coexistence preference feature on supported platforms and releases. With this feature, when the device accepts coexisting Type 2 and Type 5 routes, the device applies a route preference algorithm by default. According to the preference algorithm, the device assigns only one type of route for the same destination prefix as the active route.

See EVPN Type 2 and Type 5 Route Coexistence with EVPN-VXLAN in the EVPN User Guide for more on this feature and the default preference algorithm.

## Type 2 and Type 5 Route Coexistence Preference Algorithm

When you configure the device to accept coexisting Type 2 and Type 5 routes for a VRF, the device applies the following preference algorithm on imported routes:

- If the device learns a Type 2 route for a destination and doesn't have a matching prefix Type 5 route, it installs the Type 2 route.
- If the device learns a Type 5 route for a *local* ESI Type 2 route (a local host route) for the same prefix, it installs the Type 2 route.
- If the device learns a Type 5 route and has a matching Type 2 entry for a non-local interface, it installs the Type 5 route instead (and it deletes the Type 2 entry).

Essentially, if Type 2 and Type 5 routes coexist for the same prefix, the algorithm prefers Type 2 routes for local interfaces and prefers Type 5 routes for all other destinations. The device deletes non-local destination Type 2 route entries from the PFE when it installs a Type 5 route for the same prefix, saving next-hop table space in the PFE.

## Limitations with Type 2 and Type 5 Route Coexistence Feature

The following sections describe some routing behavior limitations or constraints in a routing instance with the Type 2 and Type 5 route coexistence feature.

### Routing Over an IRB Interface for BGP with EBGP

EBGP protocol is designed to establish peering between directly-connected interface addresses, so the EBGP message default time-to-live (TTL) is 1 (which corresponds to one next hop). In EBGP configurations with Type 2 and Type 5 route coexistence, the device prefers Type 5 routes for remote destination hosts. When the device routes a packet over an IRB interface for a Type 5 route tunnel, the routing might require more hops than traffic bridged or routed locally using Type 2 routes.

To ensure successful routing over IRB interfaces in EBGP configurations with Type 2 and Type 5 route coexistence, you must set the `multihop` option with a TTL value of 2 or more in the EBGP peer group configuration, as follows:

```
set routing-instances VRF-instance protocols bgp group BGP-peer-group-name multihop ttl 2
```

### Routing Over an IRB Interface with IS-IS or OSPF

To ensure successful routing over IRB interfaces for destination hosts where the device resolves those routes using IS-IS or OSPF, you can't allow Type 2 and Type 5 routes to coexist. ISIS and OSPF are link state protocols that expect peers to be directly connected. The device should not route IS-IS or OSPF control packets and should use Type 2 routes. When Type 2 and Type 5 routes coexist, the device must prefer the Type 2 route. As a result, in this case, define and apply a routing policy to avoid importing Type 5 routes for those host routes.

See the sample policies in:

- Step 6 in "Configure Type 2 and Type 5 Routes to Coexist" on page 325.
- EVPN Type 2 and Type 5 Route Coexistence with EVPN-VXLAN in the EVPN User Guide

These policies filter and don't import routes for specific host addresses or prefixes.

### DHCP Relay Recommendation with Type 2 and Type 5 Route Coexistence

In a VRF instance with DHCP relay enabled and Type 2 and Type 5 routes coexist, you should disable snooping in the DHCP relay configuration, as follows:

```
set routing-instances VRF-instance forwarding-options dhcp-relay no-snoop
```

## Configure Type 2 and Type 5 Routes to Coexist

To configure a VRF to have coexisting Type 2 and Type 5 routes, enable Type 5 routing (IP prefix routes) and ensure the device uses the same import and export VRF route targets in the routing instance. You also apply a routing policy in the instance for the host routes that you want the device to import or advertise. See EVPN Type 2 and Type 5 Route Coexistence with EVPN-VXLAN for more on policy options you might want to configure with this feature.

The following example configuration enables Type 2 and Type 5 route coexistence on a leaf device in the ERB overlay fabric introduced in "Edge-Routed Bridging Overlay Design and Implementation" on page 206. You can use the same general configuration from that ERB reference design example. The configuration here corresponds to the Type 5 route configuration on Leaf 10 in VRF_3. Refer to the steps there where we set the host routes export routing policy and enable Type 5 routes in routing instance VRF_3.

1. Configure a policy called EXPORT_HOST_ROUTES to match on and accept /32 and /128 host routes (these correspond to all host routes). Include direct routes and static routes in the policy too.

   ```
   set policy-options policy-statement EXPORT_HOST_ROUTES term TERM_1 from protocol evpn
   set policy-options policy-statement EXPORT_HOST_ROUTES term TERM_1 from route-filter
   0.0.0.0/0 prefix-length-range /32-/32
   set policy-options policy-statement EXPORT_HOST_ROUTES term TERM_1 then accept
   set policy-options policy-statement EXPORT_HOST_ROUTES term TERM_2 from protocol direct
   set policy-options policy-statement EXPORT_HOST_ROUTES term TERM_2 then accept
   set policy-options policy-statement EXPORT_HOST_ROUTES term TERM_3 from protocol static
   set policy-options policy-statement EXPORT_HOST_ROUTES term TERM_3 then accept
   set policy-options policy-statement EXPORT_HOST_ROUTES TERM_4 from family inet6
   set policy-options policy-statement EXPORT_HOST_ROUTES TERM_4 from protocol evpn
   set policy-options policy-statement EXPORT_HOST_ROUTES TERM_4 from route-filter 0::0/0 prefix-
   length-range /128-/128
   set policy-options policy-statement EXPORT_HOST_ROUTES TERM_4 then accept
   ```

2. Configure a tenant VRF routing instance VRF_3 for VNI_50000. (In the configuration in "Edge-Routed Bridging Overlay Design and Implementation" on page 206, you can apply a similar step for VNI_60000 and irb.600 in VRF_3, and VNI_70000 and VNI_80000 in VRF_4.)

   ```
   set routing-instances VRF_3 instance-type vrf
   set routing-instances VRF_3 interface irb.500
   set routing-instances VRF_3 interface lo0.3
   set routing-instances VRF_3 route-distinguisher 192.168.1.10:500
   set routing-instances VRF-3 routing-options multipath
   set routing-instances VRF-3 routing-options rib VRF-3.inet6.0 multipath
   ```

3. Enable Type 5 routes (IP prefix routes ) in VRF_3. Apply the EXPORT_HOST_ROUTES routing policy from step 1. Make sure that Type 2 and Type 5 routes coexist by setting the import and export route

targets to the same value. Supported devices apply the coexistence route preference algorithm when importing and installing routes.

```
set routing-instances VRF_3 protocols evpn ip-prefix-routes advertise direct-nexthop
set routing-instances VRF_3 protocols evpn ip-prefix-routes encapsulation vxlan
set routing-instances VRF_3 protocols evpn ip-prefix-routes export EXPORT_HOST_ROUTES
set routing-instances VRF_3 vrf-target import target:62273:50000
set routing-instances VRF_3 vrf-target export target:62273:50000
```

**NOTE**: You can configure the `vrf-target` *`route-target-value`* statement at the `[edit routing-instances` *`name`*`]` hierarchy without specifying separate `import` and `export` target options. In that case, if you've enabled Type 5 routes in the instance, the configuration applies the same route target value for import or export (advertisement) actions, which enables Type 2 and Type 5 route coexistence. Here we explicitly set the same route target using `import` and `export` options to show clearly that Type 2 and Type 5 routes coexist.

4. (Required on ACX7100 routers only) Set the `reject-asymmetric-vni` option in the VRF routing instances where you enable Type 5 IP prefix routes. This option configures the device to reject EVPN Type 5 route advertisements with asymmetric VNIs—the device doesn't accept traffic from the control plane with a received VNI that doesn't match the locally configured VNI. We support only symmetric VNI routes on these devices.

```
set routing-instances VRF_3 protocols evpn ip-prefix-routes reject-asymmetric-vni
```

5. (For a VRF instance that learns host IP addresses using EBGP) To support routing over IRB interfaces with Type 2 and Type 5 coexistence for hosts that use EBGP, you must set the `multihop` option with a TTL value of 2 or more in the EBGP peer group configuration. (See "Routing Over an IRB Interface for BGP with EBGP" on page 324 for more on this use case.)

   For example:

```
set routing-instances VRF_3 protocols bgp group EBGP-peer-group-name multihop ttl 2
```

6. (For a VRF instance that learns host IP addresses using IS-IS or OSPF) To support routing over IRB interfaces for hosts that use IS-IS or OSPF, ensure the instance always prefers Type 2 routes for those hosts. In other words, prevent Type 2 and Type 5 route coexistence for only those host IP addresses. (See "Routing Over an IRB Interface with IS-IS or OSPF" on page 325 for more on this limitation.)

To do this, define and apply a routing policy in the VRF instance that avoids importing Type 5 routes for those host IP addresses. For example, in the configuration for VRF_3, to filter out Type 5 routes for a host with IP address 10.1.4.106:

```
set policy-options policy-statement RejectType5HostRoutes term 1 from route-filter
10.1.4.106/32 exact
set policy-options policy-statement RejectType5HostRoutes term 1 then reject
set policy-options policy-statement RejectType5HostRoutes term 2 then accept
set routing-instances VRF_3 protocols evpn ip-prefix-routes import RejectType5HostRoutes
```

## Verify Type 2 and Type 5 Route Coexistence

To verify the coexistence preference algorithm stores only the preferred routes with Type 2 and Type 5 route coexistence, use the following commands.

These sample commands use the ERB overlay fabric configured in "Edge-Routed Bridging Overlay Design and Implementation" on page 206 on Leaf 10 with:

- MAC-VRF instance: MAC-VRF-1
- Tenant VRF: VRF_3
- A local host with IP address 10.1.4.101
- A remote host with IP address 10.1.4.102

These commands check routes to a remote host with IP address 10.1.4.102 on an Ethernet segment on another leaf device in the fabric. Leaf 10 has a host with IP address 10.1.4.101.

1. Enter the `show arp no-resolve` command to check that the remote host with IP address 10.1.4.102 doesn't appear in the ARP table. The device is saving next hop space in the PFE by avoiding installing both Type 2 and Type 5 routes for the same remote destination.

```
user@Leaf-10> show arp no-resolve interface irb.500

MAC Address       Address         Interface              Flags
ba:41:5f:01:e2:01 10.1.4.101      irb.500[ ae3.0 ]       permanent remote
fc:96:43:eb:63:33 10.1.4.2        irb.500[ .local..1408 ]  permanent remote
00:31:46:7b:e1:18 10.1.4.3        irb.500[ .local..1408 ]  permanent remote
ec:3e:f7:89:15:1a 10.1.4.4        irb.500[ .local..1408 ]  permanent remote
```

2. Enter the `show ethernet-switching mac-ip-table` command to see routes for the remote host. When Type 2 and Type 5 routes coexist, the `RTS` flag in the `Flags` output field means the device skipped adding a Type 2 route in favor of a Type 5 route with a matching prefix.

   Here, the output includes the `RTS` flag for the remote host with IP address 10.1.4.102, so the Type 5 route is the preferred route.

```
user@Leaf-10> show ethernet-switching mac-ip-table instance MAC-VRF-1 vlan-name VNI_50000
ba:42:5f:01:e2:01

MAC IP flags  (S - Static, D - Dynamic, L - Local , R - Remote, Lp - Local Proxy,
               Rp - Remote Proxy, K - Kernel, RT - Dest Route, (N)AD - (Not) Advt to remote,
               RE - Re-ARP/ND, RO - Router, OV - Override, Ur - Unresolved,
               RTS - Dest Route Skipped, RGw - Remote Gateway, FU - Fast Update)
 Routing instance : MAC-VRF-1
 Bridging domain : VNI_50000
   IP                            MAC                  Flags              Logical
Active
   address                       address                                 Interface
source
   10.1.4.102                    ba:42:5f:01:e2:01    DR,K,RTS           esi.234220
00:00:00:ff:00:02:00:01:00:03
   2001:db8::10:1:4:102          ba:42:5f:01:e2:01    DR,K,RTS           esi.234220
00:00:00:ff:00:02:00:01:00:03
```

3. Enter the `show route forwarding-table ... extensive` command to see Type 5 route forwarding entries for the tenant VRF instance VRF_3.

   The `Flags` field includes the `VxLAN Local` flag when the route is a Type 5 local route that the device preferred over a learned Type 2 route for the same destination.

```
user@Leaf-10> show route forwarding-table destination 10.1.4.102 vpn VRF_3 extensive

Routing table: VRF_3.inet [Index: 1597]
Internet:

Destination:  10.1.4.102/32
  Route type: user
  Route reference: 0                    Route interface-index: 0
  Multicast RPF nh index: 0
  P2mpidx: 0
  Flags: rt nh decoupled, VxLAN Local
  Next-hop type: unilist                Index: 13236    Reference: 1
  Next-hop type: composite              Index: 15829    Reference: 1
 .
 .
 .
```

4. Enter the `show route ... extensive` command to check that the (preferred) Type 5 route for the remote host is in the routing table. The `State` field in the output with the extensive option includes `VxlanLocalRT` for Type 5 routes. This is similar to the forwarding table output in the previous step where the `VXLAN Local` flag indicate the device preferred and stored the Type 5 local route instead of the Type 2 route.

```
user@Leaf-10> show route 10.1.4.102 table VRF_3.inet.0 extensive
VRF_3.inet.0: 28 destinations, 68 routes (28 active, 0 holddown, 0 hidden)
10.1.4.102/32 (3 entries, 1 announced)
        State: <CalcForwarding>
TSI:
KRT in-kernel 10.1.4.102/32 -> -> {list:composite(15829), composite(15828)}
        @EVPN   Preference: 170/-101
                Next hop type: Indirect, Next hop index: 0
                Address: 0x286ccd9c
                       .
                       .
                       .
                Indirect next hop: 0x1fc89b08 15827 INH Session ID: 4104
                State: <Active Int Ext VxlanLocalRT>
                Age: 31:22      Metric2: 0
                Validation State: unverified
 .
 .
 .
```

## EVPN Type 2 and Type 5 Route Coexistence Preference Feature — Release History

Table 10 on page 331 provides a history of the feature in this section and its support within this reference design.

**Table 10: EVPN Type 2 and Type 5 Route Coexistence Implementation– Release History**

| Release | Description |
|---|---|
| 21.4R2 | MX Series, QFX5110, QFX5120, and QFX10000 Series devices running Junos OS Release 21.4R2 and later releases in ERB overlay reference architectures. |
| 21.4R2-EVO | ACX7100-48L, PTX10001, PTX10004, PTX10008, PTX10016, and QFX5130-32CD devices running Junos OS Evolved Release 21.4R2 and later releases in ERB overlay reference architectures. |

# Data Center Interconnect Design and Implementation Using Type 5 Routes

**IN THIS SECTION**

## Data Center Interconnect Using EVPN Type 5 Routes

**IN THIS SECTION**

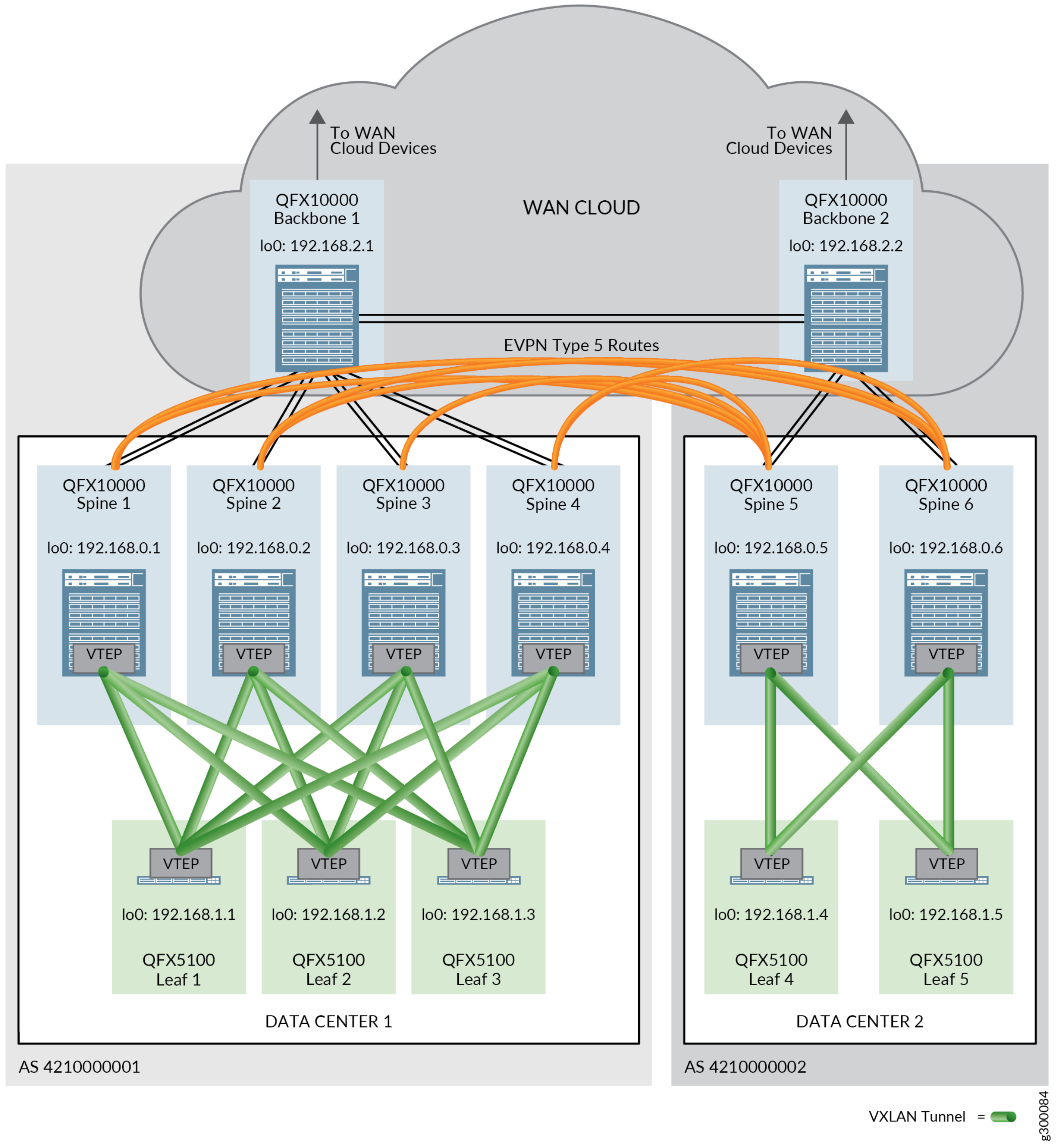
EVPN Type 5 routes, also known as IP prefix routes, are used in a DCI context to pass traffic between data centers that are using different IP address subnetting schemes.

In this reference architecture, EVPN Type 5 routes are exchanged between spine devices in different data centers to allow for the passing of traffic between data centers.

Physical connectivity between the data centers is required before EVPN Type 5 messages can be sent across data centers. This physical connectivity is provided by backbone devices in a WAN cloud. A backbone device is connected to each spine device in a single data center and participates in the overlay IBGP and underlay EBGP sessions. EBGP also runs in a separate BGP group to connect the backbone devices to each other; EVPN signaling is enabled in this BGP group.

Figure 70 on page 333 shows two data centers using EVPN Type 5 routes for DCI.

**Figure 70: DCI Using EVPN Type 5 Routes Topology Overview**



For additional information on EVPN Type 5 routes, see EVPN Type-5 Route with VXLAN encapsulation for EVPN-VXLAN.

All procedures in this section assume that EVPN Type 2 routes are successfully being passed in the data centers. See "Centrally-Routed Bridging Overlay Design and Implementation" on page 142 for setup instructions.

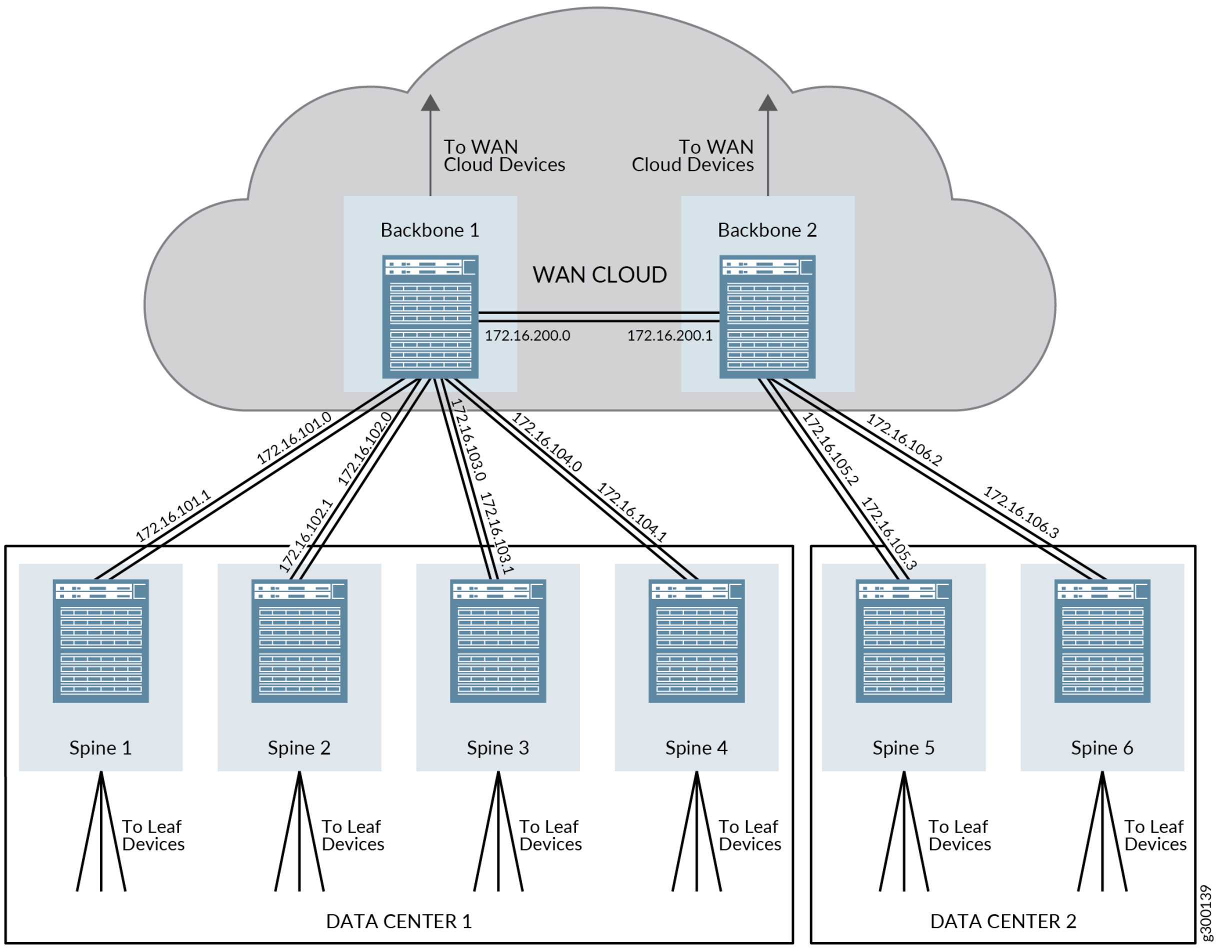
This section covers the processes for configuring a DCI using EVPN Type 5 routes, and includes the following procedures:

## Configuring Backbone Device Interfaces

The backbone devices in this architecture are part of the WAN cloud and must provide connectivity both to the spine devices in each data center as well as to the other backbone device. This connectivity must be established before EVPN Type 5 routes can be exchanged between spine devices in different data centers.

Figure 71 on page 334 provides an overview of the IP addresses that are configured in these steps.

**Figure 71: IP Address Summary for Backbone and Spine Devices**



To configure the spine device and backbone device interfaces:

Set up the interfaces and assign IP addresses:

- (Aggregated Ethernet interfaces) Configure the aggregated Ethernet interfaces on the spine device switches in Data Centers 1 and 2 and on the backbone devices.

  This step shows the assignment of the IP address to the aggregated Ethernet interfaces only. For complete step-by-step instructions on creating aggregated Ethernet interfaces, see Configuring Link Aggregation.

  *Spine Device 1 in Data Center 1*:

  ```
  set interfaces ae3 unit 0 family inet address 172.16.101.1/31
  ```

  *Spine Device 2 in Data Center 1*:

  ```
  set interfaces ae3 unit 0 family inet address 172.16.102.1/31
  ```

  *Spine Device 3 in Data Center 1*:

  ```
  set interfaces ae3 unit 0 family inet address 172.16.103.1/31
  ```

  *Spine Device 4 in Data Center 1*:

  ```
  set interfaces ae3 unit 0 family inet address 172.16.104.1/31
  ```

  *Spine Device 5 in Data Center 2*:

  ```
  set interfaces ae4 unit 0 family inet address 172.16.105.3/31
  ```

  *Spine Device 6 in Data Center 2*:

  ```
  set interfaces ae4 unit 0 family inet address 172.16.106.3/31
  ```

  *Backbone Device 1*:

  ```
  set interfaces ae1 unit 0 family inet address 172.16.101.0/31
  set interfaces ae2 unit 0 family inet address 172.16.102.0/31
  set interfaces ae3 unit 0 family inet address 172.16.103.0/31
  set interfaces ae4 unit 0 family inet address 172.16.104.0/31
  set interfaces ae200 unit 0 family inet address 172.16.200.0/31
  ```

*Backbone Device 2*:

```
set interfaces ae5 unit 0 family inet address 172.16.105.2/31
set interfaces ae6 unit 0 family inet address 172.16.106.2/31
set interfaces ae200 unit 0 family inet address 172.16.200.1/31
```

- (Standalone interfaces that are not included in aggregated Ethernet interfaces) See Configuring the Interface Address.

## Enabling EBGP as the Underlay Network Routing Protocol Between the Spine Devices and the Backbone Devices

EBGP is used as the routing protocol of the underlay network in this reference design. The backbone devices must participate in EBGP with the spine devices to support underlay connectivity.

The process for enabling EBGP on the spine and leaf devices is covered in the "IP Fabric Underlay Network Design and Implementation" on page 80 section of this guide. This procedure assumes EBGP has already been enabled on the spine and leaf devices, although some EBGP configuration on the spine devices needs to be updated to support backbone devices and is therefore included in these steps.

EBGP works in this reference design by assigning each leaf, spine, and backbone device into it’s own unique 32-bit autonomous system (AS) number.

Figure 72 on page 337 shows an overview of the EBGP topology for the spine and backbone devices when backbone devices are included in the reference design.

**Figure 72: EBGP Topology with Backbone Devices**

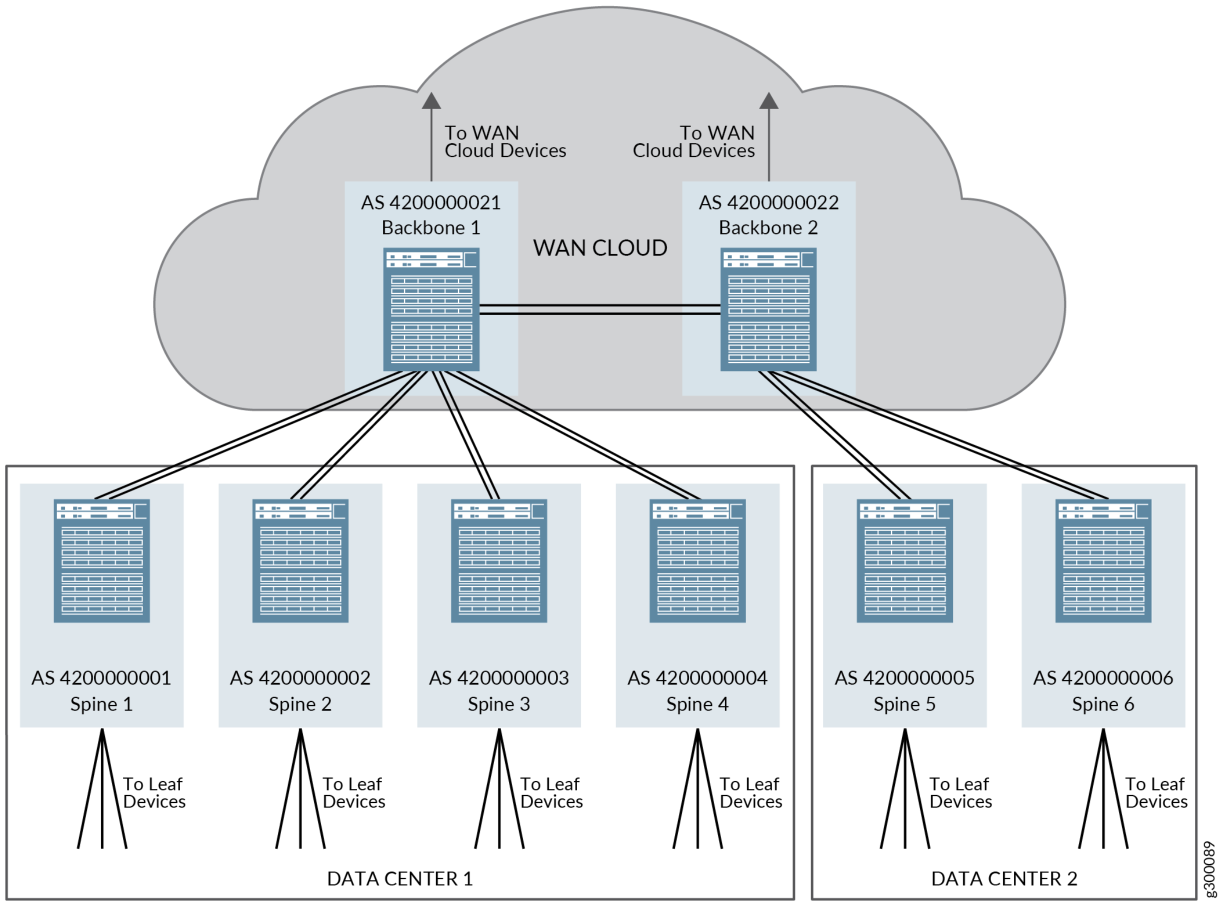
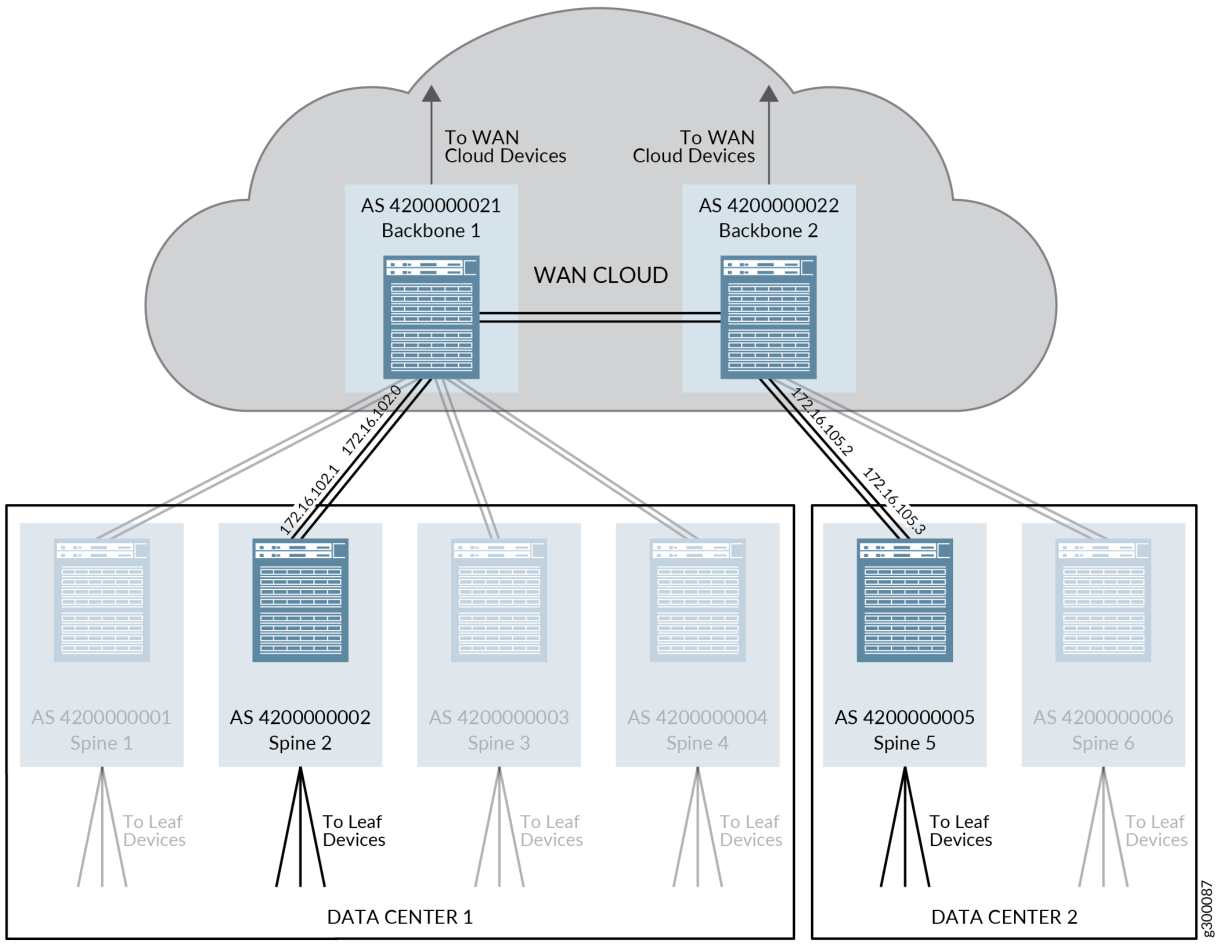


Figure 73 on page 338 illustrates the EBGP protocol parameters that are configured in this procedure. Repeat this process for the other devices in the topology to enable EBGP on the remaining devices.

**Figure 73: EBGP Configuration in a Backbone Topology**



To enable EBGP to support the underlay network in this reference design:

1. Create and name the BGP peer group. EBGP is enabled as part of this step.

   *All Spine and Backbone Devices*:

   ```
   set protocols bgp group UNDERLAY-BGP type external
   ```

2. Configure the ASN for each device in the underlay.

   In this reference design, every device is assigned a unique ASN in the underlay network. The ASN for EBGP in the underlay network is configured at the BGP peer group level using the `local-as` statement because the system ASN setting is used for MP-IBGP signaling in the overlay network.

*Spine Device 2 in Data Center 1 Example*:

```
set protocols bgp group UNDERLAY-BGP local-as 4200000002
```

*Spine Device 5 in Data Center 2 Example*:

```
set protocols bgp group UNDERLAY-BGP local-as 4200000005
```

*Backbone Device 1*:

```
set protocols bgp group UNDERLAY-BGP local-as 4200000021
```

*Backbone Device 2*:

```
set protocols bgp group UNDERLAY-BGP local-as 4200000022
```

3. Configure BGP peers by specifying the ASN of each BGP peer in the underlay network on each spine and backbone device.

   In this reference design, the backbone devices peer with every spine device in the connected data center and the other backbone device.

   The spine devices peer with the backbone device that connects them into the WAN cloud.

   *Spine Device 2 in Data Center 1 Example*:

```
set protocols bgp group UNDERLAY-BGP neighbor 172.16.102.0 peer-as 4200000021
```

*Spine Device 5 in Data Center 2 Example*:

```
set protocols bgp group UNDERLAY-BGP neighbor 172.16.105.2 peer-as 4200000022
```

*Backbone Device 1*:

```
set protocols bgp group UNDERLAY-BGP neighbor 172.16.101.1 peer-as 4200000001
set protocols bgp group UNDERLAY-BGP neighbor 172.16.102.1 peer-as 4200000002
set protocols bgp group UNDERLAY-BGP neighbor 172.16.103.1 peer-as 4200000003
set protocols bgp group UNDERLAY-BGP neighbor 172.16.104.1 peer-as 4200000004
```

*Backbone Device 2*:

```
set protocols bgp group UNDERLAY-BGP neighbor 172.16.105.3 peer-as 4200000005
set protocols bgp group UNDERLAY-BGP neighbor 172.16.106.3 peer-as 4200000006
```

4. Create a routing policy that identifies and includes the loopback interface in EBGP routing table updates and apply it.

   This export routing policy is applied and is used to advertise loopback interface reachability to all devices in the IP Fabric in the overlay network.

   *Each Spine Device and Backbone Device*:

```
set policy-options policy-statement underlay-clos-export term loopback from interface lo0.0
set policy-options policy-statement underlay-clos-export term loopback then accept
set protocols bgp group UNDERLAY-BGP export underlay-clos-export
```

5. Enable multipath to ensure all routes are installed and shared in the forwarding table.

   *Each Spine Device and Backbone Device*:

```
set protocols bgp group UNDERLAY-BGP multipath multiple-as
```

## Enabling IBGP for the Overlay Network on the Backbone Device

The backbone devices must run IBGP to have overlay network connectivity and be able to support DCI using EVPN Type 5 routes.

Figure 74 on page 341 shows the IBGP configuration of the validated reference design when backbone devices are included in the topology. In the validated reference design, all spine and leaf devices in the same data center are assigned into the same autonomous system. The backbone devices are assigned into the same autonomous system as the spine and leaf devices of the data center that is using the backbone device as the entry point into the WAN cloud.

**Figure 74: IBGP Overview with Backbone Devices**

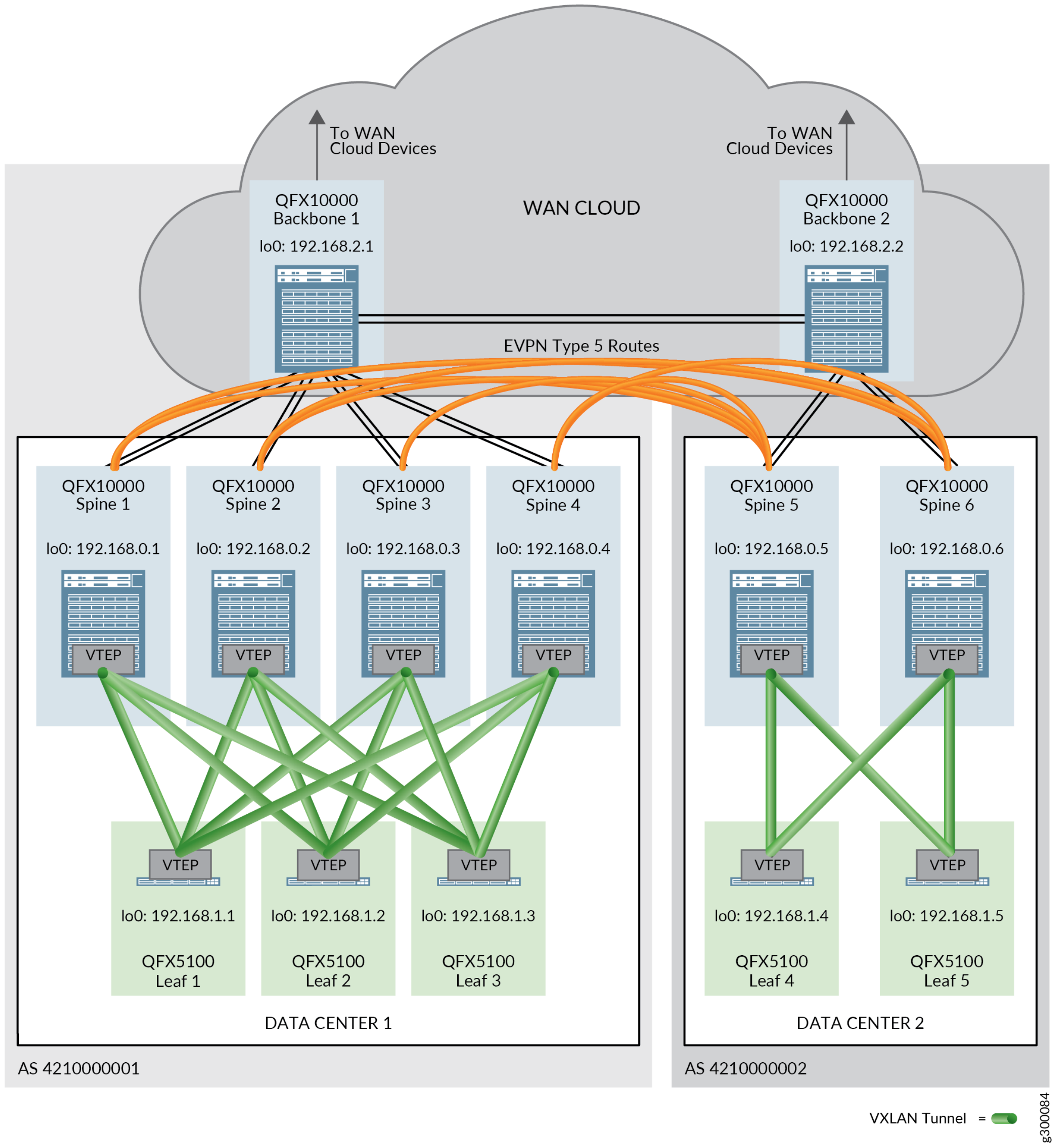
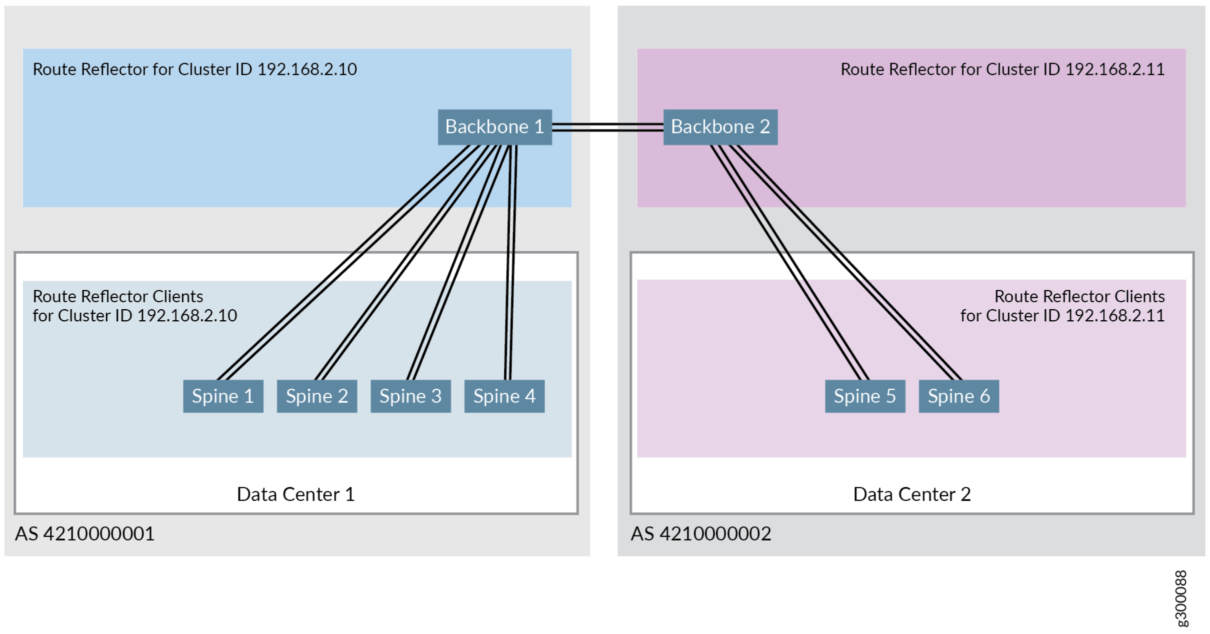


Figure 75 on page 342 illustrates the route reflector configuration in the validated reference design. One route reflector cluster—cluster ID *192.168.2.10*—includes backbone device 1 as the route reflector and all spine devices in data center 1 as route reflector clients. Another route reflector cluster—cluster ID *192.168.2.11*—includes backbone device 2 as the route reflector and all spine devices in data center 2 as route reflector clients.

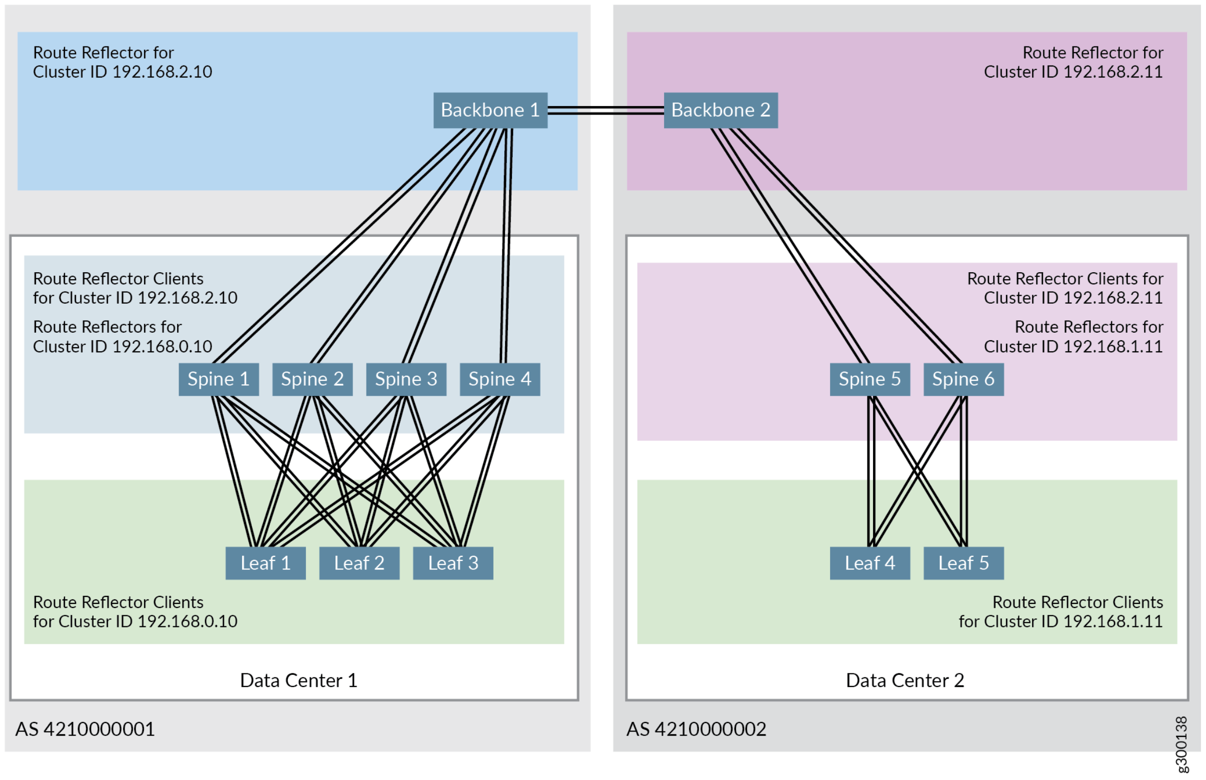
**Figure 75: IBGP Route Reflector Topology**


Route Reflector for Cluster ID 192.168.2.10
Backbone 1
Route Reflector for Cluster ID 192.168.2.11
Backbone 2
Route Reflector Clients for Cluster ID 192.168.2.10
Spine 1
Spine 2
Spine 3
Spine 4
Route Reflector Clients for Cluster ID 192.168.2.11
Spine 5
Spine 6
Data Center 1
Data Center 2
AS 4210000001
AS 4210000002
g300088

The validated reference design supports multiple hierarchical route reflectors, where one cluster includes backbone devices acting as route reflectors for the spine device clients and another cluster includes spine devices acting as route reflectors for leaf device clients. To see the configuration steps for configuring the other route reflector, see "Configure IBGP for the Overlay" on page 96.

Figure 76 on page 343 shows the full hierarchical route reflector topology when two data centers are connected:

**Figure 76: Hierarchical IBGP Route Reflector Topology**



For more information on BGP route reflectors, see Understanding BGP Route Reflectors.

This procedure assumes IBGP has been enabled for the spine and leaf devices as detailed in "Configure IBGP for the Overlay" on page 96. The spine device configurations are included in this procedure to illustrate their relationships to the backbone devices.

To setup IBGP connectivity for the backbone devices:

1. Configure an AS number for overlay IBGP. All leaf and spine devices in the same data center are configured into the same AS. The backbone devices are configured into the same AS as the spine and leaf devices in the data centers using the backbone device as the entry point into the WAN cloud.

   *Backbone Device 1 and All Spine and Leaf Devices in Data Center 1*:

   ```
   set routing-options autonomous-system 4210000001
   ```

   *Backbone Device 2 and All Spine and Leaf Devices in Data Center 2*:

   ```
   set routing-options autonomous-system 4210000002
   ```

2. Configure IBGP using EVPN signaling on the backbone devices. Form the route reflector clusters (cluster IDs 192.168.2.10 and 192.168.2.11) and configure BGP multipath and MTU Discovery.

   *Backbone Device 1*:

```
set protocols bgp group OVERLAY-BGP type internal
set protocols bgp group OVERLAY-BGP local-address 192.168.2.1
set protocols bgp group OVERLAY-BGP family evpn signaling
set protocols bgp group OVERLAY-BGP cluster 192.168.2.10
set protocols bgp group OVERLAY-BGP mtu-discovery
set protocols bgp group OVERLAY-BGP multipath
set protocols bgp group OVERLAY-BGP neighbor 192.168.0.1
set protocols bgp group OVERLAY-BGP neighbor 192.168.0.2
set protocols bgp group OVERLAY-BGP neighbor 192.168.0.3
set protocols bgp group OVERLAY-BGP neighbor 192.168.0.4
```

   *Backbone Device 2*:

```
set protocols bgp group OVERLAY-BGP type internal
set protocols bgp group OVERLAY-BGP local-address 192.168.2.2
set protocols bgp group OVERLAY-BGP family evpn signaling
set protocols bgp group OVERLAY-BGP cluster 192.168.2.11
set protocols bgp group OVERLAY-BGP mtu-discovery
set protocols bgp group OVERLAY-BGP multipath
set protocols bgp group OVERLAY-BGP neighbor 192.168.0.5
set protocols bgp group OVERLAY-BGP neighbor 192.168.0.6
```

3. Configure IBGP using EVPN signaling on the spine devices. Enable BGP multipath and MTU Discovery.

   *Spine Device 2 in Data Center 1 Example*:

```
set protocols bgp group OVERLAY-BGP-TO-RR type internal
set protocols bgp group OVERLAY-BGP-TO-RR local-address 192.168.0.2
set protocols bgp group OVERLAY-BGP-TO-RR mtu-discovery
set protocols bgp group OVERLAY-BGP-TO-RR family evpn signaling
set protocols bgp group OVERLAY-BGP-TO-RR vpn-apply-export
set protocols bgp group OVERLAY-BGP-TO-RR multipath
set protocols bgp group OVERLAY-BGP-TO-RR neighbor 192.168.2.1
```

*Spine Device 5 in Data Center 2 Example*:

```
set protocols bgp group OVERLAY-BGP-TO-RR type internal
set protocols bgp group OVERLAY-BGP-TO-RR local-address 192.168.0.5
set protocols bgp group OVERLAY-BGP-TO-RR mtu-discovery
set protocols bgp group OVERLAY-BGP-TO-RR family evpn signaling
set protocols bgp group OVERLAY-BGP-TO-RR vpn-apply-export
set protocols bgp group OVERLAY-BGP-TO-RR multipath
set protocols bgp group OVERLAY-BGP-TO-RR neighbor 192.168.2.2
```

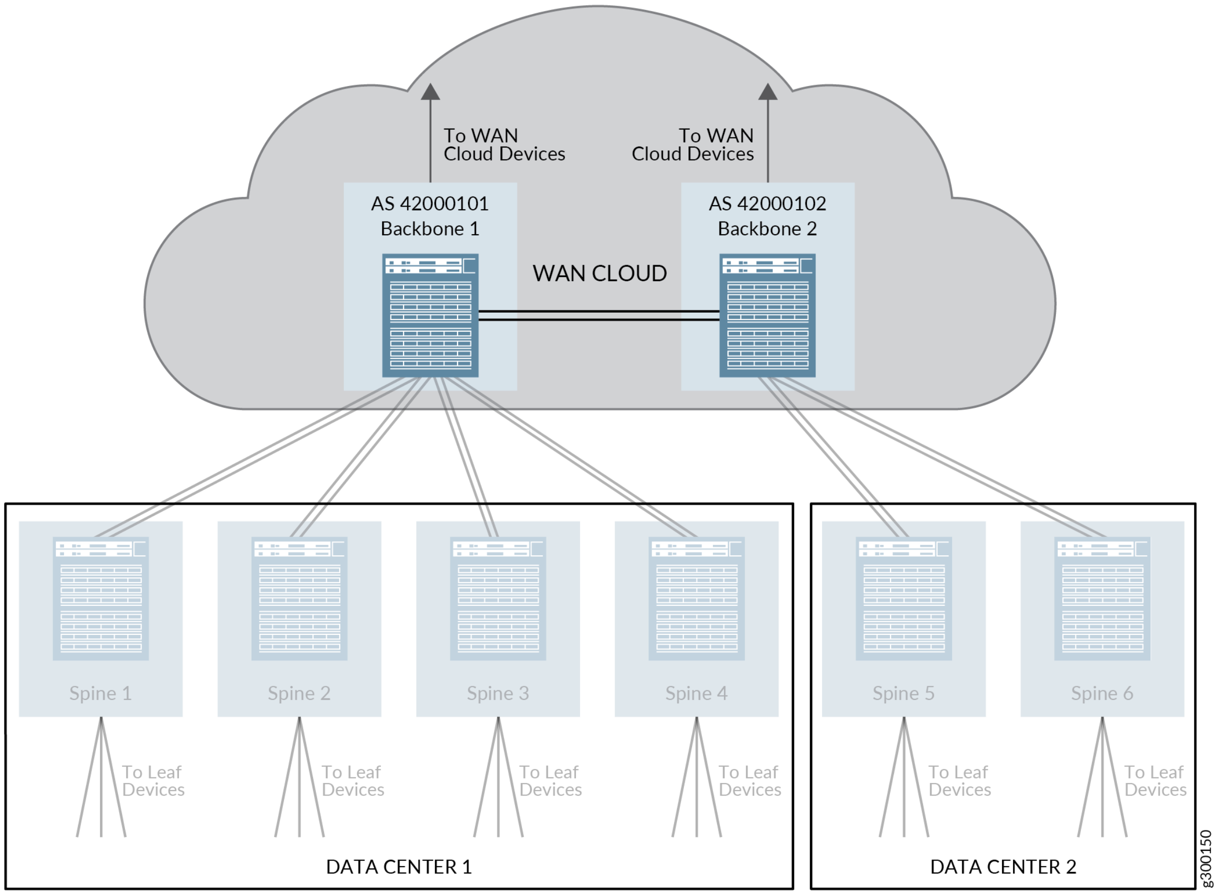
## Enabling EBGP as the Routing Protocol Between the Backbone Devices

EBGP is also used as the routing protocol between the backbone devices in this reference design. The backbone devices are connected using IP and the backbone devices must be configured as EBGP peers.

A second EBGP group—*BACKBONE-BGP*—is created in these steps to enable EBGP between the backbone devices. Each backbone device is assigned into a unique 32-bit AS number within the new EBGP group in these steps. The backbone devices, therefore, are part of two EBGP groups—*UNDERLAY-BGP* and *BACKBONE-BGP*—and have a unique AS number within each group. EVPN signaling, which has to run to support EVPN between the backbone devices, is also configured within the EBGP group during this procedure.

Figure 77 on page 346 illustrates the attributes needed to enable EBGP between the backbone devices.

**Figure 77: EBGP Topology for Backbone Device Connection**



To enable EBGP as the routing protocol between the backbone devices:

1. Create and name the BGP peer group. EBGP is enabled as part of this step.

   *Both Backbone Devices*:

   ```
   set protocols bgp group BACKBONE-BGP type external
   ```

2. Configure the ASN for each backbone device.

   *Backbone Device 1*:

   ```
   set protocols bgp group BACKBONE-BGP local-as 4200000101
   ```

*Backbone Device 2*:

```
set protocols bgp group BACKBONE-BGP local-as 4200000102
```

3. Configure the backbone devices as BGP peers.

   *Backbone Device 1*:

```
set protocols bgp group BACKBONE-BGP neighbor 172.16.200.1 peer-as 4200000102
```

   *Backbone Device 2*:

```
set protocols bgp group BACKBONE-BGP neighbor 172.16.200.0 peer-as 4200000101
```

4. Enable EVPN signaling between the backbone devices:

   *Both Backbone Devices*:

```
set protocols bgp group BACKBONE-BGP family evpn signaling
```

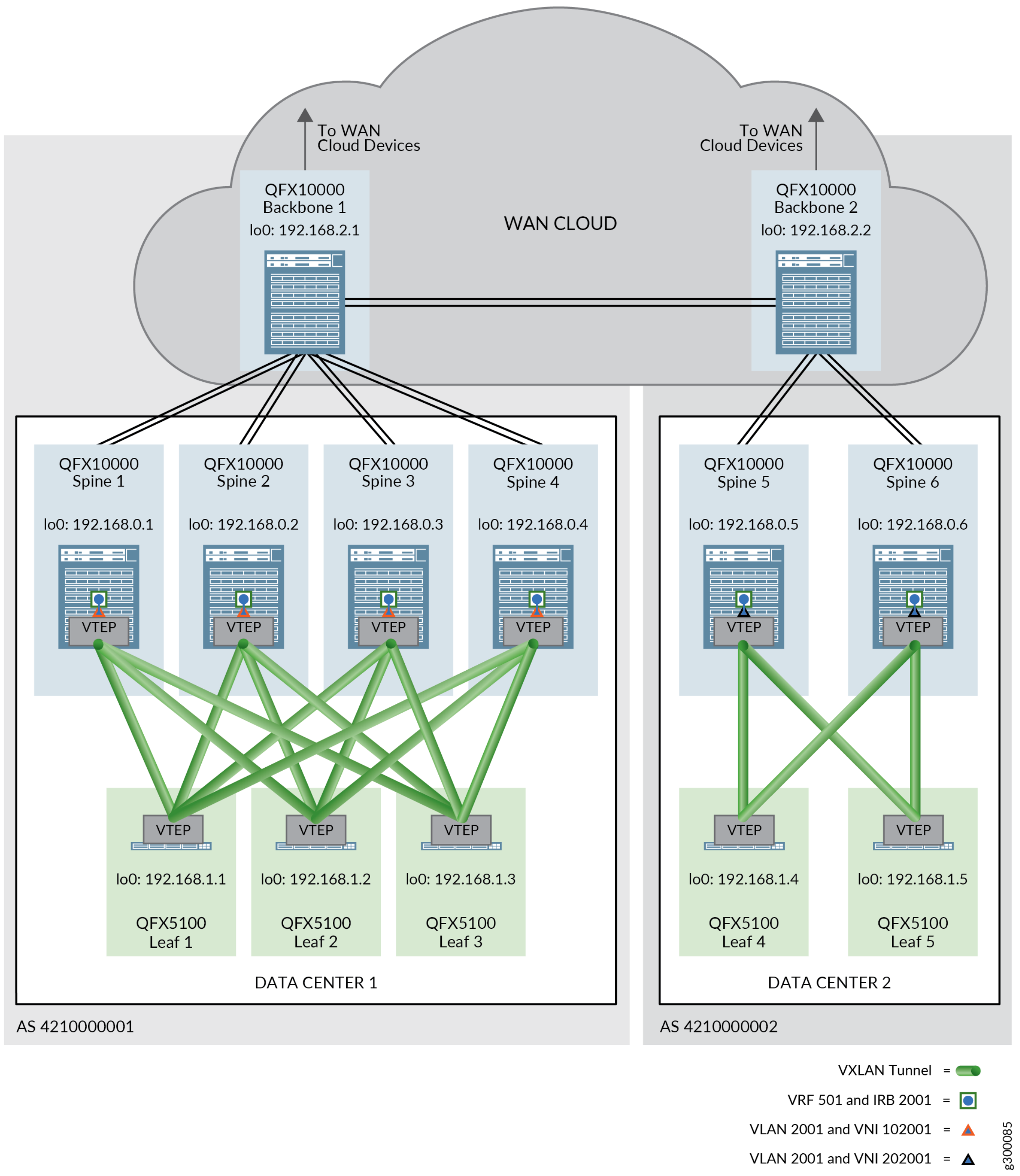
## Configuring DCI Using EVPN Type 5 Routes

EVPN Type 5 messages are exchanged between IRB interfaces on spine devices in different data centers when EVPN Type 5 routes are used for DCI. These IRB interfaces are configured in a routing instance.

Each data center has a unique virtual network identifier (VNI 102001 and 202001) in this configuration, but both VNIs are mapped to the same VLAN (VLAN 2001) in the same routing instance (VRF 501).

See Figure 78 on page 348 for an illustration of the routing instance.

**Figure 78: DCI Using EVPN Type 5 Routes**



To enable DCI using EVPN Type 5 routes:

**NOTE**: This procedure assumes that the routing instances, IRBs, & VLANs created earlier in this guide are operational. See "Centrally-Routed Bridging Overlay Design and Implementation" on page 142.

When implementing border leaf functionality on an MX router, keep in mind that the router supports virtual switch instances only. MX routers do not support default instances.

1. Configure the preferred addresses of the IRB interfaces.

   *Spine Device 2 in Data Center 1*:

```
set interfaces irb unit 2001 family inet address 10.20.1.242/24 preferred
set interfaces irb unit 2001 family inet6 address 2001:db8::10:20:1:242/112 preferred
```

   *Spine Device 5 in Data Center 2*:

```
set interfaces irb unit 2001 family inet address 10.30.1.245/24 preferred
set interfaces irb unit 2001 family inet6 address 2001:db8::10:30:1:245/112 preferred
```

2. Configure mapping between VLANs and the IRB interfaces.

   *Spine Device 2 in Data Center 1*:

```
set vlans 2001 vlan-id 2001
set vlans 2001 l3-interface irb.2001
```

   *Spine Device 5 in Data Center 2*:

```
set vlans 2001 vlan-id 2001
set vlans 2001 l3-interface irb.2001
```

3. Configure a routing instance, and map the IRB interface to this instance.

   *Spine Device 2 in Data Center 1*:

```
set routing-instances VRF-501 instance-type vrf
set routing-instances VRF-501 interface irb.2001
set routing-instances VRF-501 interface lo0.501
set routing-instances VRF-501 route-distinguisher 192.168.0.2:501
set routing-instances VRF-501 vrf-target import target:200:501
set routing-instances VRF-501 vrf-target export target:100:501
set routing-instances VRF-501 routing-options rib VRF-501.inet6.0 multipath
set routing-instances VRF-501 routing-options multipath
```

*Spine Device 5 in Data Center 2*:

```
set routing-instances VRF-501 instance-type vrf
set routing-instances VRF-501 interface irb.2001
set routing-instances VRF-501 interface lo0.501
set routing-instances VRF-501 route-distinguisher 192.168.0.5:501
set routing-instances VRF-501 vrf-target import target:100:501
set routing-instances VRF-501 vrf-target export target:200:501
set routing-instances VRF-501 routing-options rib VRF-501.inet6.0 multipath
set routing-instances VRF-501 routing-options multipath
```

4. Configure the VRF instance to generate EVPN Type 5 Routes.

   **NOTE**: The VNI of the local or remote data center—VNI 100501 or 200501 in this reference architecture—must be entered as the VNI in the **set routing-instances VRF-501 protocols evpn ip-prefix-routes vni** command.

   *Spine Device 2 in Data Center 1*:

```
set routing-instances VRF-501 protocols evpn ip-prefix-routes advertise direct-nexthop
set routing-instances VRF-501 protocols evpn ip-prefix-routes encapsulation vxlan
set routing-instances VRF-501 protocols evpn ip-prefix-routes vni 200501
```

   *Spine Device 5 in Data Center 2*:

```
set routing-instances VRF-501 protocols evpn ip-prefix-routes advertise direct-nexthop
set routing-instances VRF-501 protocols evpn ip-prefix-routes encapsulation vxlan
set routing-instances VRF-501 protocols evpn ip-prefix-routes vni 100501
```

5. On QFX5*xxx* switches that function as spine devices, enable the chained composite next hop feature. With this feature enabled, the switches can more efficiently process large amounts of EVPN Type 5 routes by directing routes that share the same destination to a common forwarding next hop.

   **NOTE**: On QFX10000 switches, this feature is enabled by default.

*Spine Device 2 in Data Center 1 and Spine Device 5 in Data Center 2*:

```
set routing-options forwarding-table chained-composite-next-hop ingress evpn
```

## Verifying That DCI Using EVPN Type 5 Routes is Operating

Enter the following commands to verify that traffic can be sent between data centers using EVPN Type 5 routes:

1. Verify that an EVPN Type 5 route has been received from the spine device in the other data center by entering the **show route table** command. Enter the VRF instance number and the route distinguisher in the command line to filter the results.

   *Spine Device 2 in Data Center 1*:

```
user@spine-device-2> show route table VRF-501.evpn.0 match-prefix 5:192.168.0.5:501*

5:192.168.0.5:501::0::10.30.1.0::24/248
                   *[BGP/170] 07:52:25, localpref 100, from 192.168.2.1
                      AS path: I, validation-state: unverified
                      to 172.16.102.0 via ae3.0
                    > to 172.16.102.2 via ae4.0
```

   *Spine Device 5 in Data Center 2*:

```
user@spine-device-5> show route table VRF-501.evpn.0 match-prefix 5:192.168.0.2:501*

VRF-501.evpn.0: 33 destinations, 49 routes (33 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

5:192.168.0.2:501::0::10.20.1.0::24/248
                    [BGP/170] 07:46:03, localpref 100, from 192.168.2.2
                      AS path: I, validation-state: unverified
                    > to 172.16.105.0 via ae3.0
                      to 172.16.105.2 via ae4.0
```

2. Verify that EVPN Type 5 routes are exported and imported in the VRF instance by entering the **show evpn ip-prefix-database l3-context** command and specifying the VRF instance.

*Spine Device 2 in Data Center 1*:

```
user@spine-device-2> show evpn ip-prefix-database l3-context VRF-501
L3 context: VRF-501

IPv4->EVPN Exported Prefixes
Prefix                                       EVPN route status
10.20.1.0/24                                 Created



IPv6->EVPN Exported Prefixes
Prefix                                       EVPN route status
2001:db8::10:20:1:0/112                      Created



EVPN->IPv4 Imported Prefixes
Prefix                                       Etag
10.30.1.0/24                                  0
  Route distinguisher    VNI/Label  Router MAC         Nexthop/Overlay GW/ESI
  192.168.0.5:501        200501     0c:86:10:cd:bf:f2  192.168.0.5

EVPN->IPv6 Imported Prefixes
Prefix                                       Etag
2000::200:0:1:0/112                          0
  Route distinguisher    VNI/Label  Router MAC         Nexthop/Overlay GW/ESI
  192.168.0.5:501        200501     0c:86:10:cd:bf:f2  192.168.0.5
```

*Spine Device 5 in Data Center 2*:

```
user@spine-device-5> show evpn ip-prefix-database l3-context VRF-501
L3 context: VRF-501

IPv4->EVPN Exported Prefixes
Prefix                                       EVPN route status
10.30.1.0/24                                 Created



IPv6->EVPN Exported Prefixes
Prefix                                       EVPN route status
2001:db8::10:30:1:0/112                      Created
```

```
EVPN->IPv4 Imported Prefixes
Prefix                                      Etag
10.20.1.0/24                                 0
  Route distinguisher    VNI/Label  Router MAC         Nexthop/Overlay GW/ESI
  192.168.0.2:501        200501     54:4b:8c:cd:c4:38  192.168.0.2


EVPN->IPv6 Imported Prefixes
Prefix                                      Etag
2001:db8::10:20:1:0/112                      0
  Route distinguisher    VNI/Label  Router MAC         Nexthop/Overlay GW/ESI
  192.168.0.2:501        200501     54:4b:8c:cd:c4:38  192.168.0.2
```

3. Verify the EVPN Type 5 route encapsulation details by entering the **show route table** command with the **extensive** option.

   *Spine Device 2 in Data Center 1*:

```
user@spine-device-2> show route table VRF-501.inet.0 match-prefix 10.30.1.0/24 extensive
VRF-501.inet.0: 21 destinations, 37 routes (21 active, 0 holddown, 0 hidden)
10.30.1.0/24 (3 entries, 1 announced)
        State: <CalcForwarding>
TSI:
KRT in-kernel 10.30.1.0/24 -> {list:composite(120781), composite(120780)}
        @EVPN   Preference: 170/-101
                Next hop type: Indirect, Next hop index: 0
                Address: 0x2874fc70
                Next-hop reference count: 11
                Next hop type: Router, Next hop index: 0
                Next hop: 172.16.102.0 via ae3.0
                Session Id: 0x0
                Next hop: 172.16.102.2 via ae4.0, selected
                Session Id: 0x0
                Protocol next hop: 192.168.0.5
                Composite next hop: 0x13f31948 120781 INH Session ID: 0x0
                  VXLAN tunnel rewrite:
                    MTU: 0, Flags: 0x0
                    Encap table ID: 0, Decap table ID: 506
                    Encap VNI: 200501, Decap VNI: 200501
                    Source VTEP: 192.168.0.2, Destination VTEP: 192.168.0.5
                    SMAC: 54:4b:8c:cd:c4:38, DMAC: 0c:86:10:cd:bf:f2
                Indirect next hop: 0xfc5ae80 2101590 INH Session ID: 0x0
```

```
                  State: <Active Int Ext>
                  Age: 7:54:59    Metric2: 0
                  Validation State: unverified
                  Task: VRF-501-EVPN-L3-context
                  AS path: I  (Originator)
                  Cluster list:  192.168.2.10
                  Originator ID: 192.168.0.5
                  Composite next hops: 1
                          Protocol next hop: 192.168.0.5
                          Composite next hop: 0x13f31948 120781 INH Session ID: 0x0
                            VXLAN tunnel rewrite:
                              MTU: 0, Flags: 0x0
                              Encap table ID: 0, Decap table ID: 506
                              Encap VNI: 200501, Decap VNI: 200501
                              Source VTEP: 192.168.0.2, Destination VTEP: 192.168.0.5
                              SMAC: 54:4b:8c:cd:c4:38, DMAC: 0c:86:10:cd:bf:f2
                          Indirect next hop: 0xfc5ae80 2101590 INH Session ID: 0x0
                          Indirect path forwarding next hops: 2
                                  Next hop type: Router
                                  Next hop: 172.16.102.0 via ae3.0
                                  Session Id: 0x0
                                  Next hop: 172.16.102.2 via ae4.0
                                  Session Id: 0x0
                                  192.168.0.5/32 Originating RIB: inet.0
                                    Node path count: 1
                                    Forwarding nexthops: 2
                                          Nexthop: 172.16.102.0 via ae3.0
                                          Session Id: 0
                                          Nexthop: 172.16.102.2 via ae4.0
                                          Session Id: 0
```

*Spine Device 5 in Data Center 2*:

```
user@spine-device-5> show route table VRF-501.inet.0 match-prefix 10.20.1.0/24 extensive

VRF-501.inet.0: 22 destinations, 39 routes (22 active, 0 holddown, 4 hidden)
10.20.1.0/24 (4 entries, 2 announced)
        State: <CalcForwarding>
TSI:
KRT in-kernel 10.20.1.0/24 -> {list:composite(108574), composite(103341)}
Page 0 idx 0, (group Internet type External) Type 1 val 0x283c765c (adv_entry)
   Advertised metrics:
     Nexthop: Self
     AS path: 4210000001 I
     Communities:
Path 10.20.1.0 Vector len 4.  Val: 0
        @EVPN   Preference: 170/-101
                Next hop type: Indirect, Next hop index: 0
                Address: 0x32a78cb0
                Next-hop reference count: 9
                Next hop type: Router, Next hop index: 0
                Next hop: 172.16.105.0 via ae3.0, selected
                Session Id: 0x0
                Next hop: 172.16.105.2 via ae4.0
                Session Id: 0x0
                Protocol next hop: 192.168.0.2
                Composite next hop: 0xfbc7e68 108574 INH Session ID: 0x0
                  VXLAN tunnel rewrite:
                    MTU: 0, Flags: 0x0
                    Encap table ID: 0, Decap table ID: 506
                    Encap VNI: 200501, Decap VNI: 200501
                    Source VTEP: 192.168.0.5, Destination VTEP: 192.168.0.2
                    SMAC: 0c:86:10:cd:bf:f2, DMAC: 54:4b:8c:cd:c4:38
                Indirect next hop: 0xfc5d400 2103405 INH Session ID: 0x0
                State: <Active Int Ext>
                Age: 7:49:18    Metric2: 0
                Validation State: unverified
                Task: VRF-501-EVPN-L3-context
                Announcement bits (2): 3-rt-export 4-BGP_RT_Background
                AS path: I  (Originator)
                Cluster list:  192.168.2.10
                Originator ID: 192.168.0.2
                Composite next hops: 1
                        Protocol next hop: 192.168.0.2
                        Composite next hop: 0xfbc7e68 108574 INH Session ID: 0x0
                         VXLAN tunnel rewrite:
                            MTU: 0, Flags: 0x0
                            Encap table ID: 0, Decap table ID: 506
                            Encap VNI: 200501, Decap VNI: 200501
                            Source VTEP: 192.168.0.5, Destination VTEP: 192.168.0.2
                            SMAC: 0c:86:10:cd:bf:f2, DMAC: 54:4b:8c:cd:c4:38
     Indirect next hop: 0xfc5d400 2103405 INH Session ID: 0x0
                        Indirect path forwarding next hops: 2
                                Next hop type: Router
                                Next hop: 172.16.105.0 via ae3.0
```

```
                                  Session Id: 0x0
                                  Next hop: 172.16.105.2 via ae4.0
                                  Session Id: 0x0
                                  192.168.0.2/32 Originating RIB: inet.0
                                    Node path count: 1
                                    Forwarding nexthops: 2
                                          Nexthop: 172.16.105.0 via ae3.0
                                          Session Id: 0
                                          Nexthop: 172.16.105.2 via ae4.0
                                          Session Id: 0
```

## DCI Using Type 5 Routes — Release History

Table 11 on page 356 provides a history of all of the features in this section and their support within this reference design.

**Table 11: DCI Using Type 5 Routes Release History**

| Release | Description |
|---|---|
| 19.1R2 | QFX10002-60C and QFX5120-32C switches running Junos OS Release 19.1R2 and later releases in the same release train support all features documented in this section. |
| 18.4R2-S2 | QFX5110 and QFX5120-48Y switches, and MX routers running Junos OS Release 18.4R2-S2 and later releases in the same release train support all features documented in this section. |

### RELATED DOCUMENTATION

*Understanding EVPN Pure Type-5 Routes*

Configure IBGP for the Overlay

Centrally-Routed Bridging Overlay Design and Implementation

# Data Center Interconnect Design and Implementation Using IPVPN

**IN THIS SECTION**

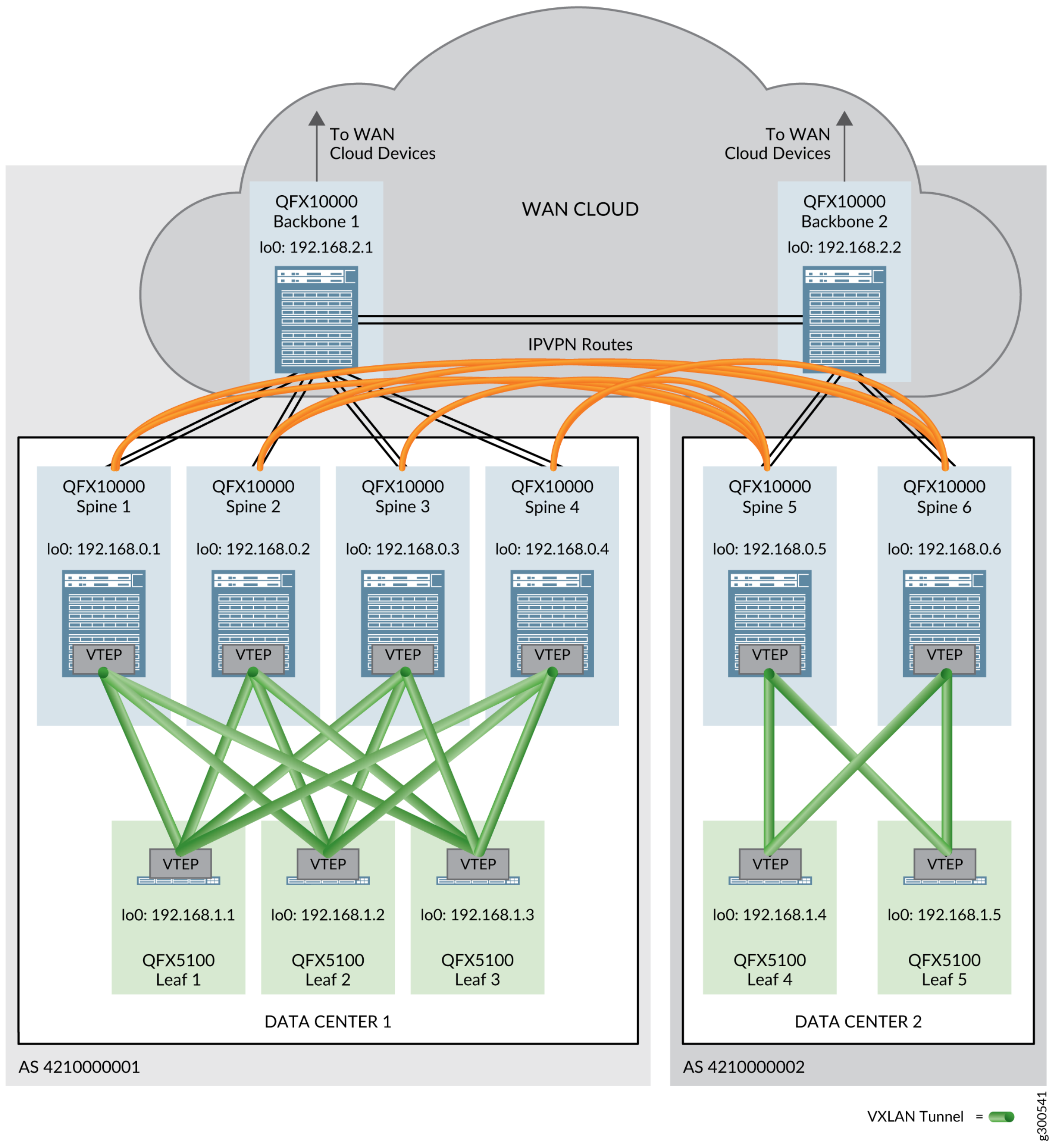
This section describes how to configure DCI using IPVPN. We are using IPVPN to pass traffic between data centers.

In this reference architecture, IPVPN routes are exchanged between spine devices in different data centers to allow for the passing of traffic between data centers.

Physical connectivity between the data centers is required before IPVPN routes can be sent across data centers. The backbone devices in a WAN cloud provide the physical connectivity. A backbone device is connected to each spine device in a single data center and participates in the overlay IBGP and underlay EBGP sessions. EBGP also runs in a separate BGP group to connect the backbone devices to each other; EVPN signaling and IPVPN (inet-vpn) is enabled in this BGP group.

Figure 79 on page 358 shows two data centers using IPVPN for DCI.

**Figure 79: Data Center Interconnect using IPVPN**



## Configuring Data Center Interconnect Using IPVPN

Configuring DCI for IPVPN is similar to configuring DCI for EVPN Type 5 routes with the exceptions shown in this section.

In this example, we are showing the configuration of IPVPN on Spine 1.

1. Configure the underlay link from the spine to the backbone.

```
set interfaces ae4 aggregated-ether-options minimum-links 1
set interfaces ae4 aggregated-ether-options lacp active
set interfaces ae4 aggregated-ether-options lacp periodic fast
set interfaces ae4 unit 0 family inet address 172.16.104.3/31
set interfaces ae4 unit 0 family mpls
```

2. On the backbone device, configure IBGP and MPLS for the overlay network. IPVPN requires that you use MPLS.

```
set protocols bgp group IPVPN-BGP local-address 192.168.0.1
set protocols bgp group IPVPN-BGP neighbor 192.168.2.1 family inet unicast
set protocols bgp group IPVPN-BGP neighbor 192.168.2.1 family inet-vpn unicast
set protocols bgp group IPVPN-BGP neighbor 192.168.2.1 family inet6-vpn unicast
set protocols rsvp interface ae4.0
set protocols rsvp interface lo0.0
set protocols mpls label-switched-path SPINE01-TO-REMOTE-SPINE5 to 192.168.0.5
set protocols mpls label-switched-path SPINE01-TO-REMOTE-SPINE5 no-cspf
set protocols mpls label-switched-path SPINE01-TO-REMOTE-SPINE6 to 192.168.0.6
set protocols mpls label-switched-path SPINE01-TO-REMOTE-SPINE6 no-cspf
set protocols mpls interface ae4.0
set protocols mpls interface lo0.0
```

3. On the spine, configure a routing instance to support DCI using IPVPN routes. This routing instance accepts L3VPN routes and also advertises data center routes as L3VPN routes to other IPVPN provider edge routers.

```
set routing-instances VRF-601 instance-type vrf
set routing-instances VRF-601 interface irb.2401
set routing-instances VRF-601 interface irb.2402
set routing-instances VRF-601 interface irb.2403
set routing-instances VRF-601 interface irb.2404
set routing-instances VRF-601 interface lo0.601
set routing-instances VRF-601 route-distinguisher 192.168.0.1:601
set routing-instances VRF-601 vrf-target target:200:601
set routing-instances VRF-601 vrf-table-label
set routing-instances VRF-601 routing-options rib VRF-601.inet6.0 multipath
set routing-instances VRF-601 routing-options multipath
```

## Verifying Data Center Interconnect Using IPVPN

1. Verify that data center routes are advertised as IPVPN routes to remote data centers.

```
host@SPINE-1> show interfaces terse irb.2401
Interface               Admin Link Proto    Local                 Remote
irb.2401                up    up   inet     30.1.145.244/24
                                            30.1.145.254/24
                                   inet6    2001:db8::30:0:191:244/112
                                            2001:db8::30:0:191:254/112
                                            fe80::e86:1009:61cd:bff2/64
```

```
host@SPINE-1> show route advertising-protocol bgp 192.168.2.1 table VRF-601.inet.0 match-
prefix 30.1.145.0 extensive
VRF-601.inet.0: 6091 destinations, 6115 routes (6091 active, 0 holddown, 0 hidden)
* 30.1.145.0/24 (1 entry, 1 announced)
BGP group underlay-bgp type External
     Route Distinguisher: 192.168.0.5:601
     VPN Label: 18
     Nexthop: Self
     Flags: Nexthop Change
     AS path: [4200000005] I
     Communities: target:200:601
```

2. On Spine 4, verify that the remote data center accepts the routes as IPVPN routes.

```
host@SPINE-4> show route table VRF-601.inet.0 match-prefix 30.1.145.0
VRF-601.inet.0: 6447 destinations, 6752 routes (6447 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

30.1.145.0/24      *[BGP/170] 1d 01:40:26, localpref 100
```

```
                      AS path: 4200000004 I, validation-state: unverified
                    > to 192.16.0.1 via ae4.0, Push 18
                    [BGP/170] 23:48:18, localpref 100
                      AS path: 4200000005 I, validation-state: unverified
                    > to 192.16.0.2 via ae5.0, Push 18
```

## Data Center Interconnect—Release History

Table 12 on page 361 provides a history of all of the features in this section and their support within this reference design.

**Table 12: DCI Using IPVPN Release History**

| Release | Description |
|---|---|
| 19.1R2 | QFX10002-60C switches running Junos OS Release 19.1R2 and later releases in the same release train support DCI using IPVPN. |
| 18.4R2-S2 | MX routers running Junos OS Release 18.4R2-S2 and later releases in the same release train also support DCI using IPVPN. |
| 18.1R3-S5 | All devices in the reference design that support Junos OS Release 18.1R3-S5 and later releases in the same release train also support DCI using IPVPN. |

# Configure VXLAN Stitching for Layer 2 Data Center Interconnect

**IN THIS SECTION**

This document describes the configuration and validation steps for implementing Data Center Interconnect (DCI) using VXLAN stitching in a gateway device. The VXLAN stitching feature enables you to stitch together specific VXLAN Virtual Network Identifiers (VNIs) to provide Layer 2 stretch between DCs on a granular basis.

Juniper Network's switching and routing devices support a number of different DCI options. For example, Over the Top (OTT) DCI can be used to extend the overlay between PODS. See OTT DCI for details. One drawback to the OTT method is it extends all VLANs between the PODs, either at layer 2 or Layer 3. Also, OTT DCI requires end-to-end VXLAN VNI significance. This can be an issue if two DC/ PODs are being merged that don't have overlapping VLAN to VNI assignments.

In some cases you want a more granular control over which VLANs are extended between PODs. The Junos VXLAN stitching feature allows you to perform DCI at the VNI level to extend Layer 2 connectivity on a per VLAN basis. Or, you might need to translate VNIs to accommodate instances where the same VNIs are assigned to different VLANs in each POD. For example, take the case where VLAN 1 is assigned VNI 10001 in POD 1, while in POD 2 the same VLAN is assigned to VNI 20002. In this case you either have to reconfigure one of the PODs to achieve a global (overlapping) mapping of VLANs to VNIs. Alternatively, you can employ translational stitching to map local POD VNI values to the VNI used over the WAN.

Juniper Networks supports VXLAN stitching for both 3-stage and 5-stage IP fabrics. In addition, VXLAN stitching is supported for centrally-routed bridging (CRB) overlay, edge-routed bridging (ERB) overlay, and bridged overlay architectures. This use case assumes that your EVPN-VXLAN POD fabrics are already configured with leaves and spines using one or a combination of the supported architectures shown in Table 13 on page 364.

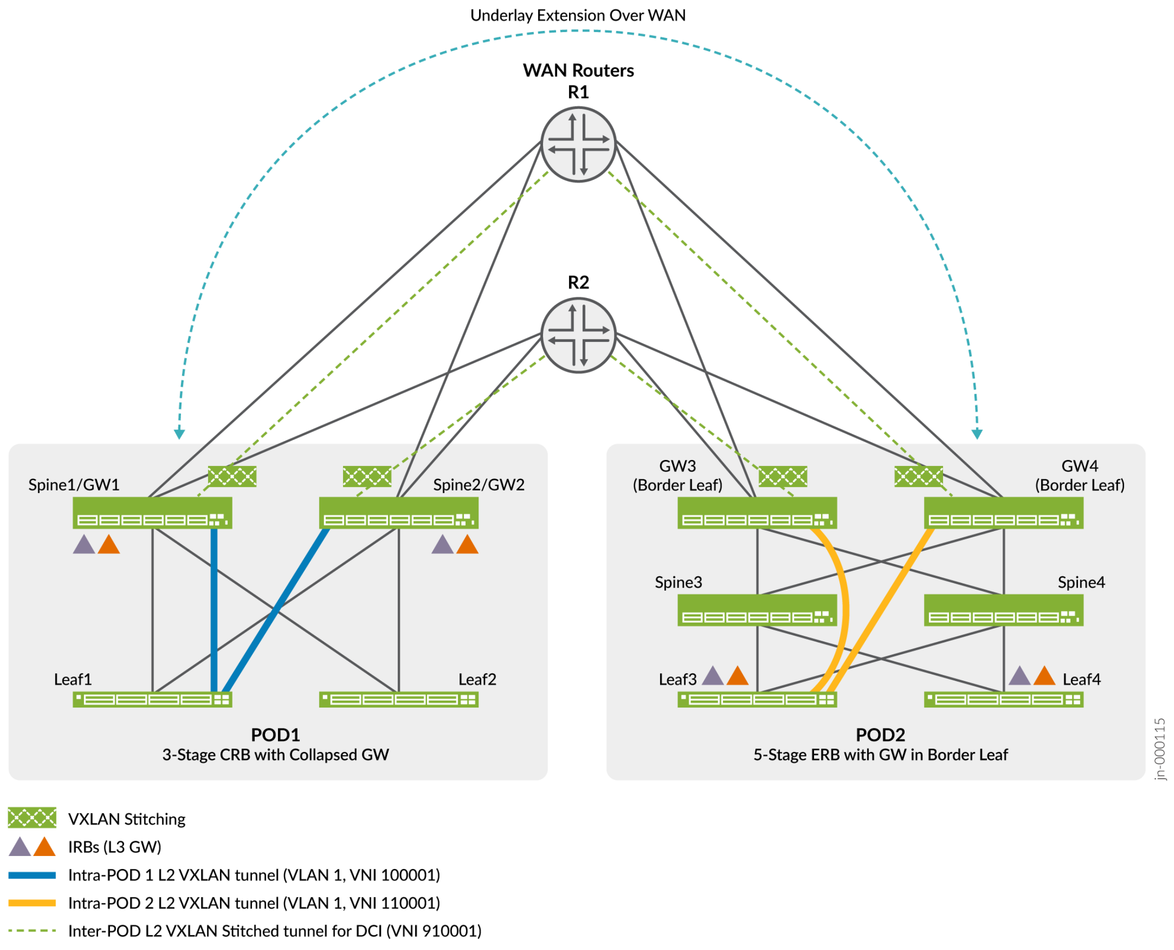
To enable VXLAN stitched connectivity between the two PODs, you add a tier of WAN routers to extend the underlay. The underlay extension extends the overlay between the PODs. Thenyou configure

VXLAN stitching on the gateway devices to extend the desired VLANs (now represented as VXLAN VNIs), between the PODs.

**NOTE**: We use the term "WAN Routers" in this document. This doesn't imply that you have an actual WAN network between the PODs. The WAN routers might be local to both PODs, as is the case in this example. You can also use VXLAN stitching over an extended WAN network when the PODs are geographically remote.

Figure 80 on page 363 provides a high level diagram showing the POD/DC fabric types we validated in this reference design.

**Figure 80: VXLAN Stitching Reference Architectures**


Underlay Extension Over WAN
WAN Routers
R1
R2
Spine1/GW1
Spine2/GW2
Leaf1
Leaf2
POD1
3-Stage CRB with Collapsed GW
GW3
(Border Leaf)
GW4
(Border Leaf)
Spine3
Spine4
Leaf3
Leaf4
POD2
5-Stage ERB with GW in Border Leaf
jn-000115
VXLAN Stitching
IRBs (L3 GW)
Intra-POD 1 L2 VXLAN tunnel (VLAN 1, VNI 100001)
Intra-POD 2 L2 VXLAN tunnel (VLAN 1, VNI 110001)
Inter-POD L2 VXLAN Stitched tunnel for DCI (VNI 910001)

In Figure 80 on page 363, each WAN router connects to each gateway device in both PODs. These connections and the related BGP peer sessions serve to extend the underlay between the two PODs.

Specifically, the devices advertise the loopback addresses of the gateway devices between the PODs. This loopback reachability establishes an EBGP based peering session to extend the overlay between the gateway devices in both pods.

POD 1 represents a 3-stage CRB architecture with the gateway function collapsed into the spine devices. Thus, in POD 1 the terms spine and gateway are each applicable. In general we'll use the term gateway when describing the spine devices as the focus here is on their gateway functionality.

POD 2, in contrast, is a 5-stage ERB architecture with lean spines and discrete gateway devices. The gateway devices in POD 2 can also be called super-spine or border leaf devices. In the context of this example, they perform the VXLAN stitching functionality and so we refer to them as gateway devices.

Table 13 on page 364 outlines the POD architectures we validated as part of this reference design.

**Table 13: Supported POD Architectures for VXLAN Stitching**

| POD 1 | POD 2 |
|---|---|
| Edge-Routed Bridging | Edge-Routed Bridging |
| Centrally-Routed Bridging | Edge-Routed Bridging |
| Centrally-Routed Bridging | Centrally-Routed Bridging |
| Bridged Overlay | Bridged Overlay |
| 3 or 5 stage fabric | 3 or 5 stage fabric |

Other items to note when using VXLAN stitching include:

- You can combine the role of spine and gateway into a collapsed design as shown for POD 1.
- The stitched VNI can have the same value (global stitching) when the PODs have overlapping VLAN to VNI assignments, or can be translated between the two PODs. The latter capability is useful when merging PODs (DCs) that don't have overlapping VNI to VLAN assignments.
- We support VXLAN stitching in the default-switch EVPN instance (EVI) and in MAC-VRF routing instances.
- We support Layer 2 stitching for unicast and BUM traffic only. With BUM traffic, the designated forwarder (DF) for the local gateway ESI LAG performs ingress replication and forwards a copy of the BUM traffic to each remote gateway. At the remote gateway devices, the DF for the remote ESI LAG performs ingress replication and sends a copy of the BUM traffic to all leaf nodes in the local POD.

- The gateway device must be a switch in the QFX10000 line running Junos Software Release 20.4R3 or higher.

- We recommend that you configure the IRB interfaces on the spine devices in a CRB fabric with the `proxy-macip-advertisement` configuration statement. This option ensures correct ARP operation over a CRB EVPN-VXLAN fabric and is part of the CRB reference architecture. See proxy-mac-ip-advertisement for more information on this option.

Note the following about the EVPN–VXLAN fabric reference design:

- This example assumes that the tiers of spine and leaf devices in the two PODs already exist and are up and running. As a result, this topic provides the configuration for the gateway to WAN router EBGP underlay peering, the inter-POD EBGP overlay peering, and the configuration needed for VXLAN stitching.

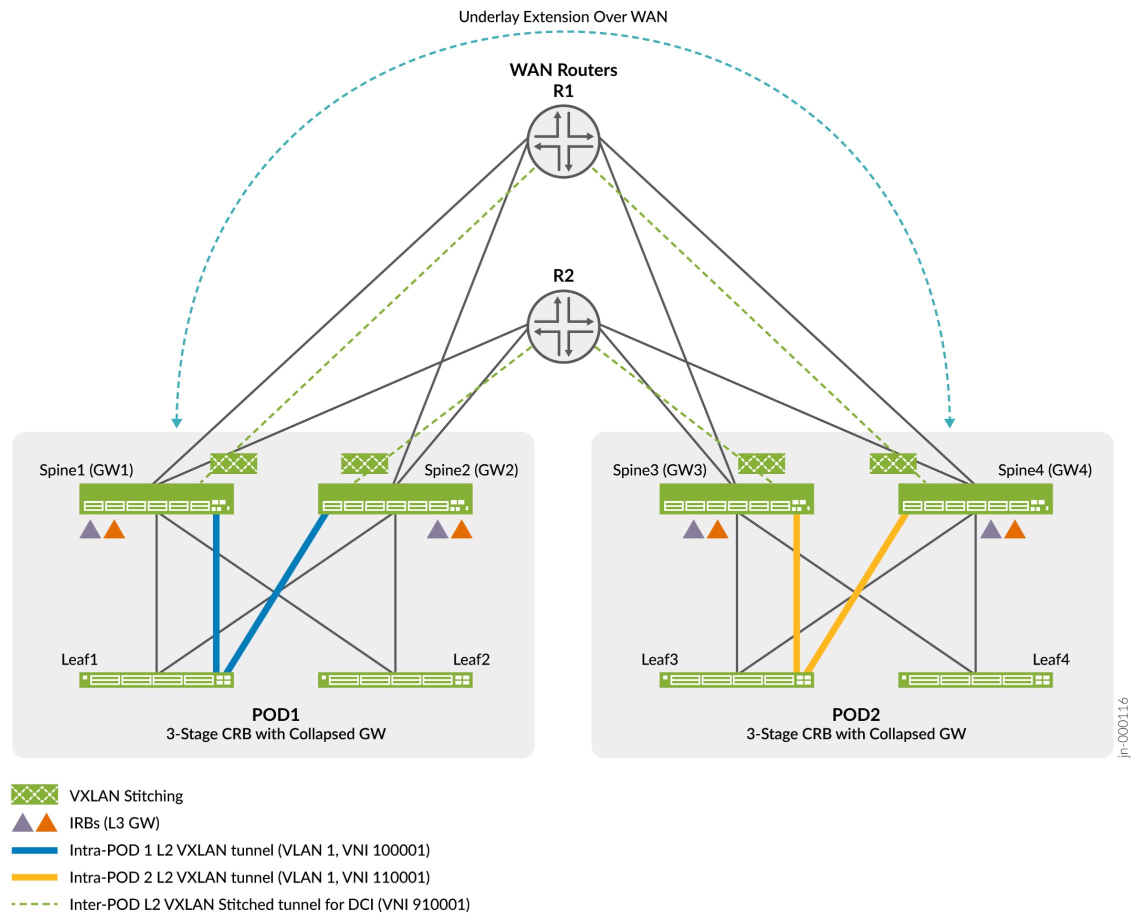
  For information about configuring the spine and leaf devices in the two PODs, see the following:

  - "IP Fabric Underlay Network Design and Implementation" on page 80
  - "Configure IBGP for the Overlay" on page 96
  - "Centrally-Routed Bridging Overlay Design and Implementation" on page 142
  - "Edge-Routed Bridging Overlay Design and Implementation" on page 206
  - "Bridged Overlay Design and Implementation" on page 112

- This example integrates the WAN routers into an existing two POD EVPN-VXLAN fabric. To keep the focus on VXLAN stitching, both PODs in the example use the same 3-stage Clos fabric based on a CRB architecture. In addition to their role as Layer 3 VXLAN gateways, the spines also perform the VXLAN stitching function. The result is an example of a collapsed gateway architecture.

  Figure 81 on page 366 shows the collapsed gateway CRB based VXLAN stitching example topology.

**Figure 81: VXLAN Stitching Example Topology**


Underlay Extension Over WAN
WAN Routers
R1
R2
Spine1 (GW1)
Spine2 (GW2)
Spine3 (GW3)
Spine4 (GW4)
Leaf1
Leaf2
Leaf3
Leaf4
POD1
3-Stage CRB with Collapsed GW
POD2
3-Stage CRB with Collapsed GW
jn-000116
VXLAN Stitching
IRBs (L3 GW)
Intra-POD 1 L2 VXLAN tunnel (VLAN 1, VNI 100001)
Intra-POD 2 L2 VXLAN tunnel (VLAN 1, VNI 110001)
Inter-POD L2 VXLAN Stitched tunnel for DCI (VNI 910001)

In this example, you add the gateway functionality to a pre-existing CRB spine configuration. As noted above, we also support 5-stage architectures with the super-spine layer performing the gateway peering and stitching functions. We recommend using a discrete gateway device for maximum scaling and performance. With a 3-stage or 5-stage ERB architecture, you add the gateway configuration to the lean spine or super spine devices, respectively.

- When configuring the overlay BGP peering between the PODs, you can use either IBGP or EBGP. Typically, you use IBGP if your data centers (PODs) use the same autonomous system (AS) number and EBGP if your PODs use different AS numbers. Our example uses different AS numbers in each POD, therefore EBGP peering is used to extend the overlay between the PODs.

- After you integrate the WAN routers to extend the underlay and overlay between the two PODs, you configure translational VXLAN stitching to extend a given VLAN between the PODs. Translational VXLAN stitching translates the VNI value used locally in each POD to a common VNI value used across the WAN segment. Note that in our example, we assign VLAN 1 a different (non-overlapping) VNI value in each POD. This is why we use translational stitching in this case. You normally use global mode stitching when the same VNI value is mapped to the same VLAN in both PODS.

## What's Next

- MAC-VRF Instance Type Overview
- EVPN-VXLAN DC IP Fabric MAC VRF L2 services

## Configure Gateway Devices to Extend the Underlay Over the WAN

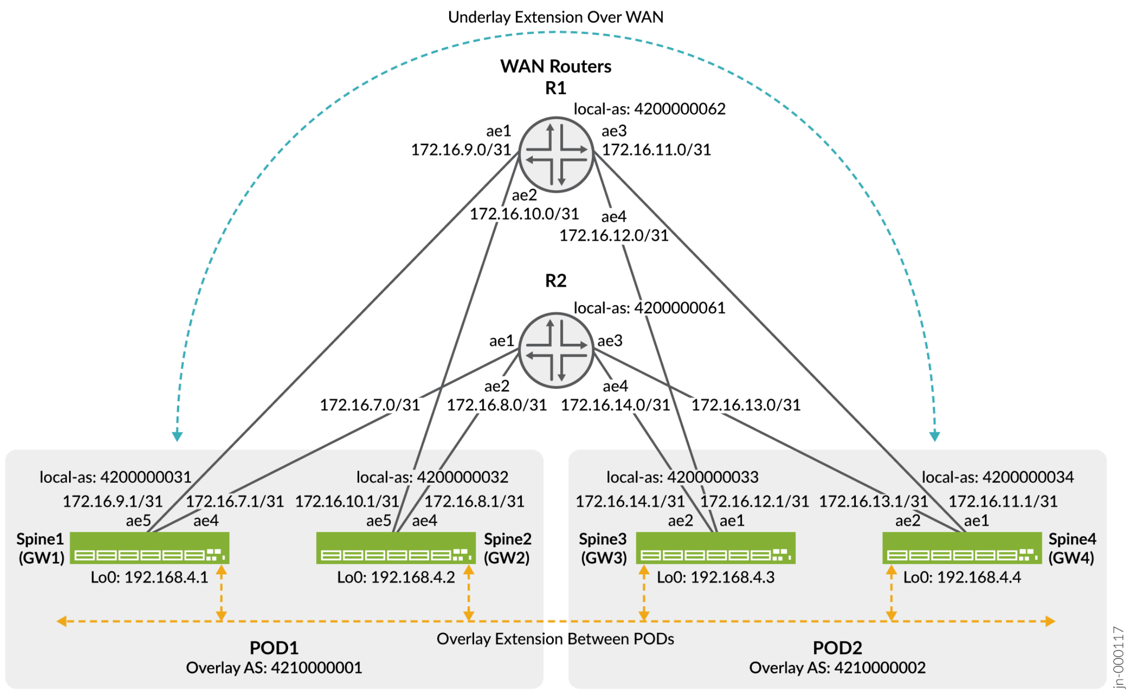
This section shows you how to configure the collapsed gateway devices (a CRB spine with added VXLAN stitching gateway functionality) so they can communicate with the WAN devices. Recall that each POD already has a fully functional underlay and CRB overlay based on the reference implementation for a 3-stage CRB architecture. See "Centrally-Routed Bridging Overlay Design and Implementation" on page 142 for details.

You configure the spine/gateway devices to peer with the WAN routers to extend the underlay between the two PODs. This involves configuring EBGP peering and policy to tag and advertise the loopback routes from each gateway. These routes establish the inter-POD EBGP peering sessions that extend the fabric overlay in the next section.

**NOTE**: Configuring the WAN routers is outside the scope of this document. They simply need to support aggregate Ethernet interfaces and EBGP peering to the gateway devices. In this example the WAN routers must re-advertise all routes received from one POD to the other. In the case of a Junos device, this is the default policy for the EBGP underlay peering in this example.

Figure 82 on page 368 provides the details regarding interfaces, IP addressing, and AS numbering for the DCI portion of the POD networks.

**Figure 82: Details for Underlay and Overlay Extension Across the WAN**



The configuration on all the gateway devices is similar. We'll walk you through configuring the gateway 1 device and then provide the full configuration delta for the other 3 gateways.

## Gateway 1

1. Configure the interfaces that connect the gateway 1 device to the two WAN routers.

   Here, we create an aggregated Ethernet (AE) interface that includes a single member. With this approach you can easily add additional member links to increase the WAN throughput or resiliency.

```
set interfaces et-0/0/1 ether-options 802.3ad ae4
set interfaces ae4 unit 0 family inet address 172.16.7.1/31
set interfaces et-0/0/2 ether-options 802.3ad ae5
set interfaces ae5 unit 0 family inet address 172.16.9.1/31
```

2. Create a BGP peer group named `underlay-bgp-wan`, and configure it as an EBGP group.

```
set protocols bgp group underlay-bgp-wan type external
```

3. Configure the EBGP Underlay AS number.

In this reference design, you assign a unique AS number to each device in the underlay network. See Figure 82 on page 368 for the AS numbers of the gateway and WAN devices.

You configure the AS number for EBGP in the underlay network at the BGP peer group level using the `local-as` statement because the system AS number setting at the `[edit routing-options autonomous-system]` hierarchy is used for MP-IBGP overlay peering in the local fabric, and for the EBGP peering used to extend the overlay between the PODs.

```
set protocols bgp group underlay-bgp-wan local-as 4200000031
```

4. Configure EBGP peering with WAN devices 1 and 2.

   You configure each WAN device as a EBGP neighbor by specifying the WAN device's IP address and AS number. See Figure 82 on page 368 for the IP addresses and AS numbers of the spine devices.

```
set protocols bgp group underlay-bgp-wan neighbor 172.16.7.0 peer-as 4200000061
set protocols bgp group underlay-bgp-wan neighbor 172.16.9.0 peer-as 4200000062
```

5. Configure an import routing policy that subtracts 10 from the local preference value of routes received from the WAN when they are tagged with a specific community. This policy ensures that the gateway device always prefers a local underlay route, even when the same gateway loopback address is also learned over the WAN peering.

   Recall that we use EBGP for gateway to WAN router peering. By default, EBGP readvertises all routes received to all other EBGP (and IBGP) neighbors. This means that when gateway 1 advertises its loopback route to WAN router 1, the WAN router *readvertises* that route to gateway 2. The result is that each gateway has both an intra-fabric route and an inter-fabric route to reach the other gateway in its local POD.

   We want to ensure that the gateway always prefers the intra-fabric path. We do this by adjusting the local preference value for routes received from the WAN (to make them less preferred regardless of AS path length). The policy also blocks the readvertisement of gateway loopback routes learned over the WAN peering into the local fabric. The result is the leaf devices see only the intra-fabric gateway loopback route while the gateway devices always prefer the intra-fabric gateway route.

   You define the referenced community in the next step.

```
set policy-options policy-statement wan-import from community wan_underlay_comm
set policy-options policy-statement wan_import then local-preference subtract 10
set protocols bgp group underlay-bgp-wan import wan-import
```

**NOTE**: This examples assumes you are starting from a baseline reference architecture in both PODs. Part of the pre-existing reference baseline is the fabric underlay and overlay related BGP peering and policy. This example is based on the spine and gateway being collapsed into a single device. Now that you have added underlay and overlay extension via WAN routers, you should modify your existing underlay policy on the gateway, or in our case the spine/gateway device, to block readvertisement of routes tagged with the *wan_underlay_comm* from the other fabric devices.

We show an example of this modification here. The newly added *from_wan* term suppresses advertisement of routes with the matching community into the fabric underlay.

```
user@Spine-1> show configuration policy-options policy-statement underlay-clos-
export
term loopback {
    from interface lo0.0;
    then accept;
}
term from_wan {
    from community wan_underlay_comm;
    then reject;
}
```

6. Configure an export routing policy to advertise the gateway loopback interface address to the WAN devices. This policy rejects all other advertisements. You now define the *wan_underlay_comm* community used to tag these routes.

```
set policy-options policy-statement underlay-clos-export-wan term loopback from interface
lo0.0
set policy-options policy-statement underlay-clos-export-wan term loopback then community add
wan_underlay_comm
set policy-options policy-statement underlay-clos-export-wan term loopback then accept
set policy-options policy-statement underlay-clos-export-wan term def then reject
set policy-options community wan_underlay_comm members 12345:12345
set protocols bgp group underlay-bgp-wan export underlay-clos-export-wan
```

7. Configure multipath with the `multiple-as` option to enable load balancing between EBGP peers in different ASs.

By default, EBGP selects one best path for each prefix and installs that route in the forwarding table. Also, you configure BGP multipath so all equal-cost paths to a given destination are installed into the routing table.

```
set protocols bgp group underlay-bgp-wan multipath multiple-as
```

8. Enable Bidirectional Forwarding Detection (BFD) for the WAN EBGP sessions. BFD enables rapid detection of failures and thereby fast reconvergence.

```
set protocols bgp group underlay-bgp-wan bfd-liveness-detection minimum-interval 1000
set protocols bgp group underlay-bgp-wan bfd-liveness-detection multiplier 3
set protocols bgp group underlay-bgp-wan bfd-liveness-detection session-mode automatic
```

## Configure Gateway Devices to Extend the Overlay Over the WAN

This section show how to extend the EVPN overlay between the two PODs using EBGP. Recall that in this example the two PODs have unique AS numbers, so EBGP is used.

As is typical for 3-stage CRB fabric, our spine devices (gateways) function as route reflectors in the overlay for the leaf devices in their respective PODs. In this section you define a new EBGP peering group that extends the overlay between the PODs. See Figure 82 on page 368 for details about the AS numbering and spine device loopback addresses.

The configuration on all the gateway devices is similar. Once again, we'll walk you through configuring the gateway 1 device, and provide the full configuration delta for the other 3 gateways.

**Gateway 1**

1. Configure the EBGP group to extend the EVPN overlay to the remote gateway devices.

   Normally, we use IBGP for an EVPN overlay. We use EBGP here because we assigned the PODS different AS numbers. Note here that you must enable the `multihop` option. By default, EBGP expects a directly connected peer. In this example, the peer is remotely attached to the far side of the WAN. Also, you must configure the `no-nexthop-change` option. This option alters the default EBGP behavior of updating the BGP next hop to a local value when re-advertising routes. With this option, you tell the gateway device to leave the BGP protocol next hop for the overlay route unchanged. This is important because the gateway IP address may not be a VXLAN VTEP address, for example, in an ERB fabric where the next hop for an EVPN type 2 route must identify that leaf device. Not overwriting the next hop ensures that the correct VTEPs are used for VXLAN tunnels.

You configure the EBGP peering between the gateway device loopback addresses.

```
set protocols bgp group overlay-ebgp-extn-dci type external
set protocols bgp group overlay-ebgp-extn-dci multihop no-nexthop-change
set protocols bgp group overlay-ebgp-extn-dci local-address 192.168.4.1
set protocols bgp group overlay-ebgp-extn-dci family evpn signaling delay-route-
advertisements minimum-delay routing-uptime 400
set protocols bgp group overlay-ebgp-extn-dci family route-target external-paths 2
set protocols bgp group overlay-ebgp-extn-dci local-as 4210000001
set protocols bgp group overlay-ebgp-extn-dci multipath multiple-as
set protocols bgp group overlay-ebgp-extn-dci neighbor 192.168.4.3 peer-as 4210000002
set protocols bgp group overlay-ebgp-extn-dci neighbor 192.168.4.4 peer-as 4210000002
```

2. As with the underlay peering, we add BFD for rapid failure detection in the overlay extension. Note that here we specify a longer interval for the overlay peering. In the underlay extension peering, we used a 1-second interval. Here we configure a 4-second interval to help ensure the overlay sessions remain up in the event of an underlay failure that requires reconvergence.

```
set protocols bgp group overlay-ebgp-extn-dci bfd-liveness-detection minimum-interval 4000
set protocols bgp group overlay-ebgp-extn-dci bfd-liveness-detection multiplier 3
set protocols bgp group overlay-ebgp-extn-dci bfd-liveness-detection session-mode automatic
```

3. Be sure to commit the changes on all gateway devices after these steps.

## Gateway Device Configurations for Underlay and Overlay Extension

This section provides the configuration delta for all four gateway devices. You add this delta to the initial CRB baseline to extend the POD underlay and overlay over the WAN.

**NOTE**: The final two statements modify the existing fabric underlay policy to block re-advertisement of routes tagged with the *wan_underlay_comm* community from the other leaf devices.

Gateway 1 (POD 1)

```
set interfaces et-0/0/1 ether-options 802.3ad ae4
set interfaces ae4 unit 0 family inet address 172.16.7.1/31
set interfaces et-0/0/2 ether-options 802.3ad ae5
set interfaces ae5 unit 0 family inet address 172.16.9.1/31
set protocols bgp group underlay-bgp-wan type external
set protocols bgp group underlay-bgp-wan local-as 4200000031
set protocols bgp group underlay-bgp-wan neighbor 172.16.7.0 peer-as 4200000061
set protocols bgp group underlay-bgp-wan neighbor 172.16.9.0 peer-as 4200000062
set policy-options policy-statement wan_import from community wan_underlay_comm
set policy-options policy-statement wan_import then local-preference subtract 10
set protocols bgp group underlay-bgp-wan import wan-import
set policy-options policy-statement underlay-clos-export-wan term loopback from interface lo0.0
set policy-options policy-statement underlay-clos-export-wan term loopback then community add
wan_underlay_comm
set policy-options policy-statement underlay-clos-export-wan term loopback then accept
set policy-options policy-statement underlay-clos-export-wan term def then reject
set policy-options community wan_underlay_comm members 12345:12345
set protocols bgp group underlay-bgp-wan export underlay-clos-export-wan
set protocols bgp group underlay-bgp-wan multipath multiple-as
set protocols bgp group underlay-bgp-wan bfd-liveness-detection minimum-interval 1000
set protocols bgp group underlay-bgp-wan bfd-liveness-detection multiplier 3
set protocols bgp group underlay-bgp-wan bfd-liveness-detection session-mode automatic

set protocols bgp group overlay-ebgp-extn-dci type external
set protocols bgp group overlay-ebgp-extn-dci multihop no-nexthop-change
set protocols bgp group overlay-ebgp-extn-dci local-address 192.168.4.1
set protocols bgp group overlay-ebgp-extn-dci family evpn signaling delay-route-advertisements
minimum-delay routing-uptime 400
set protocols bgp group overlay-ebgp-extn-dci family route-target external-paths 2
set protocols bgp group overlay-ebgp-extn-dci local-as 4210000001
set protocols bgp group overlay-ebgp-extn-dci multipath multiple-as
set protocols bgp group overlay-ebgp-extn-dci neighbor 192.168.4.3 peer-as 4210000002
set protocols bgp group overlay-ebgp-extn-dci neighbor 192.168.4.4 peer-as 4210000002
set protocols bgp group overlay-ebgp-extn-dci bfd-liveness-detection minimum-interval 4000
set protocols bgp group overlay-ebgp-extn-dci bfd-liveness-detection multiplier 3
set protocols bgp group overlay-ebgp-extn-dci bfd-liveness-detection session-mode automatic

set policy-options policy-statement underlay-clos-export term from_wan from community underlay-
clos-export
set policy-options policy-statement underlay-clos-export term from_wan then reject
```

Gateway 2 (Pod 1)

```
set interfaces et-0/0/1 ether-options 802.3ad ae4
set interfaces ae4 unit 0 family inet address 172.16.8.1/31
set interfaces et-0/0/2 ether-options 802.3ad ae5
set interfaces ae5 unit 0 family inet address 172.16.10.1/31
set protocols bgp group underlay-bgp-wan type external
set protocols bgp group underlay-bgp-wan local-as 4200000032
set protocols bgp group underlay-bgp-wan neighbor 172.16.8.0 peer-as 4200000061
set protocols bgp group underlay-bgp-wan neighbor 172.16.10.0 peer-as 4200000062
set policy-options policy-statement wan_import from community wan_underlay_comm
set policy-options policy-statement wan_import then local-preference subtract 10
set protocols bgp group underlay-bgp-wan import wan-import
set policy-options policy-statement underlay-clos-export-wan term loopback from interface lo0.0
set policy-options policy-statement underlay-clos-export-wan term loopback then community add
wan_underlay_comm
set policy-options policy-statement underlay-clos-export-wan term loopback then accept
set policy-options policy-statement underlay-clos-export-wan term def then reject
set policy-options community wan_underlay_comm members 12345:12345
set protocols bgp group underlay-bgp-wan export underlay-clos-export-wan
set protocols bgp group underlay-bgp-wan multipath multiple-as
set protocols bgp group underlay-bgp-wan bfd-liveness-detection minimum-interval 1000
set protocols bgp group underlay-bgp-wan bfd-liveness-detection multiplier 3
set protocols bgp group underlay-bgp-wan bfd-liveness-detection session-mode automatic

set protocols bgp group overlay-ebgp-extn-dci type external
set protocols bgp group overlay-ebgp-extn-dci multihop no-nexthop-change
set protocols bgp group overlay-ebgp-extn-dci local-address 192.168.4.2
set protocols bgp group overlay-ebgp-extn-dci family evpn signaling delay-route-advertisements
minimum-delay routing-uptime 400
set protocols bgp group overlay-ebgp-extn-dci family route-target external-paths 2
set protocols bgp group overlay-ebgp-extn-dci local-as 4210000001
set protocols bgp group overlay-ebgp-extn-dci multipath multiple-as
set protocols bgp group overlay-ebgp-extn-dci neighbor 192.168.4.3 peer-as 4210000002
set protocols bgp group overlay-ebgp-extn-dci neighbor 192.168.4.4 peer-as 4210000002
set protocols bgp group overlay-ebgp-extn-dci bfd-liveness-detection minimum-interval 4000
set protocols bgp group overlay-ebgp-extn-dci bfd-liveness-detection multiplier 3
set protocols bgp group overlay-ebgp-extn-dci bfd-liveness-detection session-mode automatic

set policy-options policy-statement underlay-clos-export term from_wan from community underlay-
clos-export
set policy-options policy-statement underlay-clos-export term from_wan then reject
```

Gateway 3 (POD 2)

```
set interfaces et-0/0/1 ether-options 802.3ad ae1
set interfaces ae1 unit 0 family inet address 172.16.12.1/31
set interfaces et-0/0/2 ether-options 802.3ad ae2
set interfaces ae2 unit 0 family inet address 172.16.14.1/31
set protocols bgp group underlay-bgp-wan type external
set protocols bgp group underlay-bgp-wan local-as 4200000033
set protocols bgp group underlay-bgp-wan neighbor 172.16.12.0 peer-as 4200000062
set protocols bgp group underlay-bgp-wan neighbor 172.16.14.0 peer-as 4200000061
set policy-options policy-statement wan_import from community wan_underlay_comm
set policy-options policy-statement wan_import then local-preference subtract 10
set protocols bgp group underlay-bgp-wan import wan-import
set policy-options policy-statement underlay-clos-export-wan term loopback from interface lo0.0
set policy-options policy-statement underlay-clos-export-wan term loopback then community add
wan_underlay_comm
set policy-options policy-statement underlay-clos-export-wan term loopback then accept
set policy-options policy-statement underlay-clos-export-wan term def then reject
set policy-options community wan_underlay_comm members 12345:12345
set protocols bgp group underlay-bgp-wan export underlay-clos-export-wan
set protocols bgp group underlay-bgp-wan multipath multiple-as
set protocols bgp group underlay-bgp-wan bfd-liveness-detection minimum-interval 1000
set protocols bgp group underlay-bgp-wan bfd-liveness-detection multiplier 3
set protocols bgp group underlay-bgp-wan bfd-liveness-detection session-mode automatic

set protocols bgp group overlay-ebgp-extn-dci type external
set protocols bgp group overlay-ebgp-extn-dci multihop no-nexthop-change
set protocols bgp group overlay-ebgp-extn-dci local-address 192.168.4.3
set protocols bgp group overlay-ebgp-extn-dci family evpn signaling delay-route-advertisements
minimum-delay routing-uptime 400
set protocols bgp group overlay-ebgp-extn-dci family route-target external-paths 2
set protocols bgp group overlay-ebgp-extn-dci local-as 4210000002
set protocols bgp group overlay-ebgp-extn-dci multipath multiple-as
set protocols bgp group overlay-ebgp-extn-dci neighbor 192.168.4.1 peer-as 4210000001
set protocols bgp group overlay-ebgp-extn-dci neighbor 192.168.4.2 peer-as 4210000001
set protocols bgp group overlay-ebgp-extn-dci bfd-liveness-detection minimum-interval 4000
set protocols bgp group overlay-ebgp-extn-dci bfd-liveness-detection multiplier 3
set protocols bgp group overlay-ebgp-extn-dci bfd-liveness-detection session-mode automatic

set policy-options policy-statement underlay-clos-export term from_wan from community underlay-
```

```
clos-export
set policy-options policy-statement underlay-clos-export term from_wan then reject
```

Gateway 4 (POD 2)

```
set interfaces et-0/0/1 ether-options 802.3ad ae1
set interfaces ae1 unit 0 family inet address 172.16.11.1/31
set interfaces et-0/0/2 ether-options 802.3ad ae2
set interfaces ae2 unit 0 family inet address 172.16.13.1/31
set protocols bgp group underlay-bgp-wan type external
set protocols bgp group underlay-bgp-wan local-as 4200000034
set protocols bgp group underlay-bgp-wan neighbor 172.16.11.0 peer-as 4200000062
set protocols bgp group underlay-bgp-wan neighbor 172.16.12.0 peer-as 4200000061
set policy-options policy-statement wan_import from community wan_underlay_comm
set policy-options policy-statement wan_import then local-preference subtract 10
set protocols bgp group underlay-bgp-wan import wan-import
set policy-options policy-statement underlay-clos-export-wan term loopback from interface lo0.0
set policy-options policy-statement underlay-clos-export-wan term loopback then community add
wan_underlay_comm
set policy-options policy-statement underlay-clos-export-wan term loopback then accept
set policy-options policy-statement underlay-clos-export-wan term def then reject
set policy-options community wan_underlay_comm members 12345:12345
set protocols bgp group underlay-bgp-wan export underlay-clos-export-wan
set protocols bgp group underlay-bgp-wan multipath multiple-as
set protocols bgp group underlay-bgp-wan bfd-liveness-detection minimum-interval 1000
set protocols bgp group underlay-bgp-wan bfd-liveness-detection multiplier 3
set protocols bgp group underlay-bgp-wan bfd-liveness-detection session-mode automatic

set protocols bgp group overlay-ebgp-extn-dci type external
set protocols bgp group overlay-ebgp-extn-dci multihop no-nexthop-change
set protocols bgp group overlay-ebgp-extn-dci local-address 192.168.4.4
set protocols bgp group overlay-ebgp-extn-dci family evpn signaling delay-route-advertisements
minimum-delay routing-uptime 400
set protocols bgp group overlay-ebgp-extn-dci family route-target external-paths 2
set protocols bgp group overlay-ebgp-extn-dci local-as 4210000002
set protocols bgp group overlay-ebgp-extn-dci multipath multiple-as
set protocols bgp group overlay-ebgp-extn-dci neighbor 192.168.4.1 peer-as 4210000001
set protocols bgp group overlay-ebgp-extn-dci neighbor 192.168.4.2 peer-as 4210000001
set protocols bgp group overlay-ebgp-extn-dci bfd-liveness-detection minimum-interval 4000
set protocols bgp group overlay-ebgp-extn-dci bfd-liveness-detection multiplier 3
set protocols bgp group overlay-ebgp-extn-dci bfd-liveness-detection session-mode automatic
set policy-options policy-statement underlay-clos-export term from_wan from community underlay-
clos-export
set policy-options policy-statement underlay-clos-export term from_wan then reject
```

## Verify Underlay and Overlay Extension Over the WAN

This section shows how you verify the gateway devices are properly integrated into the WAN to extend the underlay and overlay networks between the two PODs.

1. Verify that the aggregated Ethernet interfaces are operational. Proper BGP session establishment is a good sign the interface can pass traffic. If in doubt, ping the remote end of the AE link.

```
user@spine1> show interfaces ae4
Physical interface: ae4, Enabled, Physical link is Up
  Interface index: 129, SNMP ifIndex: 544
  Link-level type: Ethernet, MTU: 9192, Speed: 40Gbps, BPDU Error: None, Ethernet-Switching
Error: None, MAC-REWRITE Error: None, Loopback: Disabled,
  Source filtering: Disabled, Flow control: Disabled, Minimum links needed: 1, Minimum
bandwidth needed: 1bps
  Device flags   : Present Running
  Interface flags: SNMP-Traps Internal: 0x4000
  Current address: 80:ac:ac:24:21:98, Hardware address: 80:ac:ac:24:21:98
  Last flapped   : 2020-07-30 13:09:31 PDT (3d 05:01 ago)
  Input rate     : 42963216 bps (30206 pps)
  Output rate    : 107152 bps (76 pps)

  Logical interface ae4.0 (Index 544) (SNMP ifIndex 564)
    Flags: Up SNMP-Traps 0x4004000 Encapsulation: ENET2
    Statistics        Packets        pps         Bytes          bps
    Bundle:
        Input :    7423834047      30126 1155962320326     37535088
        Output:     149534343         82   17315939427        83824
    Adaptive Statistics:
        Adaptive Adjusts:          0
        Adaptive Scans  :          0
        Adaptive Updates:          0
    Protocol inet, MTU: 9000
    Max nh cache: 75000, New hold nh limit: 75000, Curr nh cnt: 1, Curr new hold cnt: 0, NH
drop cnt: 0
      Flags: Sendbcast-pkt-to-re, Is-Primary, User-MTU
```

```
        Addresses, Flags: Is-Preferred Is-Primary
          Destination: 172.16.7.1/31, Local: 172.16.7.1
```

The output confirms that the aggregated Ethernet interface `ae4` is operational on gateway 1. The traffic counters also confirm the interface sends and receives packets.

2. Verify that the underlay EBGP sessions to the WAN devices are established.

```
user@spine1> show bgp summary group underlay-bgp-wan
Threading mode: BGP I/O
Default eBGP mode: advertise - accept, receive - accept
Groups: 5 Peers: 11 Down peers: 0
Table          Tot Paths  Act Paths Suppressed    History Damp State    Pending
bgp.rtarget.0
                    5038       5038          0          0          0          0
bgp.evpn.0
                  363629     190125          0          0          0          0
inet.0
                      24         12          0          0          0          0
Peer                     AS      InPkt     OutPkt    OutQ   Flaps Last Up/Dwn State|#Active/
Received/Accepted/Damped...
172.16.7.0       4200000061       3201       3123       0       0  1d 0:00:43 Establ
  inet.0: 3/4/4/0
172.16.9.0       4200000062       3200       3122       0       0  1d 0:00:24 Establ
  inet.0: 3/4/4/0
0
```

The output shows that both EBGP peering sessions on the gateway 1 device are established to both WAN routers.

3. Verify that the overlay EBGP sessions are established between the gateway devices across the WAN.

```
user@spine1> show bgp summary group overlay-ebgp-extn-dci
Threading mode: BGP I/O
Default eBGP mode: advertise - accept, receive - accept
Groups: 5 Peers: 11 Down peers: 0
Table          Tot Paths  Act Paths Suppressed    History Damp State    Pending
bgp.rtarget.0
                    5038       5038          0          0          0          0
bgp.evpn.0
                  363629     190125          0          0          0          0
inet.0
                      24         12          0          0          0          0
```

```
Peer                     AS      InPkt     OutPkt    OutQ   Flaps Last Up/Dwn State|#Active/
Received/Accepted/Damped...
192.168.4.3      4210000002       4376       3968       0       0  1d 0:03:30 Establ
  bgp.rtarget.0: 2519/2519/2519/0
  bgp.evpn.0: 43376/86752/86752/0
  MACVRF-mac-vrf-ep-t2-stchd-dhcp-1.evpn.0: 86/172/172/0
  MACVRF-mac-vrf-ep-t2-stchd-transl-1.evpn.0: 10002/20004/20004/0
  MACVRF-mac-vrf-ep-t2-stchd-1.evpn.0: 10002/20004/20004/0
  default-switch.evpn.0: 20086/40172/40172/0
  __default_evpn__.evpn.0: 0/0/0/0
192.168.4.4      4210000002      52575       9330       0       0  1d 0:03:30 Establ
  bgp.rtarget.0: 2519/2519/2519/0
  bgp.evpn.0: 43376/86752/86752/0
  MACVRF-mac-vrf-ep-t2-stchd-dhcp-1.evpn.0: 86/172/172/0
  MACVRF-mac-vrf-ep-t2-stchd-transl-1.evpn.0: 10002/20004/20004/0
  MACVRF-mac-vrf-ep-t2-stchd-1.evpn.0: 10002/20004/20004/0
  default-switch.evpn.0: 20086/40172/40172/0
  __default_evpn__.evpn.0: 0/0/0/0
```

The output confirms that both the overlay EBGP sessions from the gateway 1 device are established to both remote gateways in POD 2.

With underlay and overlay extension verified, you are ready to move onto configuring VXLAN stitching for Layer 2 DCI between the PODs.

## Configure Translational VXLAN Stitching DCI in the Default Switch Instance

In this section you configure VXLAN stitching in the gateway devices to provide Layer 2 stretch between the two PODs using the default switch instance. We support VXLAN stitching in the default switch instance and in MAC-VRF instances. We begin with the default switch instance, and later show the delta for the MAC-VRF instance case.

VXLAN stitching supports both a global mode and a translational mode. In the global mode, the VNI remains the same end-to-end, that is, across both the PODs and the WAN network. You use global mode when the VLAN and VNI assignments overlap between the PODs. In translational mode, you map a local POD VNI value to a VNI used across the WAN.

You configure VXLAN stitching only on the gateway devices. The leaf devices don't require any changes. In ERB fabrics, the lean spine devices also don't require any changes if you have a super-spine layer performing the gateway function.

Table 14 on page 380 outlines the POD VLAN and VNI assignments. In this example, the PODs use a different VNI for the same VLAN. This is why you configure translational stitching in this case. With translational stitching, the VNI can be unique to each POD and still be stitched to a shared VNI assignment over the WAN.

**Table 14: VLAN to VNI Mappings**

| POD 1 | POD 2 | WAN DCI |
|---|---|---|
| **VLAN 1** | | |
| VNI: 100001 | VNI: 110001 | VNI: 910001 |
| **VLAN 2** | | |
| VNI: 100002 | VNI: 110002 | VNI: 910002 |

Figure 83 on page 381 provides a high-level view of the VXLAN stitching plan for VLAN 1 in our example.

**Figure 83: Translational VXLAN Stitching Summary for VLAN 1**

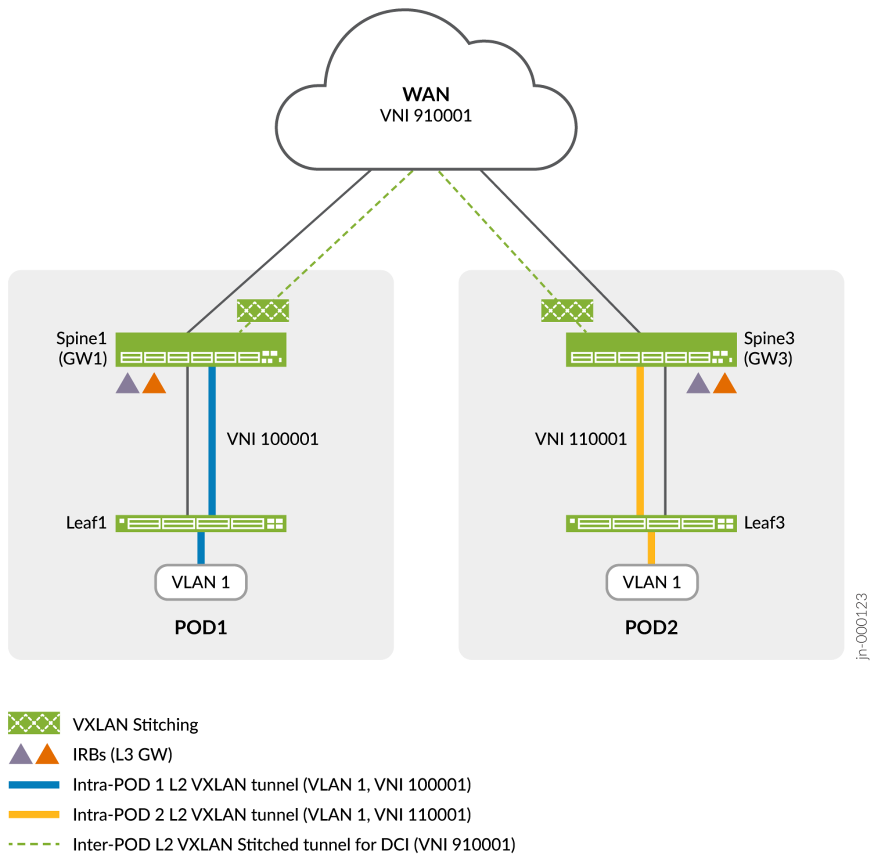

WAN
VNI 910001
Spine1
(GW1)
Spine3
(GW3)
VNI 100001
VNI 110001
Leaf1
Leaf3
VLAN 1
VLAN 1
POD1
POD2
jn-000123
VXLAN Stitching
IRBs (L3 GW)
Intra-POD 1 L2 VXLAN tunnel (VLAN 1, VNI 100001)
Intra-POD 2 L2 VXLAN tunnel (VLAN 1, VNI 110001)
Inter-POD L2 VXLAN Stitched tunnel for DCI (VNI 910001)

Figure 83 on page 381 shows VLAN 1 in POD 1 uses VNI 100001, while the same VLAN in POD 2 maps to 11000. You stitch both VLANs to a common VNI 910001 for transport over the WAN. When received from the WAN, the gateway translates the stitched VNI back to the VNI used locally within its POD.

Once again, the configuration on the gateway devices is similar. We walk you through the steps needed on the gateway 1 device, and provide the configuration delta for the other gateway nodes.

Perform these steps to configure translational VXLAN stitching on gateway 1.

**Gateway 1**

1. Configure the default switch instance EVPN parameters for route exchange between the two PODs. This configuration includes support for an all-active ESI LAG between the gateways. Setting up an ESI LAG over the WAN ensures that all WAN links are actively used to forward traffic without the risk of packet loops. You must use the same ESI value for all gateways within a given POD, and each POD must use a unique ESI value. Therefore, in this example you configure two unique ESIs, one for each pair of gateways in each POD.

The route target controls route imports. You configure the same route target on all gateway devices to ensure that all routes advertised by one POD are imported by the other. You set the route distinguisher to reflect each gateway device's loopback address.

```
set protocols evpn interconnect vrf-target target:60001:60001
set protocols evpn interconnect route-distinguisher 192.168.4.1:30000
set protocols evpn interconnect esi 00:00:ff:ff:00:11:00:00:00:01
set protocols evpn interconnect esi all-active
```

2. Configure VXLAN stitching for VLANs 1 and 2. You specify the VNIs that are stitched over the WAN at the `[edit protocols evpn interconnect interconnected-vni-list]` hierarchy. The gateway devices in both PODs must use the same VNI across the WAN for each stitched VNI.

```
set protocols evpn interconnect interconnected-vni-list 910001
set protocols evpn interconnect interconnected-vni-list 910002
```

3. Configure translational VXLAN stitching for VLAN 1 by linking the local VLAN/VNI to a translational VNI. Note that the translational VNI value matches the VNI you configured at the `protocols evpn interconnect interconnected-vni-list` hierarchy in the previous step. Thus, with the following commands you map a local VNI to a WAN VNI.

   For global VXLAN stitching, you simple omit the translational statement and configure the user VLAN to use the same VNI value you configure at the `[edit protocols evpn interconnect interconnected-vni-list]` hierarchy.

   Recall that the leaf and spine devices in each pod are already configured for the CRB reference architecture. As part of the pre-existing configuration, VLANs are defined on both the spine and leaf devices. The VLAN definition on all devices includes a VLAN ID to VXLAN VNI mapping. The spine's VLAN configuration differs from the leaf, however, in that it includes the Layer 3 IRB interface, again making this an example of CRB. The existing configuration for VLAN 1 is shown at the gateway 1 (spine 1) device for reference:

```
user@Spine-1> show configuration vlans EP-TYPE-2-VLAN-1
EP-TYPE-2-VLAN-1 {
           vlan-id 1;
           l3-interface irb.1;
           vxlan {
               vni 100001;
            }
       }
```

You now modify the configuration for VLAN 1 on the gateway 1 device to evoke translational VXLAN stitching. The VNI you specify matches one of the VNI values you configured at the `edit protocols evpn interconnect interconnected-vni-list` hierarchy in a previous step. The result is that the device translates VNI 100001 (used locally in POD 1 for VLAN 1) to VNI 910001 when sending it over the WAN. In the remote POD, a similar configuration maps from the WAN VNI back to the local VNI associated with the same VLAN in the remote POD. In configuration mode, enter the following command:

```
set vlans EP-TYPE-2-VLAN-1 vxlan translation-vni 910001
```

4. Configure translational VXLAN stitching for VLAN 2.

   You modify the configuration for VLAN 2 to invoke translational VXLAN stitching from VNI 100002 (used locally in POD 1 for VLAN 2) to VNI 910002 over the WAN.

```
set vlans EP-TYPE-2-VLAN-2 vxlan translation-vni 910002
```

5. Confirm the change for VLAN 1. We omit VLAN 2 for brevity. The following command displays the change to VLAN 1 in configuration mode:

```
[edit]
user@Spine-1# show vlans EP-TYPE-2-VLAN-1
EP-TYPE-2-VLAN-1 {
           vlan-id 1;
           l3-interface irb.1;
           vxlan {
               vni 100001;
           translation-vni 910001
            }
       }
```

6. Be sure to commit your changes on all gateway devices when done.

## Gateway Device Configurations for Translational VXLAN Stitching in Default Switch Instance

This section provides the configuration delta for all four gateway devices. You add this delta to the CRB baseline that you have modified for DCI over the WAN. Once you have extended the underlay and overlay, the following configurations perform translational VXLAN stitching between the local POD's VNI and the VNI on the WAN.

Gateway 1 (POD 1)

```
set protocols evpn interconnect vrf-target target:60001:60001
set protocols evpn interconnect route-distinguisher 192.168.4.1:30000
set protocols evpn interconnect esi 00:00:ff:ff:00:11:00:00:00:01
set protocols evpn interconnect esi all-active

set protocols evpn interconnect interconnected-vni-list 910001
set protocols evpn interconnect interconnected-vni-list 910002

set vlans EP-TYPE-2-VLAN-1 vxlan translation-vni 910001
set vlans EP-TYPE-2-VLAN-2 vxlan translation-vni 910002
```

Gateway 2 (Pod 1)

```
set protocols evpn interconnect vrf-target target:60001:60001
set protocols evpn interconnect route-distinguisher 192.168.4.2:30000
set protocols evpn interconnect esi 00:00:ff:ff:00:11:00:00:00:01
set protocols evpn interconnect esi all-active

set protocols evpn interconnect interconnected-vni-list 910001
set protocols evpn interconnect interconnected-vni-list 910002

set vlans EP-TYPE-2-VLAN-1 vxlan translation-vni 910001
set vlans EP-TYPE-2-VLAN-2 vxlan translation-vni 910002
```

Gateway 3 (POD 2)

```
set protocols evpn interconnect vrf-target target:60001:60001
set protocols evpn interconnect route-distinguisher 192.168.4.3:30000
set protocols evpn interconnect esi 00:00:ff:ff:00:22:00:00:00:01
set protocols evpn interconnect esi all-active

set protocols evpn interconnect interconnected-vni-list 910001
set protocols evpn interconnect interconnected-vni-list 910002
set vlans EP-TYPE-2-VLAN-1 vxlan translation-vni 910001
set vlans EP-TYPE-2-VLAN-2 vxlan translation-vni 910002
```

Gateway 4 (POD 2)

```
set protocols evpn interconnect vrf-target target:60001:60001
set protocols evpn interconnect route-distinguisher 192.168.4.4:30000
set protocols evpn interconnect esi 00:00:ff:ff:00:22:00:00:00:01
set protocols evpn interconnect esi all-active

set protocols evpn interconnect interconnected-vni-list 910001
set protocols evpn interconnect interconnected-vni-list 910002

set vlans EP-TYPE-2-VLAN-1 vxlan translation-vni 910001
set vlans EP-TYPE-2-VLAN-2 vxlan translation-vni 910002
```

## Verify Translational VXLAN Stitching in Default Switch Instance

1. Confirm the ESI LAG between the gateway devices is operational and in active-active mode.

```
user@Spine-1> show evpn instance default-switch esi 00:00:ff:ff:00:11:00:00:00:01 dci
extensive | except irb
Instance: default-switch
  Route Distinguisher: 192.168.4.1:20000
  Encapsulation type: VXLAN
  Duplicate MAC detection threshold: 5
  Duplicate MAC detection window: 180
  MAC database status                     Local  Remote
    MAC advertisements:                    1204    7596
    MAC+IP advertisements:                 6020   31494
    Default gateway MAC advertisements:    2408       0
  Number of local interfaces: 1 (1 up)
    Interface name  ESI                            Mode             Status     AC-Role
    .local..4       00:00:ff:ff:00:11:00:00:00:01  all-active       Up         Root
    Interface name  VLAN   VNI    Status  L3 context
  Number of protect interfaces: 0
  Number of bridge domains: 1204
  Number of neighbors: 6
    Address                 MAC    MAC+IP       AD        IM        ES Leaf-label Remote-DCI-
```

```
Peer
    192.168.0.1          1199      3167         8      1204         0
    192.168.0.2          1180      3395         6      1200         0
    192.168.0.3          1201      3562         0      1204         0
    192.168.4.2          2408      6020      1207      2008         0
DCI
    192.168.4.3           804     18468         2       804         0
DCI
    192.168.4.4           804     18468         2       804         0
DCI
  Number of ethernet segments: 1210
    ESI: 00:00:ff:ff:00:11:00:00:00:01 I-ESI
      Local interface: .local..4, Status: Up/Forwarding
      Number of remote PEs connected: 1
        Remote-PE        MAC-label  Aliasing-label  Mode
        192.168.4.2      0          0               all-active
      DF Election Algorithm: MOD based
      Designated forwarder: 192.168.4.1
      Backup forwarder: 192.168.4.2
      Last designated forwarder update: Sep 07 00:45:58
  Router-ID: 192.168.4.1
  SMET Forwarding: Disabled
  EVPN-Interconnect:
    Route-distinguisher: 192.168.4.1:30000
    Vrf-import: [ __evpn-ic-import-default-switch-internal__ ]
    Vrf-export: [ __evpn-ic-export-default-switch-internal__ ]
    Vrf-import-target: [ target:60001:60001 ]
    Vrf-export-target: [ target:60001:60001 ]
  DCI route stats                      Local
    AD  route advertisements:              1
    IM  route advertisements:            804
    MAC route advertisements:            805
    MAC+IP route advertisements:       29593
    ES  route advertisements:              0
    SG  Proxy route advertisements:        0
```

The output shows that ESI 00:00:ff:ff:00:11:00:00:00:01 is operational. The output also shows active-active forwarding (`Mode` column shows `all-active`) and both `Designated forwarder` and `Backup forwarder` device addresses.

2. View remote VXLAN VTEPs to confirm the remote gateway devices are listed as WAN VTEPs.

```
user@Spine-1> show ethernet-switching vxlan-tunnel-end-point remote summary instance default-
switch
Logical System Name       Id  SVTEP-IP         IFL   L3-Idx    SVTEP-Mode
<default>                         0  192.168.4.1      lo0.0    0
 RVTEP-IP         L2-RTT                   IFL-Idx   Interface    NH-Id   RVTEP-Mode Flags
 192.168.0.1      default-switch           3627      vtep.32792   83845   RNVE
 192.168.0.2      default-switch           3619      vtep.32784   69393   RNVE
 192.168.4.2      default-switch           3611      vtep.32779   42491   RNVE
 192.168.0.3      default-switch           3624      vtep.32789   74194   RNVE
 192.168.4.3      default-switch           3618      vtep.32783   51568   Wan-VTEP
 192.168.4.4      default-switch           3610      vtep.32775   42419   Wan-VTEP
```

The output correctly shows both remote gateways as `Wan-VTEP`.

3. View the EVPN database on the gateway 1 device for VXLAN VNI 100001. Recall that in our example this is the VNI you assigned to VLAN 1 on the CRB leaves and spines in POD 1.

```
user@Spine-1> show evpn database state dci-adv instance default-switch l2-domain-id 100001
Instance: default-switch
VLAN  DomainId  MAC address        Active source                  Timestamp        IP address
     100001     00:00:5e:00:00:04  05:fa:ef:80:81:00:01:86:a1:00  Sep 09 03:30:15  10.0.1.254

2001:db8::10:0:1:254
     100001     0c:59:9c:f2:60:00  192.168.4.2                    Sep 09 03:30:15  10.0.1.242

2001:db8::10:0:1:242

fe80::e59:9c00:1f2:6000
     100001     ba:31:2f:00:01:01  00:00:00:ff:00:01:00:03:00:05  Sep 09 04:51:03  10.0.1.1

2001:db8::10:0:1:1
     100001     ba:32:2f:00:01:01  192.168.0.3                    Sep 09 03:50:58  10.0.1.11

2001:db8::10:0:1:b
     100001     e8:a2:45:86:a8:00  irb.1                          Sep 09 03:29:48  10.0.1.241

2001:db8::10:0:1:241

fe80::eaa2:4500:186:a800
```

The output confirms the VNI value 100001 associated with VLAN 1 is advertised and used in the local POD.

4. View the EVPN database on the gateway 1 device for VXLAN VNI 910001. Recall that this is the VNI associated with VLAN 1 for translational VXLAN stitching over the WAN.

```
user@Spine-1> show evpn database state dc-adv instance default-switch l2-domain-id 910001
Instance: default-switch
VLAN  DomainId  MAC address        Active source                  Timestamp        IP address
     910001     00:00:5e:00:00:04  00:00:ff:ff:00:22:00:00:00:01  Sep 10 00:15:53  10.0.1.254

2001:db8::10:0:1:254
     910001     0c:59:9c:6f:7a:a0  00:00:ff:ff:00:22:00:00:00:01  Sep 10 00:15:51  10.0.1.245

2001:db8::10:0:1:245

fe80::e59:9c00:16f:7aa0
     910001     0c:59:9c:f9:8f:10  00:00:ff:ff:00:22:00:00:00:01  Sep 10 00:15:55  10.0.1.243

2001:db8::10:0:1:243

fe80::e59:9c00:1f9:8f10
     910001     4c:6d:58:00:00:00  00:00:ff:ff:00:22:00:00:00:01  Sep 10 00:15:50  10.0.1.246

2001:db8::10:0:1:246

fe80::4e6d:5800:100:0
     910001     50:c7:09:0a:85:60  00:00:ff:ff:00:22:00:00:00:01  Sep 10 00:15:53  10.0.1.244

2001:db8::10:0:1:244

fe80::52c7:900:10a:8560
     910001     be:31:2f:00:01:01  00:00:ff:ff:00:22:00:00:00:01  Sep 10 00:15:51  10.0.1.101

2001:db8::10:0:1:65

fe80::10:0:1:65
     910001     be:32:2f:00:01:01  00:00:ff:ff:00:22:00:00:00:01  Sep 10 00:15:52  10.0.1.111

2001:db8::10:0:1:6f

fe80::10:0:1:6f
     910001     c8:fe:6a:2d:56:00  00:00:ff:ff:00:22:00:00:00:01  Sep 10 00:15:49  10.0.1.247
```

```
2001:db8::10:0:1:247

fe80::cafe:6a00:12d:5600
```

The output confirms the VNI value 910001 associated with VLAN 1 is advertised to the remote POD. This confirms that VNI 910001 is used over the WAN. Given the local VNI differs from the VNI used on the WAN, this confirms translational VXLAN stitching for the default switch instance use case.

## VXLAN Stitching in a MAC-VRF Routing Instance

We support both global and translational VXLAN stitching in MAC-VRF routing instances. Because we demonstrated translational stitching for the previous default switch instance, for the MAC-VRF case we show global mode VXLAN stitching.

Coverage of MAC-VRF routing instances is beyond the scope of this document. Once again, we assume you have a working CRB fabric with MAC-VRF instances configured as per the reference baseline. For details on configuring MAC-VRF, see MAC-VRF Routing Instance Type Overview and a sample use case at EVPN-VXLAN DC IP Fabric MAC VRF L2 services.

To keep the focus on the VXLAN stitching feature, we call out the delta for adding VXLAN stitching to an existing MAC-VRF. As with the default switch instance, we apply the stitching configuration only to the gateway devices. In the case of MAC-VRF, however, you configure the VLAN to VNI mapping in the MAC-VRF instance, rather than at the `[edit vlans]` hierarchy. Another difference in the MAC-VRF case is that you configure the `interconnected-vni-list` statement in the routing instance instead of at the `[edit protocols evpn interconnect interconnected-vni-list]` hierarchy.

The goal in this example is to perform global VXLAN stitching for VLANs 1201 and 1202, which map to VXLAN VNIs 401201 and 401201, respectively. You configure the same VLAN to VNI mapping in both PODS. You can use global mode stitching because the VLAN to VNI assignments overlap in both PODs.

You add the following commands to the gateway devices for the MAC-VRF instance that will perform stitching. The configuration defines the ESI LAG used between the local gateways and specifies the list of interconnected VNIs.

You need a similar configuration on all gateway devices. As before, we walk though the configuration details for the gateway 1 device and then provide the complete configuration delta for the other gateways.

In the below example you configure VNIs 401201 and 401202 for VXLAN stitching over the WAN segment.

```
set routing-instances MACVRF-mac-vrf-ep-t2-stchd-1 protocols evpn interconnect vrf-target
target:60005:60001
set routing-instances MACVRF-mac-vrf-ep-t2-stchd-1 protocols evpn interconnect route-
distinguisher 192.168.4.1:46000
set routing-instances MACVRF-mac-vrf-ep-t2-stchd-1 protocols evpn interconnect esi
00:00:ff:ff:00:11:00:04:00:01
set routing-instances MACVRF-mac-vrf-ep-t2-stchd-1 protocols evpn interconnect esi all-active
set routing-instances MACVRF-mac-vrf-ep-t2-stchd-1 protocols evpn interconnect interconnected-
vni-list 401201
set routing-instances MACVRF-mac-vrf-ep-t2-stchd-1 protocols evpn interconnect interconnected-
vni-list 401202
```

**NOTE**: When configuring VXLAN stitching in a MAC-VRF context, you must include the `set forwarding-options evpn-vxlan shared-tunnels` option on all leaf nodes in the QFX5000 line of switches running Junos OS. After adding this statement, you must reboot the switch. We don't recommend using the `shared tunnels` statement on gateway nodes in the QFX10000 line of switches running Junos OS with VXLAN stitching in MAC-VRF routing instances.

Shared tunnels are enabled by default on devices running Junos OS Evolved (which supports EVPN-VXLAN only with MAC-VRF configurations).

As noted, a complete MAC-VRF routing instance configuration is beyond our scope. The configuration block below uses a pre-existing MAC-VRF instance based on the MAC-VRF reference design. We show this configuration snip to better illustrate why this is an example of global mode VXLAN stitching (for a MAC-VRF instance). The sample is from the CRB spine 1 device, which is also a gateway in our collapsed gateway example topology. For brevity, we only show the configuration for VLAN 1201.

```
user@Spine-1> show configuration routing-instances MACVRF-mac-vrf-ep-t2-stchd-1 vlans EP-TYPE-2-
VLAN-1201
vlan-id 1201;
l3-interface irb.1201;
vxlan {
    vni 401201;
}
```

In the above, the MAC-VRF definition for VLAN 1201 specifies the same VNI (401201) listed at the [edit routing-instances MACVRF-mac-vrf-ep-t2-stchd-1 protocols evpn interconnect interconnected-vni-list] hierarchy. This results in end-to-end (global) significance for that VNI.

As with the default switch instance, it is trivial to invoke translational VXLAN stitching in the MAC-VRF context.

For example, to translate from local VNI 300801 for VLAN 801 to a WAN VNI of 920001, you simply modify the VLAN definition in the related MAC-VRF instance to include the `translation-vni 920001` statement.

```
user@Spine-1>show routing-instances MACVRF-mac-vrf-ep-t2-stchd-transl-1 vlans EP-TYPE-2-VLAN-801
vlan-id 801;
l3-interface irb.801;
vxlan {
    vni 300801;
    translation-vni 920001
}
```

By adding the `translation-vni 920001` statement to the MAC-VRF VLAN configuration, you tell the gateway device to translate from local VNI 300801 to VNI 920001 when sending over the WAN.

## Gateway Device Configurations for Global VXLAN Stitching With MAC-VRF

This section provides the configuration delta for all four gateway devices to support global mode VXLAN stitching in a MAC-VRF context. You add this delta is added to the CRB baseline you modified for DCI over the WAN. After you extend the underlay and overlay, the configurations below perform global VXLAN stitching for VNIs 401201 and 401202. Because this is global mode example, you don't include the `translation-vni` statement. The VLAN and interconnect VNI values are the same.

Gateway 1 (POD 1)

```
set routing-instances MACVRF-mac-vrf-ep-t2-stchd-1 protocols evpn interconnect vrf-target
target:60005:60001
set routing-instances MACVRF-mac-vrf-ep-t2-stchd-1 protocols evpn interconnect route-
distinguisher 192.168.4.1:46000
set routing-instances MACVRF-mac-vrf-ep-t2-stchd-1 protocols evpn interconnect esi
00:00:ff:ff:00:11:00:04:00:01
set routing-instances MACVRF-mac-vrf-ep-t2-stchd-1 protocols evpn interconnect esi all-active
set routing-instances MACVRF-mac-vrf-ep-t2-stchd-1 protocols evpn interconnect interconnected-
vni-list 401201
set routing-instances MACVRF-mac-vrf-ep-t2-stchd-1 protocols evpn interconnect interconnected-
vni-list 401202

set routing-instances MACVRF-mac-vrf-ep-t2-stchd-1 vlans EP-TYPE-2-VLAN- vlan-id 1201
set routing-instances MACVRF-mac-vrf-ep-t2-stchd-1 vlans EP-TYPE-2-VLAN-1201 l3-interface
irb.1201
set routing-instances MACVRF-mac-vrf-ep-t2-stchd-1 vlans EP-TYPE-2-VLAN-1201 vxlan vni 401201
set routing-instances MACVRF-mac-vrf-ep-t2-stchd-1 vlans EP-TYPE-2-VLAN-1202 vlan-id 1202
set routing-instances MACVRF-mac-vrf-ep-t2-stchd-1 vlans EP-TYPE-2-VLAN-1202 l3-interface
irb.1202
set routing-instances MACVRF-mac-vrf-ep-t2-stchd-1 vlans EP-TYPE-2-VLAN-1202 vxlan vni 401202
```

Gateway 2 (Pod 1)

```
set routing-instances MACVRF-mac-vrf-ep-t2-stchd-1 protocols evpn interconnect vrf-target
target:60005:60001
set routing-instances MACVRF-mac-vrf-ep-t2-stchd-1 protocols evpn interconnect route-
distinguisher 192.168.4.2:46000
set routing-instances MACVRF-mac-vrf-ep-t2-stchd-1 protocols evpn interconnect esi
00:00:ff:ff:00:11:00:04:00:01
set routing-instances MACVRF-mac-vrf-ep-t2-stchd-1 protocols evpn interconnect esi all-active

set routing-instances MACVRF-mac-vrf-ep-t2-stchd-1 protocols evpn interconnect interconnected-
vni-list 401201
set routing-instances MACVRF-mac-vrf-ep-t2-stchd-1 protocols evpn interconnect interconnected-
vni-list 401202

set routing-instances MACVRF-mac-vrf-ep-t2-stchd-1 vlans EP-TYPE-2-VLAN- vlan-id 1201
set routing-instances MACVRF-mac-vrf-ep-t2-stchd-1 vlans EP-TYPE-2-VLAN-1201 l3-interface
irb.1201
set routing-instances MACVRF-mac-vrf-ep-t2-stchd-1 vlans EP-TYPE-2-VLAN-1201 vxlan vni 401201
set routing-instances MACVRF-mac-vrf-ep-t2-stchd-1 vlans EP-TYPE-2-VLAN-1202 vlan-id 1202
set routing-instances MACVRF-mac-vrf-ep-t2-stchd-1 vlans EP-TYPE-2-VLAN-1202 l3-interface
irb.1202
set routing-instances MACVRF-mac-vrf-ep-t2-stchd-1 vlans EP-TYPE-2-VLAN-1202 vxlan vni 401202
```

Gateway 3 (POD 2)

```
set routing-instances MACVRF-mac-vrf-ep-t2-stchd-1 protocols evpn interconnect vrf-target
target:60005:60001
set routing-instances MACVRF-mac-vrf-ep-t2-stchd-1 protocols evpn interconnect route-
distinguisher 192.168.4.3:46000
set routing-instances MACVRF-mac-vrf-ep-t2-stchd-1 protocols evpn interconnect esi
00:00:ff:ff:00:22:00:04:00:01
set routing-instances MACVRF-mac-vrf-ep-t2-stchd-1 protocols evpn interconnect esi all-active

set routing-instances MACVRF-mac-vrf-ep-t2-stchd-1 protocols evpn interconnect interconnected-
vni-list 401201
set routing-instances MACVRF-mac-vrf-ep-t2-stchd-1 protocols evpn interconnect interconnected-
vni-list 401202

set routing-instances MACVRF-mac-vrf-ep-t2-stchd-1 vlans EP-TYPE-2-VLAN- vlan-id 1201
set routing-instances MACVRF-mac-vrf-ep-t2-stchd-1 vlans EP-TYPE-2-VLAN-1201 l3-interface
irb.1201
set routing-instances MACVRF-mac-vrf-ep-t2-stchd-1 vlans EP-TYPE-2-VLAN-1201 vxlan vni 401201
set routing-instances MACVRF-mac-vrf-ep-t2-stchd-1 vlans EP-TYPE-2-VLAN-1202 vlan-id 1202
set routing-instances MACVRF-mac-vrf-ep-t2-stchd-1 vlans EP-TYPE-2-VLAN-1202 l3-interface
irb.1202
set routing-instances MACVRF-mac-vrf-ep-t2-stchd-1 vlans EP-TYPE-2-VLAN-1202 vxlan vni 401202
```

Gateway 4 (POD 2)

```
set routing-instances MACVRF-mac-vrf-ep-t2-stchd-1 protocols evpn interconnect vrf-target
target:60005:60001
set routing-instances MACVRF-mac-vrf-ep-t2-stchd-1 protocols evpn interconnect route-
distinguisher 192.168.4.4:46000
set routing-instances MACVRF-mac-vrf-ep-t2-stchd-1 protocols evpn interconnect esi
00:00:ff:ff:00:22:00:04:00:01
set routing-instances MACVRF-mac-vrf-ep-t2-stchd-1 protocols evpn interconnect esi all-active

set routing-instances MACVRF-mac-vrf-ep-t2-stchd-1 protocols evpn interconnect interconnected-
vni-list 401201
set routing-instances MACVRF-mac-vrf-ep-t2-stchd-1 protocols evpn interconnect interconnected-
vni-list 401202

set routing-instances MACVRF-mac-vrf-ep-t2-stchd-1 vlans EP-TYPE-2-VLAN- vlan-id 1201
set routing-instances MACVRF-mac-vrf-ep-t2-stchd-1 vlans EP-TYPE-2-VLAN-1201 l3-interface
```

```
irb.1201
set routing-instances MACVRF-mac-vrf-ep-t2-stchd-1 vlans EP-TYPE-2-VLAN-1201 vxlan vni 401201
set routing-instances MACVRF-mac-vrf-ep-t2-stchd-1 vlans EP-TYPE-2-VLAN-1202 vlan-id 1202
set routing-instances MACVRF-mac-vrf-ep-t2-stchd-1 vlans EP-TYPE-2-VLAN-1202 l3-interface
irb.1202
set routing-instances MACVRF-mac-vrf-ep-t2-stchd-1 vlans EP-TYPE-2-VLAN-1202 vxlan vni 401202
```

**NOTE**: When configuring VXLAN stitching in a MAC-VRF context, you must include the `set forwarding-options evpn-vxlan shared-tunnels` option on all leaf nodes in the QFX5000 line of switches. After adding this statement you must reboot the switch. We don't recommend configuring the shared tunnel statement on gateway nodes in the QFX10000 line of switches running Junos OS with VXLAN stitching in MAC-VRF routing instances.

Shared tunnels are enabled by default on devices running Junos OS Evolved (which supports EVPN-VXLAN only with MAC-VRF configurations).

## Verify Global VXLAN Stitching in a MAC-VRF Instance

1. Confirm the ESI LAG used between the gateway devices is operational and in active-active mode for the MAC-VRF case.

```
user@Spine-1> show evpn instance MACVRF-mac-vrf-ep-t2-stchd-1 esi
00:00:ff:ff:00:11:00:04:00:01 dci extensive | except irb
Instance: MACVRF-mac-vrf-ep-t2-stchd-1
  Route Distinguisher: 192.168.4.1:34501
  Encapsulation type: VXLAN
  Duplicate MAC detection threshold: 5
  Duplicate MAC detection window: 180
  MAC database status                     Local  Remote
    MAC advertisements:                     400    2795
    MAC+IP advertisements:                 2000   13395
    Default gateway MAC advertisements:     800       0
  Number of local interfaces: 1 (1 up)
    Interface name  ESI                            Mode             Status     AC-Role
    .local..5       00:00:ff:ff:00:11:00:04:00:01  all-active       Up         Root
    Interface name  VLAN   VNI    Status  L3 context
  Number of protect interfaces: 0
  Number of bridge domains: 400
  Number of neighbors: 6
    Address               MAC    MAC+IP        AD        IM        ES Leaf-label Remote-DCI-
Peer
    192.168.0.1           400      1145         2       400         0
    192.168.0.2           395      1134         2       400         0
    192.168.0.3           400      1050         0       400         0
    192.168.4.2           800      2000       403       800         0
DCI
    192.168.4.3           400      9200         2       400         0
DCI
    192.168.4.4           400      9200         2       400         0
DCI
  Number of ethernet segments: 403
    ESI: 00:00:ff:ff:00:11:00:04:00:01 I-ESI
      Local interface: .local..5, Status: Up/Forwarding
      Number of remote PEs connected: 1
        Remote-PE        MAC-label  Aliasing-label  Mode
        192.168.4.2      0          0               all-active
      DF Election Algorithm: MOD based
      Designated forwarder: 192.168.4.1
      Backup forwarder: 192.168.4.2
      Last designated forwarder update: Sep 07 00:46:00
  Router-ID: 192.168.4.1
  Source VTEP interface IP: 192.168.4.1
  SMET Forwarding: Disabled
  EVPN-Interconnect:
    Route-distinguisher: 192.168.4.1:46000
    Vrf-import: [ __evpn-ic-import-MACVRF-mac-vrf-ep-t2-stchd-1-internal__ ]
    Vrf-export: [ __evpn-ic-export-MACVRF-mac-vrf-ep-t2-stchd-1-internal__ ]
    Vrf-import-target: [ target:60005:60001 ]
    Vrf-export-target: [ target:60005:60001 ]
  DCI route stats                      Local
    AD  route advertisements:              1
    IM  route advertisements:            400
    MAC route advertisements:            400
    MAC+IP route advertisements:       14595
    ES  route advertisements:              0
    SG  Proxy route advertisements:        0
```

The output shows that ESI 00:00:ff:ff:00:11:00:00:00:01 is operational. Active-Active forwarding is verified by the `all-active` mode and the presence of both a designated and backup forwarder.

2. View remote VXLAN VTEPs to confirm the remote gateway devices are listed as WAN VTEPs.

```
user@Spine-1> show ethernet-switching vxlan-tunnel-end-point remote summary instance MACVRF-
mac-vrf-ep-t2-stchd-1
Logical System Name       Id  SVTEP-IP         IFL   L3-Idx    SVTEP-Mode
<default>                        0  192.168.4.1      lo0.0    0
 RVTEP-IP         L2-RTT                   IFL-Idx   Interface    NH-Id   RVTEP-Mode Flags
 192.168.0.1      default-switch           3627      vtep.32792   83845   RNVE
 192.168.0.2      default-switch           3619      vtep.32784   69393   RNVE
 192.168.4.2      default-switch           3611      vtep.32779   42491   RNVE
 192.168.0.3      default-switch           3624      vtep.32789   74194   RNVE
 192.168.4.3      default-switch           3618      vtep.32783   51568   Wan-VTEP
 192.168.4.4      default-switch           3610      vtep.32775   42419   Wan-VTEP
```

The output correctly shows both remote gateways as a `Wan-VTEP`.

3. View the EVPN database on the gateway 1 device for VXLAN VNI 401201 for advertisements to the WAN. In our example, this is the VNI assigned to VLAN 1201 in both PODs. As this is a CRB example, you defined VLAN 1201 on the spines and on the leaf devices. Only the spine devices include the Layer 3 IRB interfaces in their VLAN configurations, however.

```
user@Spine-1> show evpn database state dci-adv instance MACVRF-mac-vrf-ep-t2-stchd-1 l2-
domain-id 401201
Instance: MACVRF-mac-vrf-ep-t2-stchd-1
VLAN  DomainId  MAC address        Active source                  Timestamp        IP address
     401201     00:00:5e:00:00:04  05:fa:ef:80:81:00:06:1f:31:00  Sep 09 03:30:32
10.4.177.254

2001:db8::10:4:4b1:254
     401201     0c:59:9c:f2:60:00  192.168.4.2                    Sep 09 03:30:32
10.4.177.242

2001:db8::10:4:4b1:242

fe80::e59:9c04:b1f2:6000
     401201     ba:51:2f:04:b1:01  00:00:00:ff:00:01:00:05:00:07  Sep 09 07:08:47  10.4.177.1

2001:db8::10:4:4b1:1
     401201     ba:52:2f:04:b1:01  192.168.0.3                    Sep 09 03:50:58  10.4.177.11

2001:db8::10:4:4b1:b
     401201     e8:a2:45:86:a8:00  irb.1201                       Sep 09 03:29:44
```

```
10.4.177.241

2001:db8::10:4:4b1:241

fe80::eaa2:4504:b186:a800
```

The output confirms VNI value 401201, which is associated with VLAN 1201 and the irb.1201 interface, is advertised to the remote POD. This confirms that VNI 401201 is used over the WAN for VLAN 1201.

4. View the EVPN database on the gateway 1 device for VXLAN VNI 401201 for advertisements to the local POD. Recall this is the VNI associated with VLAN 1201 in both PODs. This is the same VNI you used in the previous command to display advertisements to the remote gateways.

```
user@Spine-1> show evpn database state dc-adv instance MACVRF-mac-vrf-ep-t2-stchd-1 l2-domain-
id 401201
Instance: MACVRF-mac-vrf-ep-t2-stchd-1
VLAN  DomainId  MAC address        Active source                  Timestamp        IP address
     401201     0c:59:9c:6f:7a:a0  00:00:ff:ff:00:22:00:04:00:01  Sep 09 23:55:02
10.4.177.245

2001:db8::10:4:4b1:245

fe80::e59:9c04:b16f:7aa0
     401201     0c:59:9c:f9:8f:10  00:00:ff:ff:00:22:00:04:00:01  Sep 09 23:55:06
10.4.177.243

2001:db8::10:4:4b1:243

fe80::e59:9c04:b1f9:8f10
     401201     4c:6d:58:00:00:00  00:00:ff:ff:00:22:00:04:00:01  Sep 09 23:55:02
10.4.177.246

2001:db8::10:4:4b1:246

fe80::4e6d:5804:b100:0
     401201     50:c7:09:0a:85:60  00:00:ff:ff:00:22:00:04:00:01  Sep 09 23:55:10
10.4.177.244

2001:db8::10:4:4b1:244

fe80::52c7:904:b10a:8560
     401201     be:51:2f:04:b1:01  00:00:ff:ff:00:22:00:04:00:01  Sep 09 23:55:03
```

```
10.4.177.101

2001:db8::10:4:4b1:65

fe80::40:0:4b1:65
     401201     be:52:2f:04:b1:01  00:00:ff:ff:00:22:00:04:00:01  Sep 09 23:55:03
10.4.177.111

2001:db8::10:4:4b1:6f

fe80::40:0:4b1:6f
     401201     c8:fe:6a:2d:56:00  00:00:ff:ff:00:22:00:04:00:01  Sep 09 23:54:58
10.4.177.247

2001:db8::10:4:4b1:247

fe80::cafe:6a04:b12d:5600
```

The output shows that VNI 401201 is advertised to the local POD. This confirms that VNI 401201 is used locally. Given the same VNI is used locally, and across the WAN, this confirms global VXLAN stitching in a MAC-VRF case.

## Virtual Machine Traffic Optimization (VMTO) with VXLAN Stitching

In some environments you may want to install /32 or /128 host routes to optimize traffic to a specific VM. When you use VXLAN stitching, configure the following on all gateway nodes to enable installation of host routes.

The first command adds host route support to the default switch instance. The second adds host route support for a specific MAC-VRF instance. You must configure both if you are using a mix of instance types.

```
set protocols evpn remote-ip-host-routes
set routing-instances MACVRF-mac-vrf-ep-t2-stchd-1 protocols evpn remote-ip-host-routes
```

## Verify Host Route Support

IN THIS SECTION

### Purpose

Confirm that /32 host routes are imported into a Layer 3 VRF table when using the default switch instance or a MAC-VRF table when using MAC-VRF.

### Action

Display the related routing instance's route table and look for routes with a /32 (or /128) bit prefix. We begin with the display of a Layer 3 VRF table used with VXLAN stitching, the default switch instance:

```
user@Spine-1> show route table VRF-ep-t2-stchd-transl-1.inet.0 protocol evpn
VRF-ep-t2-stchd-transl-1.inet.0: 52 destinations, 56 routes (52 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

10.0.1.1/32        *[EVPN/7] 23:54:12
                    >  via irb.1
10.0.1.11/32       *[EVPN/7] 23:54:12
                    >  via irb.1
10.0.1.101/32      *[EVPN/7] 23:54:12
                    >  via irb.1
10.0.1.111/32      *[EVPN/7] 23:54:12

. . .
```

Next, we display a MAC-VRF instance route table.

```
user@Spine-1> show route table VRF-mac-vrf-ep-t2-stchd-1.inet.0 protocol evpn
VRF-mac-vrf-ep-t2-stchd-1.inet.0: 52 destinations, 52 routes (52 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both
```

```
10.4.177.1/32      *[EVPN/7] 20:39:31
                    >  via irb.1201
10.4.177.11/32     *[EVPN/7] 23:57:20
                    >  via irb.1201
10.4.177.101/32    *[EVPN/7] 23:57:20
. . .
```

# EVPN Type 2 Symmetric Routing with DCI Stitching

**SUMMARY**

This document outlines the steps necessary to configure symmetric integrated routing and bridging (IRB) of Ethernet VPN (EVPN) Type 2 routes. Layer 2 Data Center Interconnect (DCI) gateway devices perform routing over stitched Virtual Extensible LAN (VXLAN) tunnels. Layer 2 DCI stitching is covered thoroughly in "Configure VXLAN Stitching for Layer 2 Data Center Interconnect" on page 361.

**IN THIS SECTION**

DCI enables you to segment the data center fabric into multiple points of delivery (PODs). Seamless stitching of VXLAN virtual network identifiers (VNIs) allows you to selectively stretch your Layer 2 network between PODs. Each POD follows a spine and leaf design. The leaf nodes in a given POD establish VXLAN tunnels only with leaves and spines in their POD. Spines transport traffic to other spines of different PODs. VXLAN stitching with DCI is ideal for larger networks because it reduces MAC flooding by reducing the required number of VXLAN tunnels between PODs.

Symmetric IRB in an EVPN-VXLAN environment occurs when the ingress and egress VXLAN tunnel end points (VTEPs) perform both routing and bridging on each side of the VXLAN tunnel. The ingress provider edge (PE) device performs a MAC lookup followed by an IP lookup. The egress PE performs the opposite, an IP lookup followed by a MAC lookup. Symmetric versus asymmetric models are covered in RFC 9135.

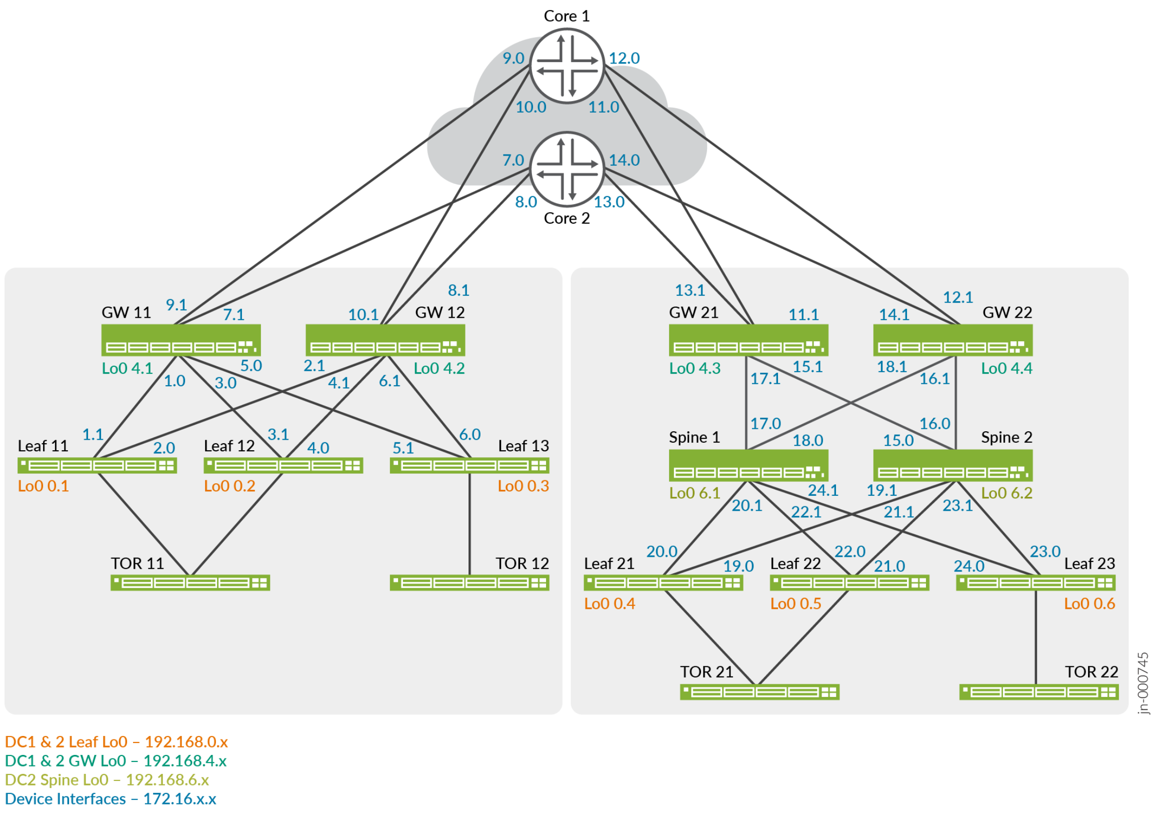
**NOTE**: Juniper Networks supports symmetric Type 2 routing with EVPN in:

- Networks using only symmetric Type 2 routing in PODs within the same DCI.
- EVPN-VXLAN environments.
- Edge-routed bridging (ERB) overlays.

Figure 84 on page 401 shows the topology for this example. The topology consists of two data centers connected over a WAN. Both data centers have similar ERB architectures. We refer to the data center on the left as DC1, and the data center on the right as DC2.

DC2 includes a layer of lean spine devices between the gateways and leaf layers. These lean spines are transit devices for the underlay. The topology uses EBGP for both the underlay and the overlay in both DC1 and DC2.

**Figure 84: Topology for DCI and Symmetric IRB Routing with EVPN Type 2 Routes**



## Inter-DC Underlay and Overlay

Configure the EBGP DCI underlay on DC2-GW21:

```
set protocols bgp group underlay-bgp-wan type external
set protocols bgp group underlay-bgp-wan import wan_import
set protocols bgp group underlay-bgp-wan export underlay-clos-export-wan
set protocols bgp group underlay-bgp-wan local-as 4200000033
set protocols bgp group underlay-bgp-wan multipath multiple-as
set protocols bgp group underlay-bgp-wan bfd-liveness-detection minimum-interval 1000
set protocols bgp group underlay-bgp-wan bfd-liveness-detection multiplier 3
set protocols bgp group underlay-bgp-wan bfd-liveness-detection session-mode automatic
set protocols bgp group underlay-bgp-wan neighbor 172.16.13.0 peer-as 4200000061
set protocols bgp group underlay-bgp-wan neighbor 172.16.11.0 peer-as 4200000062
set protocols bgp group underlay-bgp-wan log-updown
set protocols bgp group underlay-bgp-wan graceful-restart
```

```
set policy-options policy-statement wan_import from community wan_underlay_comm
set policy-options policy-statement wan_import then local-preference subtract 10
```

```
set policy-options policy-statement underlay-clos-export-wan term loopback from interface lo0.0
set policy-options policy-statement underlay-clos-export-wan term loopback then community add
wan_underlay_comm
set policy-options policy-statement underlay-clos-export-wan term loopback then accept
set policy-options policy-statement underlay-clos-export-wan term def then reject
set policy-options community wan_underlay_comm members 12345:12345
```

Verify the EBGP DCI underlay on DC2-GW21:

```
user@DC2-GW21> show bgp summary group underlay-bgp-wan

Threading mode: BGP I/O
Default eBGP mode: advertise - accept, receive - accept
Groups: 4 Peers: 8 Down peers: 0
Table          Tot Paths  Act Paths Suppressed    History Damp State    Pending
bgp.rtarget.0
                     727        515          0          0          0          0
bgp.evpn.0
```

```
                    124568      68684          0          0          0          0
inet.0
                        18         16          0          0          0          0
Peer                     AS      InPkt     OutPkt    OutQ   Flaps Last Up/Dwn State|#Active/
Received/Accepted/Damped...
172.16.11.0      4200000062        346        339       0       0     2:34:05 Establ
  inet.0: 3/4/4/0
172.16.13.0      4200000061        346        339       0       0     2:34:05 Establ
  inet.0: 3/4/4/0
```

Configure the EBGP DCI overlay on DC2-GW21:

```
set routing-options resolution rib bgp.rtarget.0 resolution-ribs inet.0
set protocols bgp group overlay-ebgp-extn-dci type external
set protocols bgp group overlay-ebgp-extn-dci multihop no-nexthop-change
set protocols bgp group overlay-ebgp-extn-dci local-address 192.168.4.3
set protocols bgp group overlay-ebgp-extn-dci family evpn signaling delay-route-advertisements
minimum-delay routing-uptime 400
set protocols bgp group overlay-ebgp-extn-dci family route-target external-paths 2
set protocols bgp group overlay-ebgp-extn-dci local-as 4210000002
set protocols bgp group overlay-ebgp-extn-dci multipath multiple-as
set protocols bgp group overlay-ebgp-extn-dci bfd-liveness-detection minimum-interval 1000
set protocols bgp group overlay-ebgp-extn-dci bfd-liveness-detection multiplier 3
set protocols bgp group overlay-ebgp-extn-dci bfd-liveness-detection session-mode automatic
set protocols bgp group overlay-ebgp-extn-dci neighbor 192.168.4.1 peer-as 4210000001
set protocols bgp group overlay-ebgp-extn-dci neighbor 192.168.4.2 peer-as 4210000001
```

Verify the EBGP DCI overlay on DC2-GW21:

```
user@DC2-GW21> show bgp summary group overlay-ebgp-extn-dci

Threading mode: BGP I/O
Default eBGP mode: advertise - accept, receive - accept
Groups: 4 Peers: 8 Down peers: 0
Table          Tot Paths  Act Paths Suppressed    History Damp State    Pending
bgp.rtarget.0
                       727        515          0          0          0          0
bgp.evpn.0
                    124568      68684          0          0          0          0
inet.0
                        18         16          0          0          0          0
```

```
Peer                     AS      InPkt     OutPkt    OutQ   Flaps Last Up/Dwn State|#Active/
Received/Accepted/Damped...
192.168.4.1      4210000001       3451       3397       0       0     2:34:58 Establ
  bgp.rtarget.0: 258/364/364/0
  bgp.evpn.0: 15468/23732/23732/0
  VRF-mac-vrf-ep-t2-symm-transl-1-1.evpn.0: 176/176/176/0
  VRF-mac-vrf-ep-t2-symm-transl-1-2.evpn.0: 141/141/141/0
  VRF-mac-vrf-ep-t2-symm-transl-1-3.evpn.0: 141/141/141/0
  VRF-mac-vrf-ep-t2-symm-transl-1-4.evpn.0: 141/141/141/0
  VRF-mac-vrf-ep-t2-symm-1-1.evpn.0: 141/141/141/0
  VRF-mac-vrf-ep-t2-symm-1-2.evpn.0: 141/141/141/0
  VRF-mac-vrf-ep-t2-symm-1-3.evpn.0: 141/141/141/0
  VRF-mac-vrf-ep-t2-symm-1-4.evpn.0: 141/141/141/0
  MACVRF-mac-vrf-ep-t2-symm-transl-1-1.evpn.0: 3034/4149/4149/0
  MACVRF-mac-vrf-ep-t2-symm-1-1.evpn.0: 3002/4104/4104/0
  MACVRF-mac-vrf-ep-t2-asymm-transl-2-1.evpn.0: 3002/6004/6004/0
  MACVRF-mac-vrf-ep-t2-asymm-2-1.evpn.0: 3002/6004/6004/0
  __default_evpn__.evpn.0: 0/0/0/0
  192.168.4.2      4210000001       3513       3274       0       0     2:34:54 Establ
  bgp.rtarget.0: 257/363/363/0
  bgp.evpn.0: 15468/23732/23732/0
  VRF-mac-vrf-ep-t2-symm-transl-1-1.evpn.0: 176/176/176/0
  VRF-mac-vrf-ep-t2-symm-transl-1-2.evpn.0: 141/141/141/0
  VRF-mac-vrf-ep-t2-symm-transl-1-3.evpn.0: 141/141/141/0
  VRF-mac-vrf-ep-t2-symm-transl-1-4.evpn.0: 141/141/141/0
  VRF-mac-vrf-ep-t2-symm-1-1.evpn.0: 141/141/141/0
  VRF-mac-vrf-ep-t2-symm-1-2.evpn.0: 141/141/141/0
  VRF-mac-vrf-ep-t2-symm-1-3.evpn.0: 141/141/141/0
  VRF-mac-vrf-ep-t2-symm-1-4.evpn.0: 141/141/141/0
  MACVRF-mac-vrf-ep-t2-symm-transl-1-1.evpn.0: 3034/4149/4149/0
  MACVRF-mac-vrf-ep-t2-symm-1-1.evpn.0: 3002/4104/4104/0
  MACVRF-mac-vrf-ep-t2-asymm-transl-2-1.evpn.0: 3002/6004/6004/0
  MACVRF-mac-vrf-ep-t2-asymm-2-1.evpn.0: 3002/6004/6004/0
  MACVRF-mac-vrf-ep-t2-symm-dhcp-1.evpn.0: 108/151/151/0
  __default_evpn__.evpn.0: 0/0/0/0
```

## Intra-DC Underlay and Overlay

Configure the EBGP Intra-DC underlay on DC2-GW21:

```
set protocols bgp group underlay-bgp type external
set protocols bgp group underlay-bgp export underlay-clos-export
set protocols bgp group underlay-bgp local-as 4200000033
set protocols bgp group underlay-bgp multipath multiple-as
set protocols bgp group underlay-bgp bfd-liveness-detection minimum-interval 1000
set protocols bgp group underlay-bgp bfd-liveness-detection multiplier 3
set protocols bgp group underlay-bgp bfd-liveness-detection session-mode automatic
set protocols bgp group underlay-bgp neighbor 172.16.15.0 peer-as 4200000042
set protocols bgp group underlay-bgp neighbor 172.16.17.0 peer-as 4200000042
set protocols bgp group underlay-bgp log-updown
set protocols bgp group underlay-bgp graceful-restart
set policy-options policy-statement underlay-clos-export term from_wan from community
wan_underlay_comm
set policy-options policy-statement underlay-clos-export term from_wan then reject
set policy-options policy-statement underlay-clos-export term loopback from interface lo0.0
set policy-options policy-statement underlay-clos-export term loopback then accept
set policy-options community wan_underlay_comm members 12345:12345
```

Verify the EBGP Intra-DC underlay on DC2-GW21:

```
user@DC2-GW21> show bgp summary group underlay-bgp

Threading mode: BGP I/O
Default eBGP mode: advertise - accept, receive - accept
Groups: 4 Peers: 8 Down peers: 0
Table          Tot Paths  Act Paths Suppressed    History Damp State    Pending
bgp.rtarget.0
                     727        515          0          0          0          0
bgp.evpn.0
                  124568      68684          0          0          0          0
inet.0
                      22         16          0          0          0          0
Peer                     AS      InPkt     OutPkt    OutQ   Flaps Last Up/Dwn State|#Active/
Received/Accepted/Damped...
172.16.15.0      4200000042        152        149       0       0     1:06:08 Establ
  inet.0: 5/7/7/0
```

```
172.16.17.0       4200000042        153        149       0       0     1:06:07 Establ
  inet.0: 5/7/7/0
```

## Layer 2 Symmetric IRB

Configure Layer 2 symmetric IRB interfaces on the DCI gateways Layer 3 virtual routing and forwarding (VRF) instance on DC2-GW21:

```
set routing-instances VRF-mac-vrf-ep-t2-symm-1-1 instance-type vrf
set routing-instances VRF-mac-vrf-ep-t2-symm-1-1 routing-options rib VRF-mac-vrf-ep-t2-
symm-1-1.inet6.0 multipath
set routing-instances VRF-mac-vrf-ep-t2-symm-1-1 routing-options multipath
set routing-instances VRF-mac-vrf-ep-t2-symm-1-1 protocols evpn irb-symmetric-routing vni 9100501
set routing-instances VRF-mac-vrf-ep-t2-symm-1-1 interface irb.501
set routing-instances VRF-mac-vrf-ep-t2-symm-1-1 interface irb.502
set routing-instances VRF-mac-vrf-ep-t2-symm-1-1 interface irb.503
set routing-instances VRF-mac-vrf-ep-t2-symm-1-1 interface irb.504
set routing-instances VRF-mac-vrf-ep-t2-symm-1-1 interface lo0.501
set routing-instances VRF-mac-vrf-ep-t2-symm-1-1 route-distinguisher 192.168.4.3:501
set routing-instances VRF-mac-vrf-ep-t2-symm-1-1 vrf-target import target:400:501
set routing-instances VRF-mac-vrf-ep-t2-symm-1-1 vrf-target export target:500:501
set routing-instances VRF-mac-vrf-ep-t2-symm-1-1 vrf-table-label
```

Verify symmetric routes on DC2-GW21 for the remote gateway. Symmetric routes will have a protocol preference of 7, while asymmetric routes will have a protocol preference of 170:

```
user@DC2-GW21> show route 10.20.245.1

VRF-mac-vrf-ep-t2-symm-1-1.inet.0: 68 destinations, 208 routes (68 active, 0 holddown, 0 hidden)
@ = Routing Use Only, # = Forwarding Use Only
+ = Active Route, - = Last Active, * = Both

10.20.245.1/32      @[EVPN/7] 00:04:36
                    >  to 172.16.11.0 via ae1.0
                       to 172.16.13.0 via irb.4004
                    [EVPN/7] 00:04:36
                    >  to 172.16.11.0 via ae1.0
                       to 172.16.13.0 via irb.4004
                    [EVPN/170] 00:04:36
                    >  to 172.16.11.0 via ae1.0
                       to 172.16.13.0 via irb.4004
                    [EVPN/170] 00:04:36
                    >  to 172.16.11.0 via ae1.0
                       to 172.16.13.0 via irb.4004
                   #[Multipath/255] 00:04:36, metric2 0
                    >  to 172.16.11.0 via ae1.0
                       to 172.16.13.0 via irb.4004
                    >  to 172.16.11.0 via ae1.0
                       to 172.16.13.0 via irb.4004
```

Layer 2 symmetric routes carry the Layer 2 and Layer 3 VNIs along with the Type 2 DCI route target (RT) and Type 5 stitching interconnect RT. They also carry router MAC addresses and interconnect Ethernet segment identifiers (iESI) details:

```
user@DC2-GW21> show route table bgp.evpn.0 match-prefix
2:192.168.4.1:38000::200501::ba:31:2f:01:f5:01::10.20.245.1/304 extensive

bgp.evpn.0: 104886 destinations, 160122 routes (104082 active, 0 holddown, 1608 hidden)
2:192.168.4.1:38000::200501::ba:31:2f:01:f5:01::10.20.245.1/304 MAC/IP (1 entry, 0 announced)
        *BGP    Preference: 170/-101
                Route Distinguisher: 192.168.4.1:38000
                Next hop type: Indirect, Next hop index: 0
                Address: 0x5601d3366b9c
                Next-hop reference count: 1067
                Kernel Table Id: 0
                Source: 192.168.4.1
                Protocol next hop: 192.168.4.1
                Label operation: Push 12531
                Label TTL action: prop-ttl
                Load balance label: Label 12531: None;
                Indirect next hop: 0x2 no-forward INH Session ID: 0
                State: <Active Ext>
                Peer AS: 4210000001
                Age: 7:41       Metric2: 0
                Validation State: unverified
                Task: BGP_4210000001_42100000.192.168.4.1
                AS path: 4210000001 I
                Communities: target:5003:6001 target:60003:60001 encapsulation:vxlan(0x8) router-
mac:4c:73:4f:e6:4f:80
                Import Accepted
                Route Label: 200501
                Route Label: 9100501
                ESI: 00:00:ff:ff:00:11:00:02:00:01
                Localpref: 100
                Router ID: 192.168.4.1
                Secondary Tables: VRF-mac-vrf-ep-t2-symm-1-1.evpn.0 MACVRF-mac-vrf-ep-t2-
symm-1-1.evpn.0
                Thread: junos-main
                Indirect next hops: 1
                        Protocol next hop: 192.168.4.1 ResolvState: Resolved
                        Label operation: Push 12531
                        Label TTL action: prop-ttl
                        Load balance label: Label 12531: None;
                        Indirect next hop: 0x2 no-forward INH Session ID: 0
                        Indirect path forwarding next hops: 2
                                Next hop type: Router
                                Next hop: 172.16.11.0 via ae1.0
                                Session Id: 0
                                Next hop: 172.16.13.0 via irb.4004
                                Session Id: 0
                                192.168.4.1/32 Originating RIB: inet.0
                                  Node path count: 1
                                  Forwarding nexthops: 2
                                        Next hop type: Router
                                        Next hop: 172.16.11.0 via ae1.0
                                        Session Id: 0
                                        Next hop: 172.16.13.0 via irb.4004
                                        Session Id: 0
user@DC2-GW21> show route table bgp.evpn.0 match-prefix
2:192.168.4.2:38000::200501::ba:31:2f:01:f5:01::10.20.245.1/304 extensive

bgp.evpn.0: 104886 destinations, 160122 routes (104082 active, 0 holddown, 1608 hidden)
2:192.168.4.2:38000::200501::ba:31:2f:01:f5:01::10.20.245.1/304 MAC/IP (1 entry, 0 announced)
        *BGP    Preference: 170/-101
                Route Distinguisher: 192.168.4.2:38000
                Next hop type: Indirect, Next hop index: 0
                Address: 0x5601cdb53edc
                Next-hop reference count: 1067
                Kernel Table Id: 0
                Source: 192.168.4.2
                Protocol next hop: 192.168.4.2
                Label operation: Push 12531
                Label TTL action: prop-ttl
                Load balance label: Label 12531: None;
                Indirect next hop: 0x2 no-forward INH Session ID: 0
                State: <Active Ext>
                Peer AS: 4210000001
                Age: 27:27      Metric2: 0
                Validation State: unverified
                Task: BGP_4210000001_42100000.192.168.4.2
                AS path: 4210000001 I
                Communities: target:5003:6001 target:60003:60001 encapsulation:vxlan(0x8) router-
mac:74:e7:98:61:44:76
                Import Accepted
                Route Label: 200501
                Route Label: 9100501
                ESI: 00:00:ff:ff:00:11:00:02:00:01
                Localpref: 100
                Router ID: 192.168.4.2
                Secondary Tables: VRF-mac-vrf-ep-t2-symm-1-1.evpn.0 MACVRF-mac-vrf-ep-t2-
symm-1-1.evpn.0
                Thread: junos-main
                Indirect next hops: 1
                        Protocol next hop: 192.168.4.2 ResolvState: Resolved
                        Label operation: Push 12531
                        Label TTL action: prop-ttl
                        Load balance label: Label 12531: None;
                        Indirect next hop: 0x2 no-forward INH Session ID: 0
                        Indirect path forwarding next hops: 2
                                Next hop type: Router
                                Next hop: 172.16.11.0 via ae1.0
                                Session Id: 0
                                Next hop: 172.16.13.0 via irb.4004
                                Session Id: 0
                                192.168.4.2/32 Originating RIB: inet.0
                                  Node path count: 1
                                  Forwarding nexthops: 2
                                        Next hop type: Router
                                        Next hop: 172.16.11.0 via ae1.0
                                        Session Id: 0
                                        Next hop: 172.16.13.0 via irb.4004
                                        Session Id: 0
```

## Layer 2 Data Center Interconnect

Configure the VLANs for symmetric routing and seamless stitching. Seamless stitching requires that interconnected gateway devices use the same `translation-vni`.

**NOTE**: You are required to configure `set forwarding-options evpn-vxlan vxlan-trans-vni-enable` on QFX5120 series switches to enable Layer 2 stitching. The packet forwarding engine (PFE) will restart once this configuration is applied, resulting in a restart of the associated FPC and interfaces.

**NOTE**: You are required to configure `set system packet-forwarding-options system-profile vxlan-stitching` on ACX Series routers to enable Layer 2 stitching. The PFE will restart once this configuration is applied.

**NOTE**: You can configure the following parameters on ACX Series routers to enable load balancing based on traffic payload.

```
set forwarding-options hash-key family inet layer-3
set forwarding-options hash-key family inet layer-4
set forwarding-options hash-key family inet6 layer-3
set forwarding-options hash-key family inet6 layer-4
set forwarding-options hash-key family multiservice source-mac
set forwarding-options hash-key family multiservice destination-mac
```

Configure the translation VNI on DC2-GW21 for all gateway devices.

```
set routing-instances MACVRF-mac-vrf-ep-t2-symm-transl-1-1 vlans EP-TYPE-2-VLAN-2 vlan-id 2
set routing-instances MACVRF-mac-vrf-ep-t2-symm-transl-1-1 vlans EP-TYPE-2-VLAN-2 l3-interface
irb.2
set routing-instances MACVRF-mac-vrf-ep-t2-symm-transl-1-1 vlans EP-TYPE-2-VLAN-2 vxlan vni
120002
set routing-instances MACVRF-mac-vrf-ep-t2-symm-transl-1-1 vlans EP-TYPE-2-VLAN-2 vxlan
translation-vni 920001
set routing-instances MACVRF-mac-vrf-ep-t2-symm-transl-1-1 vlans EP-TYPE-2-VLAN-3 vlan-id 3
set routing-instances MACVRF-mac-vrf-ep-t2-symm-transl-1-1 vlans EP-TYPE-2-VLAN-3 l3-interface
irb.3
set routing-instances MACVRF-mac-vrf-ep-t2-symm-transl-1-1 vlans EP-TYPE-2-VLAN-3 vxlan vni
120003
```

```
set routing-instances MACVRF-mac-vrf-ep-t2-symm-transl-1-1 vlans EP-TYPE-2-VLAN-3 vxlan
translation-vni 920002
set routing-instances MACVRF-mac-vrf-ep-t2-symm-transl-1-1 vlans EP-TYPE-2-VLAN-4 vlan-id 4
set routing-instances MACVRF-mac-vrf-ep-t2-symm-transl-1-1 vlans EP-TYPE-2-VLAN-4 l3-interface
irb.4
set routing-instances MACVRF-mac-vrf-ep-t2-symm-transl-1-1 vlans EP-TYPE-2-VLAN-4 vxlan vni
120004
set routing-instances MACVRF-mac-vrf-ep-t2-symm-transl-1-1 vlans EP-TYPE-2-VLAN-4 vxlan
translation-vni 920003
set routing-instances MACVRF-mac-vrf-ep-t2-symm-transl-1-1 vlans EP-TYPE-2-VLAN-5 vlan-id 5
set routing-instances MACVRF-mac-vrf-ep-t2-symm-transl-1-1 vlans EP-TYPE-2-VLAN-5 l3-interface
irb.5
set routing-instances MACVRF-mac-vrf-ep-t2-symm-transl-1-1 vlans EP-TYPE-2-VLAN-5 vxlan vni
120005
set routing-instances MACVRF-mac-vrf-ep-t2-symm-transl-1-1 vlans EP-TYPE-2-VLAN-5 vxlan
translation-vni 920004
```

Verify the EVPN routes on DC2-GW21 advertised to the DCI. Show the EVPN database for the specified MAC-VRF and VNI. The `dci-adv` option is specific to showing MAC advertisements to DCI gateway nodes:

```
user@DC2-GW21> show evpn database state dci-adv instance MACVRF-mac-vrf-ep-t2-symm-transl-1-1 l2-
domain-id 120002

Instance: MACVRF-mac-vrf-ep-t2-symm-transl-1-1
VLAN  DomainId  MAC address        Active source                  Timestamp        IP address
     120002     0c:59:9c:6f:7a:a0  192.168.0.4                    Oct 06 04:33:27  10.12.2.248

2001:db8::10:12:2:248

fe80::e59:9c00:26f:7aa0
     120002     4c:6d:58:00:00:00  192.168.0.5                    Oct 06 04:33:21  10.12.2.249

2001:db8::10:12:2:249

fe80::4e6d:5800:200:0
     120002     68:22:8e:e3:ad:df  192.168.4.4                    Oct 06 04:33:22  10.12.2.247

2001:db8::10:12:2:247

fe80::6a22:8e00:2e3:addf
     120002     be:51:2f:00:02:01  00:00:00:ff:00:11:00:01:00:03  Oct 06 04:33:27  10.12.2.101
```

```
2001:db8::10:12:2:65
     120002     be:52:2f:00:02:01  192.168.0.6                    Oct 06 04:33:28  10.12.2.111

2001:db8::10:12:2:6f
     120002     be:53:2f:00:02:01  et-0/0/8:0.0                   Oct 06 05:52:00  10.12.2.2
     120002     c8:fe:6a:2d:56:00  192.168.0.6                    Oct 06 04:33:27  10.12.2.250

2001:db8::10:12:2:250

fe80::cafe:6a00:22d:5600
     120002     f8:c1:16:01:28:07  irb.2                          Oct 06 05:52:00  10.12.2.246

2001:db8::10:12:2:246

fe80::fac1:1600:201:2807
```

Verify the EVPN routes on DC2-GW21 advertised to the data center. Show the EVPN database for the specified MAC-VRF and VNI. The `dc-adv` statement is specific to showing MAC advertisements to data center gateway nodes:

```
user@DC2-GW21> show evpn database state dc-adv instance MACVRF-mac-vrf-ep-t2-symm-transl-1-1 l2-
domain-id 920001

Instance: MACVRF-mac-vrf-ep-t2-symm-transl-1-1
VLAN  DomainId  MAC address        Active source                  Timestamp        IP address
     920001     0c:59:9c:8c:46:e4  00:00:ff:ff:00:11:00:04:00:01  Oct 06 04:33:26  10.11.2.245

2001:db8::10:11:2:245

fe80::e59:9c00:28c:46e4
     920001     4c:73:4f:e6:4f:80  00:00:ff:ff:00:11:00:04:00:01  Oct 06 04:33:26  10.11.2.241

2001:db8::10:11:2:241

fe80::4e73:4f00:2e6:4f80
     920001     74:e7:98:61:44:76  00:00:ff:ff:00:11:00:04:00:01  Oct 06 04:33:27  10.11.2.242

2001:db8::10:11:2:242

fe80::76e7:9800:261:4476
     920001     a4:51:5e:3e:1a:0e  00:00:ff:ff:00:11:00:04:00:01  Oct 06 04:33:27  10.11.2.243
```

```
2001:db8::10:11:2:243

fe80::a651:5e00:23e:1a0e
     920001     ba:51:2f:00:02:01  00:00:ff:ff:00:11:00:04:00:01  Oct 06 04:33:27  10.11.2.1

2001:db8::10:11:2:1
     920001     ba:52:2f:00:02:01  00:00:ff:ff:00:11:00:04:00:01  Oct 06 04:33:27  10.11.2.11

2001:db8::10:11:2:b
     920001     e8:a2:45:e2:a6:b0  00:00:ff:ff:00:11:00:04:00:01  Oct 06 04:33:27  10.11.2.244

2001:db8::10:11:2:244

fe80::eaa2:4500:2e2:a6b0
```

Configure the `interconnect` `vrf-target` and `route-distinguisher` for DC1 interconnect gateway (iGW) devices on DC1-GW12. These must be different from the `vrf-target` and `route-distinguisher` configured for the local DC:

```
set routing-instances MACVRF-mac-vrf-ep-t2-symm-transl-1-1 protocols evpn interconnect vrf-
target target:60005:60001
set routing-instances MACVRF-mac-vrf-ep-t2-symm-transl-1-1 protocols evpn interconnect route-
distinguisher 192.168.4.2:46000
set routing-instances MACVRF-mac-vrf-ep-t2-symm-transl-1-1 protocols evpn interconnect esi
00:00:ff:ff:00:11:00:04:00:01
set routing-instances MACVRF-mac-vrf-ep-t2-symm-transl-1-1 protocols evpn interconnect esi all-
active
```

Add the previously defined translation VNIs to the DC1 `interconnect` configuration under the MAC-VRF:

```
set routing-instances MACVRF-mac-vrf-ep-t2-symm-transl-1-1 protocols evpn interconnect
interconnected-vni-list 920001
set routing-instances MACVRF-mac-vrf-ep-t2-symm-transl-1-1 protocols evpn interconnect
interconnected-vni-list 920002
set routing-instances MACVRF-mac-vrf-ep-t2-symm-transl-1-1 protocols evpn interconnect
interconnected-vni-list 920003
set routing-instances MACVRF-mac-vrf-ep-t2-symm-transl-1-1 protocols evpn interconnect
interconnected-vni-list 920004
```

Configure the `interconnect` `vrf-target` and `route-distinguisher` for DC2 iGW devices. These must be different from the `vrf-target` and `route-distinguisher` configured for the local DC:

```
set routing-instances MACVRF-mac-vrf-ep-t2-symm-transl-1-1 protocols evpn interconnect vrf-
target target:60005:60001
set routing-instances MACVRF-mac-vrf-ep-t2-symm-transl-1-1 protocols evpn interconnect route-
distinguisher 192.168.4.3:46000
set routing-instances MACVRF-mac-vrf-ep-t2-symm-transl-1-1 protocols evpn interconnect esi
00:00:ff:ff:00:22:00:04:00:01
set routing-instances MACVRF-mac-vrf-ep-t2-symm-transl-1-1 protocols evpn interconnect esi all-
active
```

Add the previously defined translation VNIs to the DC2 `interconnect` configuration under the MAC-VRF:

```
set routing-instances MACVRF-mac-vrf-ep-t2-symm-transl-1-1 protocols evpn interconnect
interconnected-vni-list 920001
set routing-instances MACVRF-mac-vrf-ep-t2-symm-transl-1-1 protocols evpn interconnect
interconnected-vni-list 920002
set routing-instances MACVRF-mac-vrf-ep-t2-symm-transl-1-1 protocols evpn interconnect
interconnected-vni-list 920003
set routing-instances MACVRF-mac-vrf-ep-t2-symm-transl-1-1 protocols evpn interconnect
interconnected-vni-list 920004
```

Verify the remote VTEP on DC1-GW12:

```
user@DC1-GW12> show ethernet-switching vxlan-tunnel-end-point remote summary instance MACVRF-mac-
vrf-ep-t2-symm-transl-1-1

Logical System Name       Id  SVTEP-IP         IFL   L3-Idx    SVTEP-Mode    ELP-SVTEP-IP
<default>                 0   192.168.4.2      lo0.0    0
 RVTEP-IP         L2-RTT                   IFL-Idx   Interface    NH-Id   RVTEP-Mode  ELP-
IP        Flags
 192.168.0.1      MACVRF-mac-vrf-ep-t2-symm-transl-1-1 671088664 vtep-189.32778 92700 RNVE
 192.168.4.1      MACVRF-mac-vrf-ep-t2-symm-transl-1-1 671088659 vtep-189.32777 92699 I-ESI-Peer
 192.168.0.2      MACVRF-mac-vrf-ep-t2-symm-transl-1-1 671088654 vtep-189.32776 94266 RNVE
 192.168.0.3      MACVRF-mac-vrf-ep-t2-symm-transl-1-1 671088669 vtep-189.32779 123079 RNVE
 192.168.4.3      MACVRF-mac-vrf-ep-t2-symm-transl-1-1 671088649 vtep-189.32775 97029 Wan-VTEP
 192.168.4.4      MACVRF-mac-vrf-ep-t2-symm-transl-1-1 671088644 vtep-189.32774 97028 Wan-VTEP
```

Verify Type 2 stitching on DC1-GW12:

```
user@DC1-GW12> show evpn instance MACVRF-mac-vrf-ep-t2-symm-transl-1-1 esi
00:00:ff:ff:00:11:00:04:00:01 dci extensive | except irb

Instance: MACVRF-mac-vrf-ep-t2-symm-transl-1-1
  Route Distinguisher: 192.168.4.2:33302
  Encapsulation type: VXLAN
  Control word enabled
  Duplicate MAC detection threshold: 5
  Duplicate MAC detection window: 180
  MAC database status                    Local  Remote
    MAC advertisements:                    101    2426
    MAC+IP advertisements:                 505    3939
    Default gateway MAC advertisements:    202       0
  Number of local interfaces: 1 (1 up)
    Interface name  ESI                            Mode             Status     AC-Role
    .local..189     00:00:ff:ff:00:11:00:04:00:01  all-active       Up         Root
    Interface name  VLAN   VNI    Status  L3 context
  Number of protect interfaces: 0
  Number of bridge domains: 101
  Number of neighbors: 6
    Address               MAC    MAC+IP        AD        IM        ES Leaf-label Remote-DCI-Peer
Flow-label
    192.168.0.1           204       505         2       101         0
NO
    192.168.0.2           202       505         2       101         0
NO
    192.168.0.3           202       505         0       101         0
NO
    192.168.4.1           202       505       104       202         0
NO
    192.168.4.3           808      2121         2       101         0            DCI
NO
    192.168.4.4           808      2121         2       101         0            DCI
NO
  Number of ethernet segments: 104
    ESI: 00:00:ff:ff:00:11:00:04:00:01 I-ESI
      Local interface: .local..189, Status: Up/Forwarding
      Number of remote PEs connected: 1
        Remote-PE        MAC-label  Aliasing-label  Mode
        192.168.4.1      0          0               all-active
      DF Election Algorithm: MOD based
      Designated forwarder: 192.168.4.1
      Backup forwarder: 192.168.4.2
      Last designated forwarder update: Oct 05 23:33:14
  Router-ID: 192.168.4.2
  Source VTEP interface IP: 192.168.4.2
  SMET Forwarding: Disabled
  EVPN-Interconnect:
    Route-distinguisher: 192.168.4.2:46000
    Vrf-import: [ __evpn-ic-import-MACVRF-mac-vrf-ep-t2-symm-transl-1-1-internal__ ]
    Vrf-export: [ __evpn-ic-export-MACVRF-mac-vrf-ep-t2-symm-transl-1-1-internal__ ]
    Vrf-import-target: [ target:60005:60001 ]
    Vrf-export-target: [ target:60005:60001 ]
  DCI route stats                      Local
    AD  route advertisements:               1
    IM  route advertisements:             101
    MAC route advertisements:            1517
    MAC+IP route advertisements:         4040
    ES  route advertisements:               0
    SG  Proxy route advertisements:         0
```

Verify the remote VTEP on DC2-GW21:

```
user@DC2-GW21> show ethernet-switching vxlan-tunnel-end-point remote summary instance MACVRF-mac-
vrf-ep-t2-symm-transl-1-1

Logical System Name       Id  SVTEP-IP         IFL    L3-Idx    SVTEP-Mode    ELP-SVTEP-IP
<default>                 0   192.168.4.3      lo0.0    0
 RVTEP-IP         L2-RTT                   IFL-Idx   Interface    NH-Id   RVTEP-Mode  ELP-
IP        Flags
 192.168.4.1      MACVRF-mac-vrf-ep-t2-symm-transl-1-1 671088654 vtep-207.32776 93880 Wan-VTEP
 192.168.4.2      MACVRF-mac-vrf-ep-t2-symm-transl-1-1 671088659 vtep-207.32777 125332 Wan-VTEP
 192.168.0.4      MACVRF-mac-vrf-ep-t2-symm-transl-1-1 671088664 vtep-207.32778 123340 RNVE
 192.168.4.4      MACVRF-mac-vrf-ep-t2-symm-transl-1-1 671088649 vtep-207.32775 94258 I-ESI-Peer
 192.168.0.5      MACVRF-mac-vrf-ep-t2-symm-transl-1-1 671088644 vtep-207.32774 97821 RNVE
 192.168.0.6      MACVRF-mac-vrf-ep-t2-symm-transl-1-1 671088669 vtep-207.32779 123742 RNVE
```

Verify Type 2 Stitching on DC2-GW21:

```
user@DC2-GW21> show evpn instance MACVRF-mac-vrf-ep-t2-symm-transl-1-1 esi
00:00:ff:ff:00:22:00:04:00:01 dci extensive | except irb
```

```
Instance: MACVRF-mac-vrf-ep-t2-symm-transl-1-1
  Route Distinguisher: 192.168.4.3:33302
  Encapsulation type: VXLAN
  Control word enabled
  Duplicate MAC detection threshold: 5
  Duplicate MAC detection window: 180
  MAC database status                     Local  Remote
    MAC advertisements:                     101    2428
    MAC+IP advertisements:                  505    3939
    Default gateway MAC advertisements:     202       0
  Number of local interfaces: 1 (1 up)
    Interface name  ESI                            Mode             Status     AC-Role
    .local..207     00:00:ff:ff:00:22:00:04:00:01  all-active       Up         Root
    Interface name  VLAN   VNI    Status  L3 context
  Number of protect interfaces: 0
  Number of bridge domains: 101
  Number of neighbors: 6
    Address              MAC    MAC+IP        AD        IM        ES Leaf-label Remote-DCI-Peer
Flow-label
    192.168.0.4          202       505         2       101         0
NO
    192.168.0.5          202       505         2       101         0
NO
    192.168.0.6          202       505         0       101         0
NO
    192.168.4.1          810      2121         2       101         0             DCI
NO
    192.168.4.2          810      2121         2       101         0             DCI
NO
    192.168.4.4          202       505       104       202         0
NO
  Number of ethernet segments: 104
    ESI: 00:00:ff:ff:00:22:00:04:00:01 I-ESI
      Local interface: .local..207, Status: Up/Forwarding
      Number of remote PEs connected: 1
        Remote-PE        MAC-label  Aliasing-label  Mode
        192.168.4.4      0          0               all-active
      DF Election Algorithm: MOD based
      Designated forwarder: 192.168.4.3
      Backup forwarder: 192.168.4.4
      Last designated forwarder update: Oct 05 23:33:13
  Router-ID: 192.168.4.3
  Source VTEP interface IP: 192.168.4.3
  SMET Forwarding: Disabled
  EVPN-Interconnect:
    Route-distinguisher: 192.168.4.3:46000
    Vrf-import: [ __evpn-ic-import-MACVRF-mac-vrf-ep-t2-symm-transl-1-1-internal__ ]
    Vrf-export: [ __evpn-ic-export-MACVRF-mac-vrf-ep-t2-symm-transl-1-1-internal__ ]
    Vrf-import-target: [ target:60005:60001 ]
    Vrf-export-target: [ target:60005:60001 ]
  DCI route stats                      Local
    AD  route advertisements:              1
    IM  route advertisements:            101
    MAC route advertisements:           1618
    MAC+IP route advertisements:        4242
    ES  route advertisements:              0
    SG  Proxy route advertisements:        0
```

Configure Layer 2 stitching with non-translated global VNIs on DC2-GW21:

```
set routing-instances MACVRF-mac-vrf-ep-t2-symm-1-1 vlans EP-TYPE-2-VLAN-501 vlan-id 501
set routing-instances MACVRF-mac-vrf-ep-t2-symm-1-1 vlans EP-TYPE-2-VLAN-501 l3-interface irb.501
set routing-instances MACVRF-mac-vrf-ep-t2-symm-1-1 vlans EP-TYPE-2-VLAN-501 vxlan vni 200501
set routing-instances MACVRF-mac-vrf-ep-t2-symm-1-1 vlans EP-TYPE-2-VLAN-502 vlan-id 502
set routing-instances MACVRF-mac-vrf-ep-t2-symm-1-1 vlans EP-TYPE-2-VLAN-502 l3-interface irb.502
set routing-instances MACVRF-mac-vrf-ep-t2-symm-1-1 vlans EP-TYPE-2-VLAN-502 vxlan vni 200502
set routing-instances MACVRF-mac-vrf-ep-t2-symm-1-1 vlans EP-TYPE-2-VLAN-503 vlan-id 503
set routing-instances MACVRF-mac-vrf-ep-t2-symm-1-1 vlans EP-TYPE-2-VLAN-503 l3-interface irb.503
set routing-instances MACVRF-mac-vrf-ep-t2-symm-1-1 vlans EP-TYPE-2-VLAN-503 vxlan vni 200503
set routing-instances MACVRF-mac-vrf-ep-t2-symm-1-1 vlans EP-TYPE-2-VLAN-504 vlan-id 504
set routing-instances MACVRF-mac-vrf-ep-t2-symm-1-1 vlans EP-TYPE-2-VLAN-504 l3-interface irb.504
set routing-instances MACVRF-mac-vrf-ep-t2-symm-1-1 vlans EP-TYPE-2-VLAN-504 vxlan vni 200504
```

Verify non-translated global VNIs on DC2-GW21:

```
user@DC2-GW21> show evpn database state dci-adv instance MACVRF-mac-vrf-ep-t2-symm-1-1 l2-domain-
id 200501

Instance: MACVRF-mac-vrf-ep-t2-symm-1-1
VLAN  DomainId  MAC address        Active source                  Timestamp        IP address
     200501     00:00:5e:00:00:04  00:00:ff:ee:22:02:00:00:00:01  Oct 06 05:51:59  10.21.245.254

2001:db8::21:0:1f5:254
```

```
     200501     0c:59:9c:6f:7a:a0  192.168.0.4                   Oct 06 04:33:28  10.21.245.248

2001:db8::21:0:1f5:248

fe80::e59:9c01:f56f:7aa0
     200501     4c:6d:58:00:00:00  192.168.0.5                   Oct 06 04:33:21  10.21.245.249

2001:db8::21:0:1f5:249

fe80::4e6d:5801:f500:0
     200501     68:22:8e:e3:ad:df  192.168.4.4                   Oct 06 04:33:23  10.21.245.247

2001:db8::21:0:1f5:247

fe80::6a22:8e01:f5e3:addf
     200501     be:31:2f:01:f5:01  00:00:00:ff:00:11:00:02:00:04  Oct 06 04:33:29  10.21.245.101

2001:db8::21:0:1f5:65
     200501     be:32:2f:01:f5:01  192.168.0.6                   Oct 06 04:33:28  10.21.245.111

2001:db8::21:0:1f5:6f
     200501     c8:fe:6a:2d:56:00  192.168.0.6                   Oct 06 04:33:26  10.21.245.250

2001:db8::21:0:1f5:250

fe80::cafe:6a01:f52d:5600
     200501     f8:c1:16:01:28:07  irb.501                       Oct 06 05:51:59  10.21.245.246

2001:db8::21:0:1f5:246

fe80::fac1:1601:f501:2807
```

Verify the EVPN routes advertised to the data center on DC2-GW21:

```
user@DC2-GW21> show evpn database state dc-adv instance MACVRF-mac-vrf-ep-t2-symm-1-1 l2-domain-
id 200501

Instance: MACVRF-mac-vrf-ep-t2-symm-1-1
VLAN  DomainId  MAC address        Active source                  Timestamp        IP address
     200501     0c:59:9c:8c:46:e4  00:00:ff:ff:00:11:00:02:00:01  Oct 06 04:33:28  10.20.245.245

2001:db8::20:0:1f5:245
```

```
fe80::e59:9c01:f58c:46e4
     200501     4c:73:4f:e6:4f:80  00:00:ff:ff:00:11:00:02:00:01  Oct 06 04:33:27  10.20.245.241

2001:db8::20:0:1f5:241

fe80::4e73:4f01:f5e6:4f80
     200501     74:e7:98:61:44:76  00:00:ff:ff:00:11:00:02:00:01  Oct 06 04:33:28  10.20.245.242

2001:db8::20:0:1f5:242

fe80::76e7:9801:f561:4476
     200501     a4:51:5e:3e:1a:0e  00:00:ff:ff:00:11:00:02:00:01  Oct 06 04:33:28  10.20.245.243

2001:db8::20:0:1f5:243

fe80::a651:5e01:f53e:1a0e
     200501     ba:31:2f:01:f5:01  00:00:ff:ff:00:11:00:02:00:01  Oct 06 04:33:28  10.20.245.1

2001:db8::20:0:1f5:1
     200501     ba:32:2f:01:f5:01  00:00:ff:ff:00:11:00:02:00:01  Oct 06 04:33:28  10.20.245.11

2001:db8::20:0:1f5:b
     200501     e8:a2:45:e2:a6:b0  00:00:ff:ff:00:11:00:02:00:01  Oct 06 04:33:28  10.20.245.244

2001:db8::20:0:1f5:244

fe80::eaa2:4501:f5e2:a6b0
```

## Type 5 Stitching

Type 5 stitching is Juniper Networks recommended method for advertising the router MAC address. The router MAC address and Layer 3 VNI are how data is forwarded in a stitched environment. Type 2 symmetric routing only carries the Layer 2 and Layer 3 VNI information to the remote gateways and does not perform data forwarding. Configure Type 5 stitching on DC2-GW21:

```
set routing-instances VRF-mac-vrf-ep-t2-symm-1-1 protocols evpn interconnect vrf-target
target:5003:6001
set routing-instances VRF-mac-vrf-ep-t2-symm-1-1 protocols evpn interconnect route-distinguisher
192.168.4.3:44000
```

```
set routing-instances VRF-mac-vrf-ep-t2-symm-1-1 protocols evpn ip-prefix-routes advertise
direct-nexthop
set routing-instances VRF-mac-vrf-ep-t2-symm-1-1 protocols evpn ip-prefix-routes encapsulation
vxlan
set routing-instances VRF-mac-vrf-ep-t2-symm-1-1 protocols evpn ip-prefix-routes vni 9100501
```

Verify the Type 5 routes on DC2-GW21 being exported or stitched to the remote gateways and local leaf nodes:

```
user@DC2-GW21> show evpn ip-prefix-database l3-context VRF-mac-vrf-ep-t2-symm-1-1 direction
exported family inet

L3 context: VRF-mac-vrf-ep-t2-symm-1-1

IPv4->EVPN Exported Prefixes
Prefix                                       EVPN route status
10.20.245.0/24                                DC Created
10.20.245.1/32                                DC Created
10.20.245.11/32                               DC Created
10.20.245.241/32                              DC Created
```

Verify the Type 5 routes on DC2-GW21 being imported or stitched from the remote gateways and the local leaf nodes:

```
user@DC2-GW21> show evpn ip-prefix-database l3-context VRF-mac-vrf-ep-t2-symm-1-1 direction
imported family inet

L3 context: VRF-mac-vrf-ep-t2-symm-1-1

EVPN->IPv4 Imported Prefixes
Prefix                                       Etag
10.20.245.0/24                                0
  Route distinguisher    VNI/Label  Router MAC         Nexthop/Overlay GW/ESI   Route-Status
Reject-Reason
  192.168.4.1:44000      9100501    4c:73:4f:e6:4f:80  192.168.4.1               Accepted      n/
a
  192.168.4.2:44000      9100501    74:e7:98:61:44:76  192.168.4.2               Accepted      n/
a
10.20.245.1/32                                0
  Route distinguisher    VNI/Label  Router MAC         Nexthop/Overlay GW/ESI   Route-Status
Reject-Reason
```

```
  192.168.4.1:44000      9100501    4c:73:4f:e6:4f:80    192.168.4.1               Accepted
n/a
  192.168.4.2:44000      9100501    74:e7:98:61:44:76  192.168.4.2               Accepted      n/
a
10.20.245.11/32                             0
  Route distinguisher    VNI/Label  Router MAC         Nexthop/Overlay GW/ESI   Route-Status
Reject-Reason
  192.168.4.1:44000      9100501    4c:73:4f:e6:4f:80  192.168.4.1               Accepted      n/
a
  192.168.4.2:44000      9100501    74:e7:98:61:44:76  192.168.4.2               Accepted      n/
a
10.20.245.241/32                            0
  Route distinguisher    VNI/Label  Router MAC         Nexthop/Overlay GW/ESI   Route-Status
Reject-Reason
  192.168.4.1:44000      9100501    4c:73:4f:e6:4f:80  192.168.4.1               Accepted      n/
a
```

## Interconnect Multihoming for DCI Gateway Peers

Configure multihoming on DC2-GW21 for redundancy. A full discussion of multihoming is beyond the scope of this document. For more about multihoming, see *EVPN Multihoming Overview*.

```
set protocols evpn interconnect-multihoming-peer-gateways <VTEP-IP of each DCI-GW peer in local
DC>

Here is the same example with a device from our topology.

user@DC2-GW21# set protocols evpn interconnect-multihoming-peer-gateways 192.168.4.4
```

**NOTE**: You must configure `interconnect-multihoming-peer-gateways` globally.

```
set protocols evpn interconnect-multihoming-peer-gateways
```

It cannot be configured within a MAC-VRF, as shown.

```
set routing-instances <instance-name> protocols evpn interconnect-multihoming-peer-
gateways
```

Verify multihoming on DC2-GW21 is working properly. You can see the multihomed peer has the designation of I-ESI-Peer:

```
user@DC2-GW21> show ethernet-switching vxlan-tunnel-end-point remote summary instance MACVRF-mac-
vrf-ep-t2-symm-1-1

Logical System Name       Id  SVTEP-IP         IFL   L3-Idx    SVTEP-Mode    ELP-SVTEP-IP
<default>                 0   192.168.4.3      lo0.0    0
 RVTEP-IP         L2-RTT                   IFL-Idx   Interface    NH-Id   RVTEP-Mode  ELP-
IP        Flags
 192.168.4.1      MACVRF-mac-vrf-ep-t2-symm-1-1 671088652 vtep-868.32776 121062 Wan-VTEP
 192.168.4.2      MACVRF-mac-vrf-ep-t2-symm-1-1 671088657 vtep-868.32777 85558 Wan-VTEP
 192.168.0.4      MACVRF-mac-vrf-ep-t2-symm-1-1 671088662 vtep-868.32778 117187 RNVE
 192.168.4.4      MACVRF-mac-vrf-ep-t2-symm-1-1 671088647 vtep-868.32775 113257 I-ESI-Peer
 192.168.0.5      MACVRF-mac-vrf-ep-t2-symm-1-1 671088642 vtep-868.32774 89355 RNVE
 192.168.0.6      MACVRF-mac-vrf-ep-t2-symm-1-1 671088667 vtep-868.32779 116701 RNVE
```

## Configurations for DC2-GW21 and DC2-Leaf21

Configurations are provided for reference only. Do not copy and paste these configurations as-is into your devices. Configurations for other gateway and leaf nodes will be similar. These configurations have been truncated for readability.

DC2-GW21

```
set chassis aggregated-devices ethernet device-count 64
set interfaces ae1 aggregated-ether-options lacp active
set interfaces ae1 aggregated-ether-options lacp periodic fast
set interfaces ae1 aggregated-ether-options minimum-links 1
set interfaces ae1 mtu 9192
set interfaces ae1 unit 0 family inet address 172.16.11.1/31
set interfaces ae1 unit 0 family inet mtu 9000
set interfaces ae1 unit 0 family iso
set interfaces ae2 aggregated-ether-options lacp active
set interfaces ae2 aggregated-ether-options lacp periodic fast
set interfaces ae2 aggregated-ether-options minimum-links 1
set interfaces ae2 mtu 9192
set interfaces ae2 unit 0 family ethernet-switching interface-mode trunk
set interfaces ae2 unit 0 family ethernet-switching vlan members UNDERLAY-VLAN-4004
set interfaces ae3 aggregated-ether-options lacp active
set interfaces ae3 aggregated-ether-options lacp periodic fast
set interfaces ae3 aggregated-ether-options minimum-links 1
set interfaces ae3 mtu 9192
set interfaces ae3 unit 0 family inet address 172.16.15.1/31
set interfaces ae3 unit 0 family inet mtu 9000
set interfaces ae3 unit 0 family iso
set interfaces ae3 unit 0 vlan-id 4093
set interfaces ae3 vlan-tagging
set interfaces ae4 aggregated-ether-options lacp active
set interfaces ae4 aggregated-ether-options lacp periodic fast
set interfaces ae4 aggregated-ether-options minimum-links 1
set interfaces ae4 mtu 9192
set interfaces ae4 unit 0 family inet address 172.16.17.1/31
set interfaces ae4 unit 0 family inet mtu 9000
set interfaces ae4 unit 0 family iso
set interfaces et-0/0/0 ether-options 802.3ad ae1
set interfaces et-0/0/1 ether-options 802.3ad ae2
set interfaces et-0/0/2 ether-options 802.3ad ae4
set interfaces et-0/0/3 ether-options 802.3ad ae3
set interfaces et-1/0/0 ether-options 802.3ad ae1
set interfaces et-1/0/1 ether-options 802.3ad ae2
set interfaces et-1/0/2 ether-options 802.3ad ae4
set interfaces et-1/0/3 ether-options 802.3ad ae3
set interfaces irb mtu 1514
set interfaces irb unit 10 family inet address 10.12.10.246/24 preferred
set interfaces irb unit 10 family inet address 10.12.10.246/24 virtual-gateway-address 
10.12.10.254
set interfaces irb unit 10 family inet6 address 2001:db8::12:0:a:246/112 preferred
set interfaces irb unit 10 family inet6 address 2001:db8::12:0:a:246/112 virtual-gateway-address 
2001:db8::12:0:a:254
set interfaces irb unit 10 virtual-gateway-accept-data
set interfaces irb unit 10 virtual-gateway-esi all-active 00:00:ff:ee:22:04:00:00:00:09
set interfaces irb unit 10 virtual-gateway-v4-mac 00:00:5e:00:00:04
set interfaces irb unit 10 virtual-gateway-v6-mac 00:00:5e:00:00:04
set interfaces irb unit 100 family inet address 10.12.100.246/24 preferred
set interfaces irb unit 100 family inet address 10.12.100.246/24 virtual-gateway-address 
10.12.100.254
set interfaces irb unit 100 family inet6 address 2001:db8::12:0:64:246/112 preferred
set interfaces irb unit 100 family inet6 address 2001:db8::12:0:64:246/112 virtual-gateway-
address 2001:db8::12:0:64:254
set interfaces irb unit 100 virtual-gateway-accept-data
set interfaces irb unit 100 virtual-gateway-esi all-active 00:00:ff:ee:22:04:00:00:00:63
set interfaces irb unit 100 virtual-gateway-v4-mac 00:00:5e:00:00:04
set interfaces irb unit 100 virtual-gateway-v6-mac 00:00:5e:00:00:04
set interfaces irb unit 1001 family inet address 10.33.233.246/24 preferred
set interfaces irb unit 1001 family inet address 10.33.233.246/24 virtual-gateway-address
10.33.233.254
set interfaces irb unit 1001 family inet6 address 2001:db8::30:0:3e9:246/112 preferred
set interfaces irb unit 1001 family inet6 address 2001:db8::30:0:3e9:246/112 virtual-gateway-
address 2001:db8::30:0:3e9:254
set interfaces irb unit 1001 virtual-gateway-accept-data
set interfaces irb unit 1001 virtual-gateway-esi all-active 00:00:ff:ee:22:01:00:00:00:01
set interfaces irb unit 1001 virtual-gateway-v4-mac 00:00:5e:00:00:04
set interfaces irb unit 1001 virtual-gateway-v6-mac 00:00:5e:00:00:04
set interfaces irb unit <*> family inet6 nd6-stale-time 3600
set interfaces lo0 unit 0 family inet address 192.168.4.3/32 primary
set interfaces lo0 unit 10 family inet
set interfaces lo0 unit 1001 family inet
set policy-options community VRF-3-1-COMM members target:5003:6001
set policy-options community VRF-3-10-COMM members target:5003:6010
set policy-options community VRF-3-11-COMM members target:5003:6011
set policy-options community wan_underlay_comm members 12345:12345
set policy-options policy-statement per-packet-load-balance term 1 then load-balance per-packet
set policy-options policy-statement underlay-clos-export term from_wan from community
wan_underlay_comm
set policy-options policy-statement underlay-clos-export term from_wan then reject
set policy-options policy-statement underlay-clos-export term loopback from interface lo0.0
set policy-options policy-statement underlay-clos-export term loopback then accept
set policy-options policy-statement underlay-clos-export-wan term def then reject
set policy-options policy-statement underlay-clos-export-wan term loopback from interface lo0.0
set policy-options policy-statement underlay-clos-export-wan term loopback then accept
set policy-options policy-statement underlay-clos-export-wan term loopback then community add
wan_underlay_comm
set policy-options policy-statement wan_import from community wan_underlay_comm
set policy-options policy-statement wan_import then local-preference subtract 10
set protocols bgp graceful-restart
set protocols bgp group overlay-ebgp authentication-key overlay_ebgp_admin
set protocols bgp group overlay-ebgp bfd-liveness-detection minimum-interval 1000
set protocols bgp group overlay-ebgp bfd-liveness-detection multiplier 3
set protocols bgp group overlay-ebgp bfd-liveness-detection session-mode automatic
set protocols bgp group overlay-ebgp family evpn signaling
set protocols bgp group overlay-ebgp family evpn signaling delay-route-advertisements minimum-
delay routing-uptime 400
set protocols bgp group overlay-ebgp local-address 192.168.4.3
set protocols bgp group overlay-ebgp local-as 4200000033
set protocols bgp group overlay-ebgp multihop no-nexthop-change
set protocols bgp group overlay-ebgp multipath multiple-as
set protocols bgp group overlay-ebgp neighbor 192.168.6.1 peer-as 4200000042
set protocols bgp group overlay-ebgp neighbor 192.168.6.2 peer-as 4200000042
set protocols bgp group overlay-ebgp type external
set protocols bgp group overlay-ebgp vpn-apply-export
set protocols bgp group overlay-ebgp-extn-dci authentication-key overlay_ebgp_dci_admin
set protocols bgp group overlay-ebgp-extn-dci bfd-liveness-detection minimum-interval 1000
set protocols bgp group overlay-ebgp-extn-dci bfd-liveness-detection multiplier 3
set protocols bgp group overlay-ebgp-extn-dci bfd-liveness-detection session-mode automatic
set protocols bgp group overlay-ebgp-extn-dci family evpn signaling
set protocols bgp group overlay-ebgp-extn-dci family evpn signaling delay-route-advertisements
minimum-delay routing-uptime 400
set protocols bgp group overlay-ebgp-extn-dci family route-target external-paths 2
set protocols bgp group overlay-ebgp-extn-dci local-address 192.168.4.3
set protocols bgp group overlay-ebgp-extn-dci local-as 4210000002
set protocols bgp group overlay-ebgp-extn-dci multihop no-nexthop-change
set protocols bgp group overlay-ebgp-extn-dci multipath multiple-as
set protocols bgp group overlay-ebgp-extn-dci neighbor 192.168.4.1 peer-as 4210000001
set protocols bgp group overlay-ebgp-extn-dci neighbor 192.168.4.2 peer-as 4210000001
set protocols bgp group overlay-ebgp-extn-dci type external
set protocols bgp group underlay-bgp authentication-key underlay_ebgp_admin
set protocols bgp group underlay-bgp bfd-liveness-detection minimum-interval 1000
set protocols bgp group underlay-bgp bfd-liveness-detection multiplier 3
set protocols bgp group underlay-bgp bfd-liveness-detection session-mode automatic
set protocols bgp group underlay-bgp export underlay-clos-export
set protocols bgp group underlay-bgp family inet
set protocols bgp group underlay-bgp local-as 4200000033
set protocols bgp group underlay-bgp multipath multiple-as
set protocols bgp group underlay-bgp neighbor 172.16.15.0 peer-as 4200000042
set protocols bgp group underlay-bgp neighbor 172.16.17.0 peer-as 4200000042
set protocols bgp group underlay-bgp type external
set protocols bgp group underlay-bgp-wan authentication-key underlay_ebgp_admin
set protocols bgp group underlay-bgp-wan bfd-liveness-detection minimum-interval 1000
set protocols bgp group underlay-bgp-wan bfd-liveness-detection multiplier 3
set protocols bgp group underlay-bgp-wan bfd-liveness-detection session-mode automatic
set protocols bgp group underlay-bgp-wan export underlay-clos-export-wan
set protocols bgp group underlay-bgp-wan import wan_import
set protocols bgp group underlay-bgp-wan local-as 4200000033
set protocols bgp group underlay-bgp-wan multipath multiple-as
set protocols bgp group underlay-bgp-wan neighbor 172.16.11.0 peer-as 4200000062
set protocols bgp group underlay-bgp-wan neighbor 172.16.13.0 peer-as 4200000061
set protocols bgp group underlay-bgp-wan type external
set protocols bgp log-updown
set protocols evpn interconnect-multihoming-peer-gateways [ 192.168.4.4 ]
set protocols evpn remote-ip-host-routes
set protocols l2-learning global-mac-ip-table-aging-time 3600
set protocols l2-learning global-mac-table-aging-time 4200
set protocols lldp interface all
set routing-instances MACVRF-mac-vrf-ep-t2-asymm-2-1 instance-type mac-vrf
set routing-instances MACVRF-mac-vrf-ep-t2-asymm-2-1 protocols evpn default-gateway no-gateway-
community
set routing-instances MACVRF-mac-vrf-ep-t2-asymm-2-1 protocols evpn encapsulation vxlan
set routing-instances MACVRF-mac-vrf-ep-t2-asymm-2-1 protocols evpn extended-vni-list all
set routing-instances MACVRF-mac-vrf-ep-t2-asymm-2-1 protocols evpn interconnect esi
00:00:ff:ff:00:22:00:0:00:1 all-active
set routing-instances MACVRF-mac-vrf-ep-t2-asymm-2-1 protocols evpn interconnect interconnected-
vni-list 401501
set routing-instances MACVRF-mac-vrf-ep-t2-asymm-2-1 protocols evpn interconnect interconnected-
vni-list 401502
set routing-instances MACVRF-mac-vrf-ep-t2-asymm-2-1 protocols evpn interconnect interconnected-
vni-list 401503
set routing-instances MACVRF-mac-vrf-ep-t2-asymm-2-1 protocols evpn interconnect route-
distinguisher 192.168.4.3:30000
set routing-instances MACVRF-mac-vrf-ep-t2-asymm-2-1 protocols evpn interconnect vrf-target
target:60001:60001
set routing-instances MACVRF-mac-vrf-ep-t2-asymm-2-1 protocols evpn remote-ip-host-routes
set routing-instances MACVRF-mac-vrf-ep-t2-asymm-2-1 protocols evpn vni-options vni 401501 vrf-
target target:2:401501
set routing-instances MACVRF-mac-vrf-ep-t2-asymm-2-1 protocols evpn vni-options vni 401502 vrf-
target target:2:401502
set routing-instances MACVRF-mac-vrf-ep-t2-asymm-2-1 protocols evpn vni-options vni 401503 vrf-
target target:2:401503
set routing-instances MACVRF-mac-vrf-ep-t2-asymm-2-1 route-distinguisher 192.168.4.3:34801
set routing-instances MACVRF-mac-vrf-ep-t2-asymm-2-1 service-type vlan-aware
set routing-instances MACVRF-mac-vrf-ep-t2-asymm-2-1 vlans EP-TYPE-2-VLAN-1501 l3-interface
irb.1501
set routing-instances MACVRF-mac-vrf-ep-t2-asymm-2-1 vlans EP-TYPE-2-VLAN-1501 vlan-id 1501
set routing-instances MACVRF-mac-vrf-ep-t2-asymm-2-1 vlans EP-TYPE-2-VLAN-1501 vxlan vni 401501
set routing-instances MACVRF-mac-vrf-ep-t2-asymm-2-1 vlans EP-TYPE-2-VLAN-1502 l3-interface
irb.1502
set routing-instances MACVRF-mac-vrf-ep-t2-asymm-2-1 vlans EP-TYPE-2-VLAN-1502 vlan-id 1502
set routing-instances MACVRF-mac-vrf-ep-t2-asymm-2-1 vlans EP-TYPE-2-VLAN-1502 vxlan vni 401502
set routing-instances MACVRF-mac-vrf-ep-t2-asymm-2-1 vlans EP-TYPE-2-VLAN-1503 l3-interface
irb.1503
set routing-instances MACVRF-mac-vrf-ep-t2-asymm-2-1 vlans EP-TYPE-2-VLAN-1503 vlan-id 1503
set routing-instances MACVRF-mac-vrf-ep-t2-asymm-2-1 vlans EP-TYPE-2-VLAN-1503 vxlan vni 401503
set routing-instances MACVRF-mac-vrf-ep-t2-asymm-2-1 vrf-target target:33302:1501
set routing-instances MACVRF-mac-vrf-ep-t2-asymm-2-1 vtep-source-interface lo0.0
set routing-instances MACVRF-mac-vrf-ep-t2-asymm-transl-2-1 instance-type mac-vrf
set routing-instances MACVRF-mac-vrf-ep-t2-asymm-transl-2-1 protocols evpn default-gateway no-
gateway-community
set routing-instances MACVRF-mac-vrf-ep-t2-asymm-transl-2-1 protocols evpn encapsulation vxlan
set routing-instances MACVRF-mac-vrf-ep-t2-asymm-transl-2-1 protocols evpn extended-vni-list all
set routing-instances MACVRF-mac-vrf-ep-t2-asymm-transl-2-1 protocols evpn interconnect esi
00:00:ff:ff:00:22:00:1:00:1 all-active
set routing-instances MACVRF-mac-vrf-ep-t2-asymm-transl-2-1 protocols evpn interconnect
interconnected-vni-list 930001
set routing-instances MACVRF-mac-vrf-ep-t2-asymm-transl-2-1 protocols evpn interconnect
interconnected-vni-list 930002
set routing-instances MACVRF-mac-vrf-ep-t2-asymm-transl-2-1 protocols evpn interconnect
interconnected-vni-list 930003
set routing-instances MACVRF-mac-vrf-ep-t2-asymm-transl-2-1 protocols evpn interconnect route-
distinguisher 192.168.4.3:34000
set routing-instances MACVRF-mac-vrf-ep-t2-asymm-transl-2-1 protocols evpn interconnect vrf-
target target:60002:60001
set routing-instances MACVRF-mac-vrf-ep-t2-asymm-transl-2-1 protocols evpn remote-ip-host-routes
set routing-instances MACVRF-mac-vrf-ep-t2-asymm-transl-2-1 protocols evpn vni-options vni
311001 vrf-target target:2:311001
set routing-instances MACVRF-mac-vrf-ep-t2-asymm-transl-2-1 protocols evpn vni-options vni
311002 vrf-target target:2:311002
set routing-instances MACVRF-mac-vrf-ep-t2-asymm-transl-2-1 protocols evpn vni-options vni
311003 vrf-target target:2:311003
set routing-instances MACVRF-mac-vrf-ep-t2-asymm-transl-2-1 route-distinguisher 192.168.4.3:34301
set routing-instances MACVRF-mac-vrf-ep-t2-asymm-transl-2-1 service-type vlan-aware
set routing-instances MACVRF-mac-vrf-ep-t2-asymm-transl-2-1 vlans EP-TYPE-2-VLAN-1001 l3-
interface irb.1001
set routing-instances MACVRF-mac-vrf-ep-t2-asymm-transl-2-1 vlans EP-TYPE-2-VLAN-1001 vlan-id
1001
set routing-instances MACVRF-mac-vrf-ep-t2-asymm-transl-2-1 vlans EP-TYPE-2-VLAN-1001 vxlan
translation-vni 930001
set routing-instances MACVRF-mac-vrf-ep-t2-asymm-transl-2-1 vlans EP-TYPE-2-VLAN-1001 vxlan vni
311001
set routing-instances MACVRF-mac-vrf-ep-t2-asymm-transl-2-1 vlans EP-TYPE-2-VLAN-1002 l3-
interface irb.1002
set routing-instances MACVRF-mac-vrf-ep-t2-asymm-transl-2-1 vlans EP-TYPE-2-VLAN-1002 vlan-id
1002
set routing-instances MACVRF-mac-vrf-ep-t2-asymm-transl-2-1 vlans EP-TYPE-2-VLAN-1002 vxlan
translation-vni 930002
set routing-instances MACVRF-mac-vrf-ep-t2-asymm-transl-2-1 vlans EP-TYPE-2-VLAN-1002 vxlan vni
```

```
311002
set routing-instances MACVRF-mac-vrf-ep-t2-asymm-transl-2-1 vlans EP-TYPE-2-VLAN-1003 l3-
interface irb.1003
set routing-instances MACVRF-mac-vrf-ep-t2-asymm-transl-2-1 vlans EP-TYPE-2-VLAN-1003 vlan-id
1003
set routing-instances MACVRF-mac-vrf-ep-t2-asymm-transl-2-1 vlans EP-TYPE-2-VLAN-1003 vxlan
translation-vni 930003
set routing-instances MACVRF-mac-vrf-ep-t2-asymm-transl-2-1 vlans EP-TYPE-2-VLAN-1003 vxlan vni
311003
set routing-instances MACVRF-mac-vrf-ep-t2-asymm-transl-2-1 vrf-target target:33302:1001
set routing-instances MACVRF-mac-vrf-ep-t2-asymm-transl-2-1 vtep-source-interface lo0.0
set routing-instances MACVRF-mac-vrf-ep-t2-symm-1-1 instance-type mac-vrf
set routing-instances MACVRF-mac-vrf-ep-t2-symm-1-1 protocols evpn default-gateway no-gateway-
community
set routing-instances MACVRF-mac-vrf-ep-t2-symm-1-1 protocols evpn encapsulation vxlan
set routing-instances MACVRF-mac-vrf-ep-t2-symm-1-1 protocols evpn extended-vni-list all
set routing-instances MACVRF-mac-vrf-ep-t2-symm-1-1 protocols evpn interconnect esi
00:00:ff:ff:00:22:00:2:00:1 all-active
set routing-instances MACVRF-mac-vrf-ep-t2-symm-1-1 protocols evpn interconnect interconnected-
vni-list 200501
set routing-instances MACVRF-mac-vrf-ep-t2-symm-1-1 protocols evpn interconnect interconnected-
vni-list 200502
set routing-instances MACVRF-mac-vrf-ep-t2-symm-1-1 protocols evpn interconnect interconnected-
vni-list 200503
set routing-instances MACVRF-mac-vrf-ep-t2-symm-1-1 protocols evpn interconnect route-
distinguisher 192.168.4.3:38000
set routing-instances MACVRF-mac-vrf-ep-t2-symm-1-1 protocols evpn interconnect vrf-target
target:60003:60001
set routing-instances MACVRF-mac-vrf-ep-t2-symm-1-1 route-distinguisher 192.168.4.3:33801
set routing-instances MACVRF-mac-vrf-ep-t2-symm-1-1 service-type vlan-aware
set routing-instances MACVRF-mac-vrf-ep-t2-symm-1-1 vlans EP-TYPE-2-VLAN-501 l3-interface irb.501
set routing-instances MACVRF-mac-vrf-ep-t2-symm-1-1 vlans EP-TYPE-2-VLAN-501 vlan-id 501
set routing-instances MACVRF-mac-vrf-ep-t2-symm-1-1 vlans EP-TYPE-2-VLAN-501 vxlan vni 200501
set routing-instances MACVRF-mac-vrf-ep-t2-symm-1-1 vlans EP-TYPE-2-VLAN-502 l3-interface irb.502
set routing-instances MACVRF-mac-vrf-ep-t2-symm-1-1 vlans EP-TYPE-2-VLAN-502 vlan-id 502
set routing-instances MACVRF-mac-vrf-ep-t2-symm-1-1 vlans EP-TYPE-2-VLAN-502 vxlan vni 200502
set routing-instances MACVRF-mac-vrf-ep-t2-symm-1-1 vlans EP-TYPE-2-VLAN-503 l3-interface irb.503
set routing-instances MACVRF-mac-vrf-ep-t2-symm-1-1 vlans EP-TYPE-2-VLAN-503 vlan-id 503
set routing-instances MACVRF-mac-vrf-ep-t2-symm-1-1 vlans EP-TYPE-2-VLAN-503 vxlan vni 200503
set routing-instances MACVRF-mac-vrf-ep-t2-symm-1-1 vrf-target target:33302:501
set routing-instances MACVRF-mac-vrf-ep-t2-symm-1-1 vtep-source-interface lo0.0
set routing-instances MACVRF-mac-vrf-ep-t2-symm-transl-1-1 instance-type mac-vrf
set routing-instances MACVRF-mac-vrf-ep-t2-symm-transl-1-1 protocols evpn default-gateway no-
```

```
gateway-community
set routing-instances MACVRF-mac-vrf-ep-t2-symm-transl-1-1 protocols evpn encapsulation vxlan
set routing-instances MACVRF-mac-vrf-ep-t2-symm-transl-1-1 protocols evpn extended-vni-list all
set routing-instances MACVRF-mac-vrf-ep-t2-symm-transl-1-1 protocols evpn interconnect esi
00:00:ff:ff:00:22:00:4:00:1 all-active
set routing-instances MACVRF-mac-vrf-ep-t2-symm-transl-1-1 protocols evpn interconnect
interconnected-vni-list 920001
set routing-instances MACVRF-mac-vrf-ep-t2-symm-transl-1-1 protocols evpn interconnect
interconnected-vni-list 920002
set routing-instances MACVRF-mac-vrf-ep-t2-symm-transl-1-1 protocols evpn interconnect
interconnected-vni-list 920003
set routing-instances MACVRF-mac-vrf-ep-t2-symm-transl-1-1 protocols evpn interconnect route-
distinguisher 192.168.4.3:46000
set routing-instances MACVRF-mac-vrf-ep-t2-symm-transl-1-1 protocols evpn interconnect vrf-
target target:60005:60001
set routing-instances MACVRF-mac-vrf-ep-t2-symm-transl-1-1 route-distinguisher 192.168.4.3:33302
set routing-instances MACVRF-mac-vrf-ep-t2-symm-transl-1-1 service-type vlan-aware
set routing-instances MACVRF-mac-vrf-ep-t2-symm-transl-1-1 vlans EP-TYPE-2-VLAN-10 l3-interface
irb.10
set routing-instances MACVRF-mac-vrf-ep-t2-symm-transl-1-1 vlans EP-TYPE-2-VLAN-10 vlan-id 10
set routing-instances MACVRF-mac-vrf-ep-t2-symm-transl-1-1 vlans EP-TYPE-2-VLAN-10 vxlan
translation-vni 920009
set routing-instances MACVRF-mac-vrf-ep-t2-symm-transl-1-1 vlans EP-TYPE-2-VLAN-10 vxlan vni
120010
set routing-instances MACVRF-mac-vrf-ep-t2-symm-transl-1-1 vlans EP-TYPE-2-VLAN-100 l3-interface
irb.100
set routing-instances MACVRF-mac-vrf-ep-t2-symm-transl-1-1 vlans EP-TYPE-2-VLAN-100 vlan-id 100
set routing-instances MACVRF-mac-vrf-ep-t2-symm-transl-1-1 vlans EP-TYPE-2-VLAN-100 vxlan
translation-vni 920099
set routing-instances MACVRF-mac-vrf-ep-t2-symm-transl-1-1 vlans EP-TYPE-2-VLAN-100 vxlan vni
120100
set routing-instances MACVRF-mac-vrf-ep-t2-symm-transl-1-1 vlans EP-TYPE-2-VLAN-101 l3-interface
irb.101
set routing-instances MACVRF-mac-vrf-ep-t2-symm-transl-1-1 vlans EP-TYPE-2-VLAN-101 vlan-id 101
set routing-instances MACVRF-mac-vrf-ep-t2-symm-transl-1-1 vlans EP-TYPE-2-VLAN-101 vxlan
translation-vni 920100
set routing-instances MACVRF-mac-vrf-ep-t2-symm-transl-1-1 vlans EP-TYPE-2-VLAN-101 vxlan vni
120101
set routing-instances MACVRF-mac-vrf-ep-t2-symm-transl-1-1 vrf-target target:33302:2
set routing-instances MACVRF-mac-vrf-ep-t2-symm-transl-1-1 vtep-source-interface lo0.0
set routing-instances MACVRF-mac-vrf-ep-t2-symm-transl-Tenant-1 instance-type mac-vrf
set routing-instances MACVRF-mac-vrf-ep-t2-symm-transl-Tenant-1 protocols evpn default-gateway
no-gateway-community
```

```
set routing-instances MACVRF-mac-vrf-ep-t2-symm-transl-Tenant-1 protocols evpn encapsulation
vxlan
set routing-instances MACVRF-mac-vrf-ep-t2-symm-transl-Tenant-1 protocols evpn extended-vni-list
all
set routing-instances MACVRF-mac-vrf-ep-t2-symm-transl-Tenant-1 protocols evpn interconnect esi
00:00:ff:ff:00:22:00:5:00:1 all-active
set routing-instances MACVRF-mac-vrf-ep-t2-symm-transl-Tenant-1 protocols evpn interconnect
interconnected-vni-list 940001
set routing-instances MACVRF-mac-vrf-ep-t2-symm-transl-Tenant-1 protocols evpn interconnect
interconnected-vni-list 940002
set routing-instances MACVRF-mac-vrf-ep-t2-symm-transl-Tenant-1 protocols evpn interconnect
interconnected-vni-list 940003
set routing-instances MACVRF-mac-vrf-ep-t2-symm-transl-Tenant-1 protocols evpn interconnect
interconnected-vni-list 940004
set routing-instances MACVRF-mac-vrf-ep-t2-symm-transl-Tenant-1 protocols evpn interconnect
route-distinguisher 192.168.4.3:50000
set routing-instances MACVRF-mac-vrf-ep-t2-symm-transl-Tenant-1 protocols evpn interconnect vrf-
target target:60006:60001
set routing-instances MACVRF-mac-vrf-ep-t2-symm-transl-Tenant-1 route-distinguisher
192.168.4.3:37310
set routing-instances MACVRF-mac-vrf-ep-t2-symm-transl-Tenant-1 service-type vlan-aware
set routing-instances MACVRF-mac-vrf-ep-t2-symm-transl-Tenant-1 vlans EP-TYPE-2-VLAN-4010 l3-
interface irb.4010
set routing-instances MACVRF-mac-vrf-ep-t2-symm-transl-Tenant-1 vlans EP-TYPE-2-VLAN-4010 vlan-
id 4010
set routing-instances MACVRF-mac-vrf-ep-t2-symm-transl-Tenant-1 vlans EP-TYPE-2-VLAN-4010 vxlan
translation-vni 940001
set routing-instances MACVRF-mac-vrf-ep-t2-symm-transl-Tenant-1 vlans EP-TYPE-2-VLAN-4010 vxlan
vni 244010
set routing-instances MACVRF-mac-vrf-ep-t2-symm-transl-Tenant-1 vlans EP-TYPE-2-VLAN-4011 l3-
interface irb.4011
set routing-instances MACVRF-mac-vrf-ep-t2-symm-transl-Tenant-1 vlans EP-TYPE-2-VLAN-4011 vlan-
id 4011
set routing-instances MACVRF-mac-vrf-ep-t2-symm-transl-Tenant-1 vlans EP-TYPE-2-VLAN-4011 vxlan
translation-vni 940002
set routing-instances MACVRF-mac-vrf-ep-t2-symm-transl-Tenant-1 vlans EP-TYPE-2-VLAN-4011 vxlan
vni 244011
set routing-instances MACVRF-mac-vrf-ep-t2-symm-transl-Tenant-1 vlans EP-TYPE-2-VLAN-4012 l3-
interface irb.4012
set routing-instances MACVRF-mac-vrf-ep-t2-symm-transl-Tenant-1 vlans EP-TYPE-2-VLAN-4012 vlan-
id 4012
set routing-instances MACVRF-mac-vrf-ep-t2-symm-transl-Tenant-1 vlans EP-TYPE-2-VLAN-4012 vxlan
translation-vni 940003
```

```
set routing-instances MACVRF-mac-vrf-ep-t2-symm-transl-Tenant-1 vlans EP-TYPE-2-VLAN-4012 vxlan
vni 244012
set routing-instances MACVRF-mac-vrf-ep-t2-symm-transl-Tenant-1 vlans EP-TYPE-2-VLAN-4013 l3-
interface irb.4013
set routing-instances MACVRF-mac-vrf-ep-t2-symm-transl-Tenant-1 vlans EP-TYPE-2-VLAN-4013 vlan-
id 4013
set routing-instances MACVRF-mac-vrf-ep-t2-symm-transl-Tenant-1 vlans EP-TYPE-2-VLAN-4013 vxlan
translation-vni 940004
set routing-instances MACVRF-mac-vrf-ep-t2-symm-transl-Tenant-1 vlans EP-TYPE-2-VLAN-4013 vxlan
vni 244013
set routing-instances MACVRF-mac-vrf-ep-t2-symm-transl-Tenant-1 vrf-target target:33302:4010
set routing-instances MACVRF-mac-vrf-ep-t2-symm-transl-Tenant-1 vtep-source-interface lo0.0
set routing-instances VRF-mac-vrf-ep-t2-asymm-2-1 instance-type vrf
set routing-instances VRF-mac-vrf-ep-t2-asymm-2-1 interface irb.1501
set routing-instances VRF-mac-vrf-ep-t2-asymm-2-1 interface irb.1502
set routing-instances VRF-mac-vrf-ep-t2-asymm-2-1 interface irb.1503
set routing-instances VRF-mac-vrf-ep-t2-asymm-2-1 interface irb.1504
set routing-instances VRF-mac-vrf-ep-t2-asymm-2-1 interface lo0.1501
set routing-instances VRF-mac-vrf-ep-t2-asymm-2-1 route-distinguisher 192.168.4.3:1501
set routing-instances VRF-mac-vrf-ep-t2-asymm-2-1 vrf-table-label
set routing-instances VRF-mac-vrf-ep-t2-asymm-2-1 vrf-target target:100:1501
set routing-instances VRF-mac-vrf-ep-t2-asymm-2-10 instance-type vrf
set routing-instances VRF-mac-vrf-ep-t2-asymm-2-10 interface irb.1537
set routing-instances VRF-mac-vrf-ep-t2-asymm-2-10 interface irb.1538
set routing-instances VRF-mac-vrf-ep-t2-asymm-2-10 interface irb.1539
set routing-instances VRF-mac-vrf-ep-t2-asymm-2-10 interface irb.1540
set routing-instances VRF-mac-vrf-ep-t2-asymm-2-10 interface lo0.1510
set routing-instances VRF-mac-vrf-ep-t2-asymm-2-10 route-distinguisher 192.168.4.3:1510
set routing-instances VRF-mac-vrf-ep-t2-asymm-2-10 vrf-table-label
set routing-instances VRF-mac-vrf-ep-t2-asymm-2-10 vrf-target target:100:1510
set routing-instances VRF-mac-vrf-ep-t2-asymm-transl-2-1 instance-type vrf
set routing-instances VRF-mac-vrf-ep-t2-asymm-transl-2-1 interface irb.1001
set routing-instances VRF-mac-vrf-ep-t2-asymm-transl-2-1 interface irb.1002
set routing-instances VRF-mac-vrf-ep-t2-asymm-transl-2-1 interface irb.1003
set routing-instances VRF-mac-vrf-ep-t2-asymm-transl-2-1 interface irb.1004
set routing-instances VRF-mac-vrf-ep-t2-asymm-transl-2-1 interface lo0.1001
set routing-instances VRF-mac-vrf-ep-t2-asymm-transl-2-1 route-distinguisher 192.168.4.3:1001
set routing-instances VRF-mac-vrf-ep-t2-asymm-transl-2-1 vrf-table-label
set routing-instances VRF-mac-vrf-ep-t2-asymm-transl-2-1 vrf-target target:100:1001
set routing-instances VRF-mac-vrf-ep-t2-asymm-transl-2-10 instance-type vrf
set routing-instances VRF-mac-vrf-ep-t2-asymm-transl-2-10 interface irb.1037
set routing-instances VRF-mac-vrf-ep-t2-asymm-transl-2-10 interface irb.1038
set routing-instances VRF-mac-vrf-ep-t2-asymm-transl-2-10 interface irb.1039
set routing-instances VRF-mac-vrf-ep-t2-asymm-transl-2-10 interface irb.1040
set routing-instances VRF-mac-vrf-ep-t2-asymm-transl-2-10 interface lo0.1010
set routing-instances VRF-mac-vrf-ep-t2-asymm-transl-2-10 route-distinguisher 192.168.4.3:1010
set routing-instances VRF-mac-vrf-ep-t2-asymm-transl-2-10 vrf-table-label
set routing-instances VRF-mac-vrf-ep-t2-asymm-transl-2-10 vrf-target target:100:1010
set routing-instances VRF-mac-vrf-ep-t2-symm-1-1 instance-type vrf
set routing-instances VRF-mac-vrf-ep-t2-symm-1-1 interface irb.501
set routing-instances VRF-mac-vrf-ep-t2-symm-1-1 interface irb.502
set routing-instances VRF-mac-vrf-ep-t2-symm-1-1 interface irb.503
set routing-instances VRF-mac-vrf-ep-t2-symm-1-1 interface irb.504
set routing-instances VRF-mac-vrf-ep-t2-symm-1-1 interface lo0.501
set routing-instances VRF-mac-vrf-ep-t2-symm-1-1 protocols evpn interconnect route-distinguisher 
192.168.4.3:44000
set routing-instances VRF-mac-vrf-ep-t2-symm-1-1 protocols evpn interconnect vrf-target 
target:5003:6001
set routing-instances VRF-mac-vrf-ep-t2-symm-1-1 protocols evpn ip-prefix-routes advertise 
direct-nexthop
set routing-instances VRF-mac-vrf-ep-t2-symm-1-1 protocols evpn ip-prefix-routes encapsulation 
vxlan
set routing-instances VRF-mac-vrf-ep-t2-symm-1-1 protocols evpn ip-prefix-routes vni 9100501
set routing-instances VRF-mac-vrf-ep-t2-symm-1-1 protocols evpn irb-symmetric-routing vni 9100501
set routing-instances VRF-mac-vrf-ep-t2-symm-1-1 route-distinguisher 192.168.4.3:501
set routing-instances VRF-mac-vrf-ep-t2-symm-1-1 routing-options multipath
set routing-instances VRF-mac-vrf-ep-t2-symm-1-1 routing-options rib VRF-mac-vrf-ep-t2-
symm-1-1.inet6.0 multipath
set routing-instances VRF-mac-vrf-ep-t2-symm-1-1 vrf-table-label
set routing-instances VRF-mac-vrf-ep-t2-symm-1-1 vrf-target export target:500:501
set routing-instances VRF-mac-vrf-ep-t2-symm-1-1 vrf-target import target:400:501
set routing-instances VRF-mac-vrf-ep-t2-symm-transl-1-1 instance-type vrf
set routing-instances VRF-mac-vrf-ep-t2-symm-transl-1-1 interface irb.2
set routing-instances VRF-mac-vrf-ep-t2-symm-transl-1-1 interface irb.3
set routing-instances VRF-mac-vrf-ep-t2-symm-transl-1-1 interface irb.4
set routing-instances VRF-mac-vrf-ep-t2-symm-transl-1-1 interface irb.4094
set routing-instances VRF-mac-vrf-ep-t2-symm-transl-1-1 interface irb.5
set routing-instances VRF-mac-vrf-ep-t2-symm-transl-1-1 interface lo0.2
set routing-instances VRF-mac-vrf-ep-t2-symm-transl-1-1 protocols evpn interconnect route-
distinguisher 192.168.4.3:48000
set routing-instances VRF-mac-vrf-ep-t2-symm-transl-1-1 protocols evpn interconnect vrf-target 
target:5005:6001
set routing-instances VRF-mac-vrf-ep-t2-symm-transl-1-1 protocols evpn ip-prefix-routes 
advertise direct-nexthop
set routing-instances VRF-mac-vrf-ep-t2-symm-transl-1-1 protocols evpn ip-prefix-routes 
encapsulation vxlan
```

```
set routing-instances VRF-mac-vrf-ep-t2-symm-transl-1-1 protocols evpn ip-prefix-routes vni
9100002
set routing-instances VRF-mac-vrf-ep-t2-symm-transl-1-1 protocols evpn irb-symmetric-routing vni
9100002
set routing-instances VRF-mac-vrf-ep-t2-symm-transl-1-1 route-distinguisher 192.168.4.3:2
set routing-instances VRF-mac-vrf-ep-t2-symm-transl-1-1 routing-options multipath
set routing-instances VRF-mac-vrf-ep-t2-symm-transl-1-1 routing-options rib VRF-mac-vrf-ep-t2-
symm-transl-1-1.inet6.0 multipath
set routing-instances VRF-mac-vrf-ep-t2-symm-transl-1-1 vrf-table-label
set routing-instances VRF-mac-vrf-ep-t2-symm-transl-1-1 vrf-target export target:500:2
set routing-instances VRF-mac-vrf-ep-t2-symm-transl-1-1 vrf-target import target:400:2
set routing-instances VRF-mac-vrf-ep-t2-symm-transl-Tenant-1 instance-type vrf
set routing-instances VRF-mac-vrf-ep-t2-symm-transl-Tenant-1 interface irb.4010
set routing-instances VRF-mac-vrf-ep-t2-symm-transl-Tenant-1 interface irb.4011
set routing-instances VRF-mac-vrf-ep-t2-symm-transl-Tenant-1 interface irb.4012
set routing-instances VRF-mac-vrf-ep-t2-symm-transl-Tenant-1 interface irb.4013
set routing-instances VRF-mac-vrf-ep-t2-symm-transl-Tenant-1 interface lo0.4010
set routing-instances VRF-mac-vrf-ep-t2-symm-transl-Tenant-1 protocols evpn interconnect route-
distinguisher 192.168.4.3:50000
set routing-instances VRF-mac-vrf-ep-t2-symm-transl-Tenant-1 protocols evpn interconnect vrf-
target target:5006:6001
set routing-instances VRF-mac-vrf-ep-t2-symm-transl-Tenant-1 protocols evpn ip-prefix-routes
advertise direct-nexthop
set routing-instances VRF-mac-vrf-ep-t2-symm-transl-Tenant-1 protocols evpn ip-prefix-routes
encapsulation vxlan
set routing-instances VRF-mac-vrf-ep-t2-symm-transl-Tenant-1 protocols evpn ip-prefix-routes vni
9104010
set routing-instances VRF-mac-vrf-ep-t2-symm-transl-Tenant-1 protocols evpn irb-symmetric-
routing vni 9104010
set routing-instances VRF-mac-vrf-ep-t2-symm-transl-Tenant-1 route-distinguisher 192.168.4.3:4010
set routing-instances VRF-mac-vrf-ep-t2-symm-transl-Tenant-1 routing-options multipath
set routing-instances VRF-mac-vrf-ep-t2-symm-transl-Tenant-1 routing-options rib VRF-mac-vrf-ep-
t2-symm-transl-Tenant-1.inet6.0 multipath
set routing-instances VRF-mac-vrf-ep-t2-symm-transl-Tenant-1 vrf-target export target:500:4010
set routing-instances VRF-mac-vrf-ep-t2-symm-transl-Tenant-1 vrf-target import target:400:4010
set routing-options forwarding-table chained-composite-next-hop ingress evpn
set routing-options forwarding-table ecmp-fast-reroute
set routing-options forwarding-table export per-packet-load-balance
set routing-options resolution rib bgp.rtarget.0 resolution-ribs inet.0
set routing-options router-id 192.168.4.3
set system arp aging-timer 60
set vlans UNDERLAY-VLAN-4004 l3-interface irb.4004
set vlans UNDERLAY-VLAN-4004 vlan-id 4004
```

**NOTE**: On QFX5130 and QFX5700 switches, also configure the `host-profile` unified forwarding profile option to support an EVPN-VXLAN environment (see *Layer 2 Forwarding Tables* for details):

**`set system packet-forwarding-options forwarding-profile host-profile`**

DC2-Leaf21

```
set chassis aggregated-devices ethernet device-count 64
set forwarding-options evpn-vxlan vxlan-trans-vni-enable
set forwarding-options vxlan-routing interface-num 8192
set forwarding-options vxlan-routing next-hop 49152
set forwarding-options vxlan-routing overlay-ecmp
set interfaces ae1 aggregated-ether-options lacp active
set interfaces ae1 aggregated-ether-options lacp periodic fast
set interfaces ae1 aggregated-ether-options minimum-links 1
set interfaces ae1 mtu 9192
set interfaces ae1 unit 0 family inet address 172.16.19.0/31
set interfaces ae1 unit 0 family inet mtu 9000
set interfaces ae1 unit 0 family iso
set interfaces ae2 aggregated-ether-options lacp active
set interfaces ae2 aggregated-ether-options lacp periodic fast
set interfaces ae2 aggregated-ether-options minimum-links 1
set interfaces ae2 mtu 9192
set interfaces ae2 unit 0 family inet address 172.16.20.0/31
set interfaces ae2 unit 0 family inet mtu 9000
set interfaces ae2 unit 0 family iso
set interfaces ae3 aggregated-ether-options lacp active
set interfaces ae3 aggregated-ether-options lacp hold-time up 10
set interfaces ae3 aggregated-ether-options lacp periodic fast
set interfaces ae3 aggregated-ether-options lacp system-id 00:00:00:99:99:01
set interfaces ae3 esi 00:00:00:ff:00:11:00:1:00:3
set interfaces ae3 esi all-active
set interfaces ae3 native-vlan-id 4094
set interfaces ae3 unit 0 family ethernet-switching interface-mode trunk
set interfaces ae3 unit 0 family ethernet-switching vlan members EP-TYPE-2-VLAN-10
set interfaces ae3 unit 0 family ethernet-switching vlan members EP-TYPE-2-VLAN-100
set interfaces ae3 unit 0 family ethernet-switching vlan members EP-TYPE-2-VLAN-101
set interfaces ae4 aggregated-ether-options lacp active
set interfaces ae4 aggregated-ether-options lacp hold-time up 10
set interfaces ae4 aggregated-ether-options lacp periodic fast
set interfaces ae4 aggregated-ether-options lacp system-id 00:00:00:99:99:01
set interfaces ae4 esi 00:00:00:ff:00:11:00:2:00:4
set interfaces ae4 esi all-active
set interfaces ae4 unit 0 family ethernet-switching interface-mode trunk
set interfaces ae4 unit 0 family ethernet-switching vlan members EP-TYPE-2-VLAN-501
set interfaces ae4 unit 0 family ethernet-switching vlan members EP-TYPE-2-VLAN-502
set interfaces ae4 unit 0 family ethernet-switching vlan members EP-TYPE-2-VLAN-503
set interfaces ae5 aggregated-ether-options lacp active
set interfaces ae5 aggregated-ether-options lacp hold-time up 10
set interfaces ae5 aggregated-ether-options lacp periodic fast
set interfaces ae5 aggregated-ether-options lacp system-id 00:00:00:99:99:01
set interfaces ae5 esi 00:00:00:ff:00:11:00:3:00:5
set interfaces ae5 esi all-active
set interfaces ae5 unit 0 family ethernet-switching interface-mode trunk
set interfaces ae5 unit 0 family ethernet-switching vlan members EP-TYPE-2-VLAN-1001
set interfaces ae5 unit 0 family ethernet-switching vlan members EP-TYPE-2-VLAN-1002
set interfaces ae5 unit 0 family ethernet-switching vlan members EP-TYPE-2-VLAN-1003
set interfaces ae6 aggregated-ether-options lacp active
set interfaces ae6 aggregated-ether-options lacp hold-time up 10
set interfaces ae6 aggregated-ether-options lacp periodic fast
set interfaces ae6 aggregated-ether-options lacp system-id 00:00:00:99:99:01
set interfaces ae6 esi 00:00:00:ff:00:11:00:4:00:6
set interfaces ae6 esi all-active
set interfaces ae6 unit 0 family ethernet-switching interface-mode trunk
set interfaces ae6 unit 0 family ethernet-switching vlan members EP-TYPE-2-VLAN-1501
set interfaces ae6 unit 0 family ethernet-switching vlan members EP-TYPE-2-VLAN-1502
set interfaces ae6 unit 0 family ethernet-switching vlan members EP-TYPE-2-VLAN-1503
set interfaces ae7 aggregated-ether-options lacp active
set interfaces ae7 aggregated-ether-options lacp hold-time up 10
set interfaces ae7 aggregated-ether-options lacp periodic fast
set interfaces ae7 aggregated-ether-options lacp system-id 00:00:00:99:99:01
set interfaces ae7 esi 00:00:00:ff:00:11:00:5:00:7
set interfaces ae7 esi all-active
set interfaces ae7 unit 0 family ethernet-switching interface-mode trunk
set interfaces ae7 unit 0 family ethernet-switching vlan members EP-TYPE-2-VLAN-3001
set interfaces ae7 unit 0 family ethernet-switching vlan members EP-TYPE-2-VLAN-3002
set interfaces ae7 unit 0 family ethernet-switching vlan members EP-TYPE-2-VLAN-3003
set interfaces ae7 unit 0 family ethernet-switching vlan members EP-TYPE-2-VLAN-3004
set interfaces ae8 aggregated-ether-options lacp active
set interfaces ae8 aggregated-ether-options lacp hold-time up 10
set interfaces ae8 aggregated-ether-options lacp periodic fast
set interfaces ae8 aggregated-ether-options lacp system-id 00:00:00:99:99:01
set interfaces ae8 esi 00:00:00:ff:00:11:00:6:00:8
set interfaces ae8 esi all-active
set interfaces ae8 unit 0 family ethernet-switching interface-mode trunk
set interfaces ae8 unit 0 family ethernet-switching vlan members EP-TYPE-2-VLAN-4010
set interfaces ae8 unit 0 family ethernet-switching vlan members EP-TYPE-2-VLAN-4011
set interfaces ae8 unit 0 family ethernet-switching vlan members EP-TYPE-2-VLAN-4012
set interfaces ae8 unit 0 family ethernet-switching vlan members EP-TYPE-2-VLAN-4013
set interfaces et-0/0/0 ether-options 802.3ad ae7
set interfaces et-0/0/0 hold-time down 1000
set interfaces et-0/0/0 hold-time up 500000
set interfaces et-0/0/1 ether-options 802.3ad ae4
set interfaces et-0/0/1 hold-time down 1000
set interfaces et-0/0/1 hold-time up 500000
set interfaces et-0/0/11 ether-options 802.3ad ae1
set interfaces et-0/0/12 ether-options 802.3ad ae2
set interfaces et-0/0/13 ether-options 802.3ad ae2
set interfaces et-0/0/2 ether-options 802.3ad ae8
set interfaces et-0/0/2 hold-time down 1000
set interfaces et-0/0/2 hold-time up 500000
set interfaces et-0/0/3 ether-options 802.3ad ae3
set interfaces et-0/0/3 hold-time down 1000
set interfaces et-0/0/3 hold-time up 500000
set interfaces et-0/0/4 ether-options 802.3ad ae5
set interfaces et-0/0/4 hold-time down 1000
set interfaces et-0/0/4 hold-time up 500000
set interfaces et-0/0/6 ether-options 802.3ad ae6
set interfaces et-0/0/6 hold-time down 1000
set interfaces et-0/0/6 hold-time up 500000
set interfaces et-0/0/9 ether-options 802.3ad ae1
set interfaces irb mtu 1500
set interfaces irb unit 10 family inet address 10.120.10.248/24 preferred
set interfaces irb unit 10 family inet address 10.120.10.248/24 virtual-gateway-address
10.120.10.254
set interfaces irb unit 10 family inet6 address 2001:db8::12:0:a:248/112 preferred
set interfaces irb unit 10 family inet6 address 2001:db8::12:0:a:248/112 virtual-gateway-address
2001:db8::12:0:a:254
set interfaces irb unit 10 no-auto-virtual-gateway-esi
set interfaces irb unit 10 virtual-gateway-accept-data
set interfaces irb unit 10 virtual-gateway-v4-mac 00:00:5e:00:00:04
set interfaces irb unit 10 virtual-gateway-v6-mac 00:00:5e:00:00:04
set interfaces irb unit 100 family inet address 10.120.100.248/24 preferred
set interfaces irb unit 100 family inet address 10.120.100.248/24 virtual-gateway-address
10.120.100.254
set interfaces irb unit 100 family inet6 address 2001:db8::12:0:64:248/112 preferred
set interfaces irb unit 100 family inet6 address 2001:db8::12:0:64:248/112 virtual-gateway-
address 2001:db8::12:0:64:254
set interfaces irb unit 100 no-auto-virtual-gateway-esi
set interfaces irb unit 100 virtual-gateway-accept-data
set interfaces irb unit 100 virtual-gateway-v4-mac 00:00:5e:00:00:04
set interfaces irb unit 1001 family inet address 10.33.233.248/24 preferred
set interfaces irb unit 1001 family inet address 10.33.233.248/24 virtual-gateway-address
10.33.233.254
set interfaces irb unit 1001 family inet6 address 2001:db8::30:0:3e9:248/112 preferred
set interfaces irb unit 1001 family inet6 address 2001:db8::30:0:3e9:248/112 virtual-gateway-
address 2001:db8::30:0:3e9:254
set interfaces irb unit 1001 no-auto-virtual-gateway-esi
set interfaces irb unit 1001 virtual-gateway-accept-data
set interfaces irb unit 1001 virtual-gateway-v4-mac 00:00:5e:00:00:04
set interfaces irb unit 1001 virtual-gateway-v6-mac 00:00:5e:00:00:04
set interfaces irb unit 1002 family inet address 10.33.234.248/24 preferred
set interfaces irb unit 1002 family inet address 10.33.234.248/24 virtual-gateway-address
10.33.234.254
set interfaces irb unit 1002 family inet6 address 2001:db8::30:0:3ea:248/112 preferred
set interfaces irb unit 1002 family inet6 address 2001:db8::30:0:3ea:248/112 virtual-gateway-
address 2001:db8::30:0:3ea:254
set interfaces irb unit 1002 no-auto-virtual-gateway-esi
set interfaces irb unit 1002 virtual-gateway-accept-data
set interfaces irb unit 1002 virtual-gateway-v4-mac 00:00:5e:00:00:04
set interfaces irb unit 1002 virtual-gateway-v6-mac 00:00:5e:00:00:04
set interfaces irb unit <*> family inet6 nd6-stale-time 3600
set interfaces lo0 unit 0 family inet address 192.168.0.4/32 preferred
set interfaces lo0 unit 0 family inet address 192.168.0.4/32 primary
set interfaces lo0 unit 10 family inet
set interfaces lo0 unit 1001 family inet
set policy-options policy-statement per-packet-load-balance term 1 then load-balance per-packet
set policy-options policy-statement underlay-clos-export term loopback from interface lo0.0
set policy-options policy-statement underlay-clos-export term loopback then accept
set protocols bgp graceful-restart
set protocols bgp group overlay-ebgp authentication-key overlay_ebgp_admin
set protocols bgp group overlay-ebgp bfd-liveness-detection minimum-interval 1000
set protocols bgp group overlay-ebgp bfd-liveness-detection multiplier 3
set protocols bgp group overlay-ebgp bfd-liveness-detection session-mode automatic
set protocols bgp group overlay-ebgp family evpn signaling
set protocols bgp group overlay-ebgp family evpn signaling delay-route-advertisements minimum-
```

```
delay routing-uptime 400
set protocols bgp group overlay-ebgp local-address 192.168.0.4
set protocols bgp group overlay-ebgp local-as 4200000014
set protocols bgp group overlay-ebgp multihop no-nexthop-change
set protocols bgp group overlay-ebgp multipath multiple-as
set protocols bgp group overlay-ebgp neighbor 192.168.6.1 peer-as 4200000042
set protocols bgp group overlay-ebgp neighbor 192.168.6.2 peer-as 4200000042
set protocols bgp group overlay-ebgp type external
set protocols bgp group overlay-ebgp vpn-apply-export
set protocols bgp group underlay-bgp authentication-key underlay_ebgp_admin
set protocols bgp group underlay-bgp bfd-liveness-detection minimum-interval 1000
set protocols bgp group underlay-bgp bfd-liveness-detection multiplier 3
set protocols bgp group underlay-bgp bfd-liveness-detection session-mode automatic
set protocols bgp group underlay-bgp export underlay-clos-export
set protocols bgp group underlay-bgp family inet
set protocols bgp group underlay-bgp local-as 4200000014
set protocols bgp group underlay-bgp multipath multiple-as
set protocols bgp group underlay-bgp neighbor 172.16.19.1 peer-as 4200000042
set protocols bgp group underlay-bgp neighbor 172.16.20.1 peer-as 4200000042
set protocols bgp group underlay-bgp type external
set protocols bgp log-updown
set protocols evpn default-gateway no-gateway-community
set protocols evpn encapsulation vxlan
set protocols evpn extended-vni-list all
set protocols l2-learning global-mac-ip-table-aging-time 3600
set protocols l2-learning global-mac-table-aging-time 4200
set protocols lldp interface all
set routing-instances MACVRF-mac-vrf-ep-t2-asymm-2-1 instance-type mac-vrf
set routing-instances MACVRF-mac-vrf-ep-t2-asymm-2-1 interface ae6.0
set routing-instances MACVRF-mac-vrf-ep-t2-asymm-2-1 protocols evpn default-gateway no-gateway-
community
set routing-instances MACVRF-mac-vrf-ep-t2-asymm-2-1 protocols evpn encapsulation vxlan
set routing-instances MACVRF-mac-vrf-ep-t2-asymm-2-1 protocols evpn extended-vni-list all
set routing-instances MACVRF-mac-vrf-ep-t2-asymm-2-1 protocols evpn vni-options vni 401501 vrf-
target target:2:401501
set routing-instances MACVRF-mac-vrf-ep-t2-asymm-2-1 protocols evpn vni-options vni 401502 vrf-
target target:2:401502
set routing-instances MACVRF-mac-vrf-ep-t2-asymm-2-1 protocols evpn vni-options vni 401503 vrf-
target target:2:401503
set routing-instances MACVRF-mac-vrf-ep-t2-asymm-2-1 route-distinguisher 192.168.0.4:34801
set routing-instances MACVRF-mac-vrf-ep-t2-asymm-2-1 service-type vlan-aware
set routing-instances MACVRF-mac-vrf-ep-t2-asymm-2-1 vlans EP-TYPE-2-VLAN-1501 l3-interface
irb.1501
```

```
set routing-instances MACVRF-mac-vrf-ep-t2-asymm-2-1 vlans EP-TYPE-2-VLAN-1501 vlan-id 1501
set routing-instances MACVRF-mac-vrf-ep-t2-asymm-2-1 vlans EP-TYPE-2-VLAN-1501 vxlan vni 401501
set routing-instances MACVRF-mac-vrf-ep-t2-asymm-2-1 vlans EP-TYPE-2-VLAN-1502 l3-interface
irb.1502
set routing-instances MACVRF-mac-vrf-ep-t2-asymm-2-1 vlans EP-TYPE-2-VLAN-1502 vlan-id 1502
set routing-instances MACVRF-mac-vrf-ep-t2-asymm-2-1 vlans EP-TYPE-2-VLAN-1502 vxlan vni 401502
set routing-instances MACVRF-mac-vrf-ep-t2-asymm-2-1 vrf-target target:33302:1501
set routing-instances MACVRF-mac-vrf-ep-t2-asymm-2-1 vtep-source-interface lo0.0
set routing-instances MACVRF-mac-vrf-ep-t2-asymm-transl-2-1 instance-type mac-vrf
set routing-instances MACVRF-mac-vrf-ep-t2-asymm-transl-2-1 interface ae5.0
set routing-instances MACVRF-mac-vrf-ep-t2-asymm-transl-2-1 protocols evpn default-gateway no-
gateway-community
set routing-instances MACVRF-mac-vrf-ep-t2-asymm-transl-2-1 protocols evpn encapsulation vxlan
set routing-instances MACVRF-mac-vrf-ep-t2-asymm-transl-2-1 protocols evpn extended-vni-list all
set routing-instances MACVRF-mac-vrf-ep-t2-asymm-transl-2-1 protocols evpn vni-options vni
311001 vrf-target target:2:311001
set routing-instances MACVRF-mac-vrf-ep-t2-asymm-transl-2-1 protocols evpn vni-options vni
311002 vrf-target target:2:311002
set routing-instances MACVRF-mac-vrf-ep-t2-asymm-transl-2-1 protocols evpn vni-options vni
311003 vrf-target target:2:311003
set routing-instances MACVRF-mac-vrf-ep-t2-asymm-transl-2-1 route-distinguisher 192.168.0.4:34301
set routing-instances MACVRF-mac-vrf-ep-t2-asymm-transl-2-1 service-type vlan-aware
set routing-instances MACVRF-mac-vrf-ep-t2-asymm-transl-2-1 vlans EP-TYPE-2-VLAN-1001 l3-
interface irb.1001
set routing-instances MACVRF-mac-vrf-ep-t2-asymm-transl-2-1 vlans EP-TYPE-2-VLAN-1001 vlan-id
1001
set routing-instances MACVRF-mac-vrf-ep-t2-asymm-transl-2-1 vlans EP-TYPE-2-VLAN-1001 vxlan vni
311001
set routing-instances MACVRF-mac-vrf-ep-t2-asymm-transl-2-1 vlans EP-TYPE-2-VLAN-1002 l3-
interface irb.1002
set routing-instances MACVRF-mac-vrf-ep-t2-asymm-transl-2-1 vlans EP-TYPE-2-VLAN-1002 vlan-id
1002
set routing-instances MACVRF-mac-vrf-ep-t2-asymm-transl-2-1 vlans EP-TYPE-2-VLAN-1002 vxlan vni
311002
set routing-instances MACVRF-mac-vrf-ep-t2-asymm-transl-2-1 vlans EP-TYPE-2-VLAN-1003 l3-
interface irb.1003
set routing-instances MACVRF-mac-vrf-ep-t2-asymm-transl-2-1 vlans EP-TYPE-2-VLAN-1003 vlan-id
1003
set routing-instances MACVRF-mac-vrf-ep-t2-asymm-transl-2-1 vlans EP-TYPE-2-VLAN-1003 vxlan vni
311003
set routing-instances MACVRF-mac-vrf-ep-t2-asymm-transl-2-1 vrf-target target:33302:1001
set routing-instances MACVRF-mac-vrf-ep-t2-asymm-transl-2-1 vtep-source-interface lo0.0
set routing-instances MACVRF-mac-vrf-ep-t2-symm-1-1 instance-type mac-vrf
set routing-instances MACVRF-mac-vrf-ep-t2-symm-1-1 interface ae4.0
set routing-instances MACVRF-mac-vrf-ep-t2-symm-1-1 protocols evpn default-gateway no-gateway-
community
set routing-instances MACVRF-mac-vrf-ep-t2-symm-1-1 protocols evpn encapsulation vxlan
set routing-instances MACVRF-mac-vrf-ep-t2-symm-1-1 protocols evpn extended-vni-list all
set routing-instances MACVRF-mac-vrf-ep-t2-symm-1-1 route-distinguisher 192.168.0.4:33801
set routing-instances MACVRF-mac-vrf-ep-t2-symm-1-1 service-type vlan-aware
set routing-instances MACVRF-mac-vrf-ep-t2-symm-1-1 vlans EP-TYPE-2-VLAN-501 l3-interface irb.501
set routing-instances MACVRF-mac-vrf-ep-t2-symm-1-1 vlans EP-TYPE-2-VLAN-501 vlan-id 501
set routing-instances MACVRF-mac-vrf-ep-t2-symm-1-1 vlans EP-TYPE-2-VLAN-501 vxlan vni 200501
set routing-instances MACVRF-mac-vrf-ep-t2-symm-1-1 vlans EP-TYPE-2-VLAN-502 l3-interface irb.502
set routing-instances MACVRF-mac-vrf-ep-t2-symm-1-1 vlans EP-TYPE-2-VLAN-502 vlan-id 502
set routing-instances MACVRF-mac-vrf-ep-t2-symm-1-1 vlans EP-TYPE-2-VLAN-502 vxlan vni 200502
set routing-instances MACVRF-mac-vrf-ep-t2-symm-1-1 vlans EP-TYPE-2-VLAN-503 l3-interface irb.503
set routing-instances MACVRF-mac-vrf-ep-t2-symm-1-1 vlans EP-TYPE-2-VLAN-503 vlan-id 503
set routing-instances MACVRF-mac-vrf-ep-t2-symm-1-1 vlans EP-TYPE-2-VLAN-503 vxlan vni 200503
set routing-instances MACVRF-mac-vrf-ep-t2-symm-1-1 vrf-target target:33302:501
set routing-instances MACVRF-mac-vrf-ep-t2-symm-1-1 vtep-source-interface lo0.0
set routing-instances MACVRF-mac-vrf-ep-t2-symm-dhcp-1 instance-type mac-vrf
set routing-instances MACVRF-mac-vrf-ep-t2-symm-dhcp-1 interface ae7.0
set routing-instances MACVRF-mac-vrf-ep-t2-symm-dhcp-1 protocols evpn default-gateway no-gateway-
community
set routing-instances MACVRF-mac-vrf-ep-t2-symm-dhcp-1 protocols evpn encapsulation vxlan
set routing-instances MACVRF-mac-vrf-ep-t2-symm-dhcp-1 protocols evpn extended-vni-list all
set routing-instances MACVRF-mac-vrf-ep-t2-symm-dhcp-1 route-distinguisher 192.168.0.4:36301
set routing-instances MACVRF-mac-vrf-ep-t2-symm-dhcp-1 service-type vlan-aware
set routing-instances MACVRF-mac-vrf-ep-t2-symm-dhcp-1 vlans EP-TYPE-2-VLAN-3001 l3-interface
irb.3001
set routing-instances MACVRF-mac-vrf-ep-t2-symm-dhcp-1 vlans EP-TYPE-2-VLAN-3001 vlan-id 3001
set routing-instances MACVRF-mac-vrf-ep-t2-symm-dhcp-1 vlans EP-TYPE-2-VLAN-3001 vxlan vni 223001
set routing-instances MACVRF-mac-vrf-ep-t2-symm-dhcp-1 vlans EP-TYPE-2-VLAN-3002 l3-interface
irb.3002
set routing-instances MACVRF-mac-vrf-ep-t2-symm-dhcp-1 vlans EP-TYPE-2-VLAN-3002 vlan-id 3002
set routing-instances MACVRF-mac-vrf-ep-t2-symm-dhcp-1 vlans EP-TYPE-2-VLAN-3002 vxlan vni 223002
set routing-instances MACVRF-mac-vrf-ep-t2-symm-dhcp-1 vlans EP-TYPE-2-VLAN-3003 l3-interface
irb.3003
set routing-instances MACVRF-mac-vrf-ep-t2-symm-dhcp-1 vlans EP-TYPE-2-VLAN-3003 vlan-id 3003
set routing-instances MACVRF-mac-vrf-ep-t2-symm-dhcp-1 vlans EP-TYPE-2-VLAN-3003 vxlan vni 223003
set routing-instances MACVRF-mac-vrf-ep-t2-symm-dhcp-1 vlans EP-TYPE-2-VLAN-3004 l3-interface
irb.3004
set routing-instances MACVRF-mac-vrf-ep-t2-symm-dhcp-1 vlans EP-TYPE-2-VLAN-3004 vlan-id 3004
set routing-instances MACVRF-mac-vrf-ep-t2-symm-dhcp-1 vlans EP-TYPE-2-VLAN-3004 vxlan vni 223004
set routing-instances MACVRF-mac-vrf-ep-t2-symm-dhcp-1 vrf-target target:33302:3001
set routing-instances MACVRF-mac-vrf-ep-t2-symm-dhcp-1 vtep-source-interface lo0.0
set routing-instances MACVRF-mac-vrf-ep-t2-symm-transl-1-1 instance-type mac-vrf
set routing-instances MACVRF-mac-vrf-ep-t2-symm-transl-1-1 interface ae3.0
set routing-instances MACVRF-mac-vrf-ep-t2-symm-transl-1-1 protocols evpn default-gateway no-
gateway-community
set routing-instances MACVRF-mac-vrf-ep-t2-symm-transl-1-1 protocols evpn encapsulation vxlan
set routing-instances MACVRF-mac-vrf-ep-t2-symm-transl-1-1 protocols evpn extended-vni-list all
set routing-instances MACVRF-mac-vrf-ep-t2-symm-transl-1-1 route-distinguisher 192.168.0.4:33302
set routing-instances MACVRF-mac-vrf-ep-t2-symm-transl-1-1 service-type vlan-aware
set routing-instances MACVRF-mac-vrf-ep-t2-symm-transl-1-1 vlans EP-TYPE-2-VLAN-10 vlan-id 10
set routing-instances MACVRF-mac-vrf-ep-t2-symm-transl-1-1 vlans EP-TYPE-2-VLAN-10 vxlan vni
120010
set routing-instances MACVRF-mac-vrf-ep-t2-symm-transl-1-1 vlans EP-TYPE-2-VLAN-2 l3-interface
irb.2
set routing-instances MACVRF-mac-vrf-ep-t2-symm-transl-1-1 vlans EP-TYPE-2-VLAN-2 vlan-id 2
set routing-instances MACVRF-mac-vrf-ep-t2-symm-transl-1-1 vlans EP-TYPE-2-VLAN-2 vxlan vni
120002
set routing-instances MACVRF-mac-vrf-ep-t2-symm-transl-1-1 vlans EP-TYPE-2-VLAN-3 l3-interface
irb.3
set routing-instances MACVRF-mac-vrf-ep-t2-symm-transl-1-1 vlans EP-TYPE-2-VLAN-3 vlan-id 3
set routing-instances MACVRF-mac-vrf-ep-t2-symm-transl-1-1 vlans EP-TYPE-2-VLAN-3 vxlan vni
120003
set routing-instances MACVRF-mac-vrf-ep-t2-symm-transl-1-1 vrf-target target:33302:2
set routing-instances MACVRF-mac-vrf-ep-t2-symm-transl-1-1 vtep-source-interface lo0.0
set routing-instances MACVRF-mac-vrf-ep-t2-symm-transl-Tenant-1 instance-type mac-vrf
set routing-instances MACVRF-mac-vrf-ep-t2-symm-transl-Tenant-1 interface ae8.0
set routing-instances MACVRF-mac-vrf-ep-t2-symm-transl-Tenant-1 protocols evpn default-gateway
no-gateway-community
set routing-instances MACVRF-mac-vrf-ep-t2-symm-transl-Tenant-1 protocols evpn encapsulation
vxlan
set routing-instances MACVRF-mac-vrf-ep-t2-symm-transl-Tenant-1 protocols evpn extended-vni-list
all
set routing-instances MACVRF-mac-vrf-ep-t2-symm-transl-Tenant-1 route-distinguisher
192.168.0.4:37310
set routing-instances MACVRF-mac-vrf-ep-t2-symm-transl-Tenant-1 service-type vlan-aware
set routing-instances MACVRF-mac-vrf-ep-t2-symm-transl-Tenant-1 vlans EP-TYPE-2-VLAN-4010 l3-
interface irb.4010
set routing-instances MACVRF-mac-vrf-ep-t2-symm-transl-Tenant-1 vlans EP-TYPE-2-VLAN-4010 vlan-
id 4010
set routing-instances MACVRF-mac-vrf-ep-t2-symm-transl-Tenant-1 vlans EP-TYPE-2-VLAN-4010 vxlan
vni 244010
set routing-instances MACVRF-mac-vrf-ep-t2-symm-transl-Tenant-1 vlans EP-TYPE-2-VLAN-4011 l3-
interface irb.4011
```

```
set routing-instances MACVRF-mac-vrf-ep-t2-symm-transl-Tenant-1 vlans EP-TYPE-2-VLAN-4011 vlan-
id 4011
set routing-instances MACVRF-mac-vrf-ep-t2-symm-transl-Tenant-1 vlans EP-TYPE-2-VLAN-4011 vxlan
vni 244011
set routing-instances MACVRF-mac-vrf-ep-t2-symm-transl-Tenant-1 vlans EP-TYPE-2-VLAN-4012 l3-
interface irb.4012
set routing-instances MACVRF-mac-vrf-ep-t2-symm-transl-Tenant-1 vlans EP-TYPE-2-VLAN-4012 vlan-
id 4012
set routing-instances MACVRF-mac-vrf-ep-t2-symm-transl-Tenant-1 vlans EP-TYPE-2-VLAN-4012 vxlan
vni 244012
set routing-instances MACVRF-mac-vrf-ep-t2-symm-transl-Tenant-1 vlans EP-TYPE-2-VLAN-4013 l3-
interface irb.4013
set routing-instances MACVRF-mac-vrf-ep-t2-symm-transl-Tenant-1 vlans EP-TYPE-2-VLAN-4013 vlan-
id 4013
set routing-instances MACVRF-mac-vrf-ep-t2-symm-transl-Tenant-1 vlans EP-TYPE-2-VLAN-4013 vxlan
vni 244013
set routing-instances MACVRF-mac-vrf-ep-t2-symm-transl-Tenant-1 vrf-target target:33302:4010
set routing-instances MACVRF-mac-vrf-ep-t2-symm-transl-Tenant-1 vtep-source-interface lo0.0
set routing-instances VRF-mac-vrf-ep-t2-asymm-2-1 instance-type vrf
set routing-instances VRF-mac-vrf-ep-t2-asymm-2-1 interface irb.1501
set routing-instances VRF-mac-vrf-ep-t2-asymm-2-1 interface irb.1502
set routing-instances VRF-mac-vrf-ep-t2-asymm-2-1 interface irb.1503
set routing-instances VRF-mac-vrf-ep-t2-asymm-2-1 interface irb.1504
set routing-instances VRF-mac-vrf-ep-t2-asymm-2-1 interface lo0.1501
set routing-instances VRF-mac-vrf-ep-t2-asymm-2-1 route-distinguisher 192.168.0.4:1501
set routing-instances VRF-mac-vrf-ep-t2-asymm-2-1 vrf-table-label
set routing-instances VRF-mac-vrf-ep-t2-asymm-2-1 vrf-target target:100:1501
set routing-instances VRF-mac-vrf-ep-t2-asymm-2-10 instance-type vrf
set routing-instances VRF-mac-vrf-ep-t2-asymm-2-10 interface irb.1537
set routing-instances VRF-mac-vrf-ep-t2-asymm-2-10 interface irb.1538
set routing-instances VRF-mac-vrf-ep-t2-asymm-2-10 interface irb.1539
set routing-instances VRF-mac-vrf-ep-t2-asymm-2-10 interface irb.1540
set routing-instances VRF-mac-vrf-ep-t2-asymm-2-10 interface lo0.1510
set routing-instances VRF-mac-vrf-ep-t2-asymm-2-10 route-distinguisher 192.168.0.4:1510
set routing-instances VRF-mac-vrf-ep-t2-asymm-2-10 vrf-table-label
set routing-instances VRF-mac-vrf-ep-t2-asymm-2-10 vrf-target target:100:1510
set routing-instances VRF-mac-vrf-ep-t2-asymm-2-11 instance-type vrf
set routing-instances VRF-mac-vrf-ep-t2-asymm-2-11 interface irb.1541
set routing-instances VRF-mac-vrf-ep-t2-asymm-2-11 interface irb.1542
set routing-instances VRF-mac-vrf-ep-t2-asymm-2-11 interface irb.1543
set routing-instances VRF-mac-vrf-ep-t2-asymm-2-11 interface irb.1544
set routing-instances VRF-mac-vrf-ep-t2-asymm-2-11 interface lo0.1511
set routing-instances VRF-mac-vrf-ep-t2-asymm-transl-2-1 instance-type vrf
set routing-instances VRF-mac-vrf-ep-t2-asymm-transl-2-1 interface irb.1001
set routing-instances VRF-mac-vrf-ep-t2-asymm-transl-2-1 interface irb.1002
set routing-instances VRF-mac-vrf-ep-t2-asymm-transl-2-1 interface irb.1003
set routing-instances VRF-mac-vrf-ep-t2-asymm-transl-2-1 interface irb.1004
set routing-instances VRF-mac-vrf-ep-t2-asymm-transl-2-1 interface lo0.1001
set routing-instances VRF-mac-vrf-ep-t2-asymm-transl-2-1 route-distinguisher 192.168.0.4:1001
set routing-instances VRF-mac-vrf-ep-t2-asymm-transl-2-1 vrf-table-label
set routing-instances VRF-mac-vrf-ep-t2-asymm-transl-2-1 vrf-target target:100:1001
set routing-instances VRF-mac-vrf-ep-t2-asymm-transl-2-10 instance-type vrf
set routing-instances VRF-mac-vrf-ep-t2-asymm-transl-2-10 interface irb.1037
set routing-instances VRF-mac-vrf-ep-t2-asymm-transl-2-10 interface irb.1038
set routing-instances VRF-mac-vrf-ep-t2-asymm-transl-2-10 interface irb.1039
set routing-instances VRF-mac-vrf-ep-t2-asymm-transl-2-10 interface irb.1040
set routing-instances VRF-mac-vrf-ep-t2-asymm-transl-2-10 interface lo0.1010
set routing-instances VRF-mac-vrf-ep-t2-asymm-transl-2-10 route-distinguisher 192.168.0.4:1010
set routing-instances VRF-mac-vrf-ep-t2-asymm-transl-2-10 vrf-table-label
set routing-instances VRF-mac-vrf-ep-t2-asymm-transl-2-10 vrf-target target:100:1010
set routing-instances VRF-mac-vrf-ep-t2-asymm-transl-2-11 instance-type vrf
set routing-instances VRF-mac-vrf-ep-t2-asymm-transl-2-11 interface irb.1041
set routing-instances VRF-mac-vrf-ep-t2-asymm-transl-2-11 interface irb.1042
set routing-instances VRF-mac-vrf-ep-t2-asymm-transl-2-11 interface irb.1043
set routing-instances VRF-mac-vrf-ep-t2-asymm-transl-2-11 interface irb.1044
set routing-instances VRF-mac-vrf-ep-t2-asymm-transl-2-11 interface lo0.1011
set routing-instances VRF-mac-vrf-ep-t2-asymm-transl-2-11 route-distinguisher 192.168.0.4:1011
set routing-instances VRF-mac-vrf-ep-t2-asymm-transl-2-11 vrf-table-label
set routing-instances VRF-mac-vrf-ep-t2-asymm-transl-2-11 vrf-target target:100:1011
set routing-instances VRF-mac-vrf-ep-t2-symm-1-1 instance-type vrf
set routing-instances VRF-mac-vrf-ep-t2-symm-1-1 interface irb.501
set routing-instances VRF-mac-vrf-ep-t2-symm-1-1 interface irb.502
set routing-instances VRF-mac-vrf-ep-t2-symm-1-1 interface irb.503
set routing-instances VRF-mac-vrf-ep-t2-symm-1-1 interface irb.504
set routing-instances VRF-mac-vrf-ep-t2-symm-1-1 interface lo0.501
set routing-instances VRF-mac-vrf-ep-t2-symm-1-1 protocols evpn ip-prefix-routes advertise
direct-nexthop
set routing-instances VRF-mac-vrf-ep-t2-symm-1-1 protocols evpn ip-prefix-routes encapsulation
vxlan
set routing-instances VRF-mac-vrf-ep-t2-symm-1-1 protocols evpn ip-prefix-routes vni 9100501
set routing-instances VRF-mac-vrf-ep-t2-symm-1-1 protocols evpn irb-symmetric-routing vni 9100501
set routing-instances VRF-mac-vrf-ep-t2-symm-1-1 route-distinguisher 192.168.0.4:501
set routing-instances VRF-mac-vrf-ep-t2-symm-1-1 routing-options multipath
set routing-instances VRF-mac-vrf-ep-t2-symm-1-1 routing-options rib VRF-mac-vrf-ep-t2-
symm-1-1.inet6.0 multipath
set routing-instances VRF-mac-vrf-ep-t2-symm-1-1 vrf-table-label
set routing-instances VRF-mac-vrf-ep-t2-symm-1-1 vrf-target export target:400:501
set routing-instances VRF-mac-vrf-ep-t2-symm-1-1 vrf-target import target:500:501
set routing-instances VRF-mac-vrf-ep-t2-symm-1-10 instance-type vrf
set routing-instances VRF-mac-vrf-ep-t2-symm-1-10 interface irb.537
set routing-instances VRF-mac-vrf-ep-t2-symm-1-10 interface irb.538
set routing-instances VRF-mac-vrf-ep-t2-symm-1-10 interface irb.539
set routing-instances VRF-mac-vrf-ep-t2-symm-1-10 interface irb.540
set routing-instances VRF-mac-vrf-ep-t2-symm-1-10 interface lo0.510
set routing-instances VRF-mac-vrf-ep-t2-symm-1-10 protocols evpn ip-prefix-routes advertise
direct-nexthop
set routing-instances VRF-mac-vrf-ep-t2-symm-1-10 protocols evpn ip-prefix-routes encapsulation
vxlan
set routing-instances VRF-mac-vrf-ep-t2-symm-1-10 protocols evpn ip-prefix-routes vni 9100510
set routing-instances VRF-mac-vrf-ep-t2-symm-1-10 protocols evpn irb-symmetric-routing vni
9100510
set routing-instances VRF-mac-vrf-ep-t2-symm-1-10 route-distinguisher 192.168.0.4:510
set routing-instances VRF-mac-vrf-ep-t2-symm-1-10 routing-options multipath
set routing-instances VRF-mac-vrf-ep-t2-symm-1-10 routing-options rib VRF-mac-vrf-ep-t2-
symm-1-10.inet6.0 multipath
set routing-instances VRF-mac-vrf-ep-t2-symm-1-10 vrf-table-label
set routing-instances VRF-mac-vrf-ep-t2-symm-1-10 vrf-target export target:400:510
set routing-instances VRF-mac-vrf-ep-t2-symm-1-10 vrf-target import target:500:510
set routing-instances VRF-mac-vrf-ep-t2-symm-1-11 instance-type vrf
set routing-instances VRF-mac-vrf-ep-t2-symm-1-11 interface irb.541
set routing-instances VRF-mac-vrf-ep-t2-symm-1-11 interface irb.542
set routing-instances VRF-mac-vrf-ep-t2-symm-1-11 interface irb.543
set routing-instances VRF-mac-vrf-ep-t2-symm-1-11 interface irb.544
set routing-instances VRF-mac-vrf-ep-t2-symm-1-11 interface lo0.511
set routing-instances VRF-mac-vrf-ep-t2-symm-transl-1-1 instance-type vrf
set routing-instances VRF-mac-vrf-ep-t2-symm-transl-1-1 interface irb.2
set routing-instances VRF-mac-vrf-ep-t2-symm-transl-1-1 interface irb.3
set routing-instances VRF-mac-vrf-ep-t2-symm-transl-1-1 interface irb.4
set routing-instances VRF-mac-vrf-ep-t2-symm-transl-1-1 interface irb.4094
set routing-instances VRF-mac-vrf-ep-t2-symm-transl-1-1 interface irb.5
set routing-instances VRF-mac-vrf-ep-t2-symm-transl-1-1 interface lo0.2
set routing-instances VRF-mac-vrf-ep-t2-symm-transl-1-1 protocols evpn ip-prefix-routes
advertise direct-nexthop
set routing-instances VRF-mac-vrf-ep-t2-symm-transl-1-1 protocols evpn ip-prefix-routes
encapsulation vxlan
set routing-instances VRF-mac-vrf-ep-t2-symm-transl-1-1 protocols evpn ip-prefix-routes vni
9100002
set routing-instances VRF-mac-vrf-ep-t2-symm-transl-1-1 protocols evpn irb-symmetric-routing vni
9100002
```

```
set routing-instances VRF-mac-vrf-ep-t2-symm-transl-1-1 route-distinguisher 192.168.0.4:2
set routing-instances VRF-mac-vrf-ep-t2-symm-transl-1-1 routing-options multipath
set routing-instances VRF-mac-vrf-ep-t2-symm-transl-1-1 routing-options rib VRF-mac-vrf-ep-t2-
symm-transl-1-1.inet6.0 multipath
set routing-instances VRF-mac-vrf-ep-t2-symm-transl-1-1 vrf-table-label
set routing-instances VRF-mac-vrf-ep-t2-symm-transl-1-1 vrf-target export target:400:2
set routing-instances VRF-mac-vrf-ep-t2-symm-transl-1-1 vrf-target import target:500:2
set routing-instances VRF-mac-vrf-ep-t2-symm-transl-1-10 instance-type vrf
set routing-instances VRF-mac-vrf-ep-t2-symm-transl-1-10 interface irb.38
set routing-instances VRF-mac-vrf-ep-t2-symm-transl-1-10 interface irb.39
set routing-instances VRF-mac-vrf-ep-t2-symm-transl-1-10 interface irb.40
set routing-instances VRF-mac-vrf-ep-t2-symm-transl-1-10 interface irb.41
set routing-instances VRF-mac-vrf-ep-t2-symm-transl-1-10 interface lo0.11
set routing-instances VRF-mac-vrf-ep-t2-symm-transl-1-10 protocols evpn ip-prefix-routes
advertise direct-nexthop
set routing-instances VRF-mac-vrf-ep-t2-symm-transl-1-10 protocols evpn ip-prefix-routes
encapsulation vxlan
set routing-instances VRF-mac-vrf-ep-t2-symm-transl-1-10 protocols evpn ip-prefix-routes vni
9100011
set routing-instances VRF-mac-vrf-ep-t2-symm-transl-1-10 protocols evpn irb-symmetric-routing
vni 9100011
set routing-instances VRF-mac-vrf-ep-t2-symm-transl-1-10 route-distinguisher 192.168.0.4:11
set routing-instances VRF-mac-vrf-ep-t2-symm-transl-1-10 routing-options multipath
set routing-instances VRF-mac-vrf-ep-t2-symm-transl-1-10 routing-options rib VRF-mac-vrf-ep-t2-
symm-transl-1-10.inet6.0 multipath
set routing-instances VRF-mac-vrf-ep-t2-symm-transl-1-10 vrf-table-label
set routing-instances VRF-mac-vrf-ep-t2-symm-transl-1-10 vrf-target export target:400:11
set routing-instances VRF-mac-vrf-ep-t2-symm-transl-1-10 vrf-target import target:500:11
set routing-instances VRF-mac-vrf-ep-t2-symm-transl-1-11 instance-type vrf
set routing-instances VRF-mac-vrf-ep-t2-symm-transl-1-11 interface irb.42
set routing-instances VRF-mac-vrf-ep-t2-symm-transl-1-11 interface irb.43
set routing-instances VRF-mac-vrf-ep-t2-symm-transl-1-11 interface irb.44
set routing-instances VRF-mac-vrf-ep-t2-symm-transl-1-11 interface irb.45
set routing-instances VRF-mac-vrf-ep-t2-symm-transl-1-11 interface lo0.12
set routing-options forwarding-table chained-composite-next-hop ingress evpn
set routing-options forwarding-table ecmp-fast-reroute
set routing-options forwarding-table export per-packet-load-balance
set routing-options protect core
set routing-options router-id 192.168.0.4
set switch-options route-distinguisher 192.168.0.4:20000
set switch-options vrf-target target:10458:2
set switch-options vtep-source-interface lo0.0
set system arp aging-timer 60
```

# Configure EVPN-VXLAN Data Center Stitching Through Interconnected EVPN-MPLS WAN Gateways

**SUMMARY**

You can seamlessly stitch Ethernet VPN Virtual Extensible LAN (EVPN-VXLAN) data centers through WAN gateway devices running EVPN-MPLS.

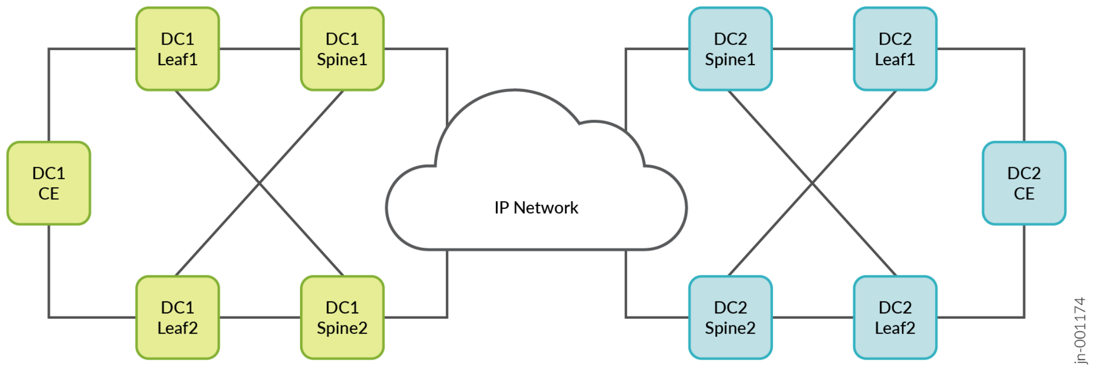
**IN THIS SECTION**

This article outlines the configuration necessary to stitch an EVPN-VXLAN data center to another EVPN-VXLAN data center, while traversing an EVPN-MPLS WAN fabric. Your WAN could be geographically dispersed or local to the same data center where the EVPN-VXLAN points of delivery (PODS) reside. See RFC 9014.

## Topology

The following diagram shows two EVPN-VXLAN data centers connected through an EVPN-MPLS WAN, using the gateway model. Each gateway is configured with an EVPN MAC-VRF routing instance. Each MAC-VRF instance uses VXLAN encapsulation, and the interconnect within each MAC-VRF instance uses MPLS encapsulation.

EVPN-VLXAN through EVPN-MPLS WAN



## Configuration

Follow the steps below to configure a pair of gateways. We'll show the relevant configuration for GW11 and GW21. Full device configurations are beyond the scope of this document.

1. Configure a MAC VRF routing instance.

```
set routing-instances instance-name instance-type mac-vrf
```

2. Configure the EVPN protocol.

```
set routing-instance instance-name protocols evpn
```

3. Configure VXLAN encapsulation and supporting elements.

```
set routing-instance instance-name protocols evpn encapsulation vxlan
set routing-instance instance-name protocols evpn default-gateway no-gateway-community
set routing-instance instance-name protocols evpn extended-vni-list all
```

4. Configure the interconnect statement and supporting elements.

```
set routing-instance instance-name protocols evpn interconnect vrf-target target:x:x
set routing-instance instance-name protocols evpn interconnect route-distiguisher rd
set routing-instance instance-name protocols evpn interconnect interconnected-vlan-list
```

```
[ vlans ]
set routing-instance instance-name protocols evpn interconnect encapsulation mpls
```

Based on the platform you're configuring, set these platform-specific options:

- (QFX5120 only) You are required to configure `set forwarding-options evpn-vxlan vxlan-trans-vni-enable` on QFX5120 Series switches to enable Layer 2 stitching. The packet forwarding engine (PFE) will restart once this configuration is applied, resulting in a restart of the associated FPC and interfaces.

- (ACX series) You are required to configure `set system packet-forwarding-options system-profile vxlan-extended` on ACX Series routers to support an IPv6 underlay. Refer to *vxlan-extended* for additional information.

- (ACX series) You are required to configure `set system packet-forwarding-options system-profile vxlan-stitching` on ACX Series routers to enable Layer 2 stitching. The PFE will restart once this configuration is applied. Refer to *vxlan-stitching* for additional details related to control-word support in EVPN-VXLAN and EVPN-MPLS environments, and further ACX requirements.

- (ACX series) You can configure the following parameters on ACX Series routers to enable load balancing based on traffic payload:

  ```
  set forwarding-options hash-key family inet layer-3
  set forwarding-options hash-key family inet layer-4
  set forwarding-options hash-key family inet6 layer-3
  set forwarding-options hash-key family inet6 layer-4
  set forwarding-options hash-key family multiservice source-mac
  set forwarding-options hash-key family multiservice destination-mac
  ```

  Refer to *hash-key (Forwarding Options)* for additional information and requirements related to ACX and PTX devices.

- (ACX series) For features such as EVPN-VXLAN, if there is a requirement for a higher Layer 2 MAC scale, then it is recommended to migrate to the “cloud-metro” profile. ACX7024 Series devices support lean-edge and cloud-metro options only. Refer to *hw-db-profile* for additional information and ACX requirements.

5. Configure the interconnect ESI and supporting elements.

```
set routing-instance instance-name protocols evpn interconnect esi esi
set routing-instance instance-name protocols evpn interconnect esi all-active
```

6. Configure additional elements of the MAC-VRF instance.

```
set routing-instance instance-name vtep-source-interface lo0.0
set routing-instance instance-name service-type vlan-aware
set routing-instance instance-name interface interface
set routing-instance instance-name route-distinguisher rd
set routing-instance instance-name vrf-target target:x:x
```

7. Configure VLANs.

```
set routing-instance instance-name instance-type mac-vrf vlans vlan-name vlan-id value
set routing-instance instance-name instance-type mac-vrf vlans vlan-name l3-interface
interface
set routing-instance instance-name instance-type mac-vrf vlans vlan-name vxlan vni vni
```

Displayed here is the configuration for GW11. Change any values to match your existing network.

```
user@device> show configuration routing-instances evpn-vxlan
instance-type mac-vrf;
protocols {
    evpn {
        encapsulation vxlan;
        default-gateway no-gateway-community;
        extended-vni-list all;
        interconnect {
            vrf-target target:2:2;
            route-distinguisher 100:110;
            esi {
                00:0a:0b:0c:0d:0a:0b:0c:0d:0a;
                all-active;
            }
            interconnected-vlan-list [ 51 52 ];
            encapsulation mpls;
        }

    }
}
vtep-source-interface lo0.0;
service-type vlan-aware;
interface et-0/0/7.0;
interface et-0/0/9.0;
```

```
route-distinguisher 100:11;
vrf-target target:1:1;
vlans {
    bd51 {
        vlan-id 51;
        l3-interface irb.51;
        vxlan {
            vni 501;
        }
    }
    bd52 {
        vlan-id 52;
        l3-interface irb.52;
        vxlan {
            vni 502;
        }
    }
}
```

Displayed here is the configuration for GW21. Change any values to match your existing network.

```
user@device> show configuration routing-instances evpn-vxlan
instance-type mac-vrf;
protocols {
    evpn {
        encapsulation vxlan;
        default-gateway no-gateway-community;
        extended-vni-list all;
        interconnect {
            vrf-target target:2:2;
            route-distinguisher 200:210;
            esi {
                00:aa:bb:cc:dd:aa:bb:cc:dd:aa;
                all-active;
            }
            interconnected-vlan-list [ 51 52 ];
            encapsulation mpls;
        }
    }
}
vtep-source-interface lo0.0;
service-type vlan-aware;
```

```
interface et-0/0/7.0;
interface et-0/0/9.0;
route-distinguisher 200:21;
vrf-target target:3:3;
vlans {
    bd51 {
        vlan-id 51;
        l3-interface irb.51;
        vxlan {
            vni 501;
        }
    }
    bd52 {
        vlan-id 52;
        l3-interface irb.52;
        vxlan {
            vni 502;
        }
    }
}
```

**NOTE**: For multihomed gateway devices, you must include the following statement at the global level:

`set protocols evpn interconnect-multihoming-peer-gateways` *`VTEP IP of each DCI-GW peer in local DC`*

The above statement can't be configured within a routing instance. Also, the statement `interconnect-multihoming-peer-gateways` is renamed in Junos OS Release 24.2R1 to `multihoming-peer-gateways`. Starting in Junos OS and Junos OS Evolved Release 24.4R1, instead we provide this *static-multihoming-peer* statement for OISM multihoming peer leaf devices. You won't see the *multihoming-peer-gateways* statement in the Junos OS CLI anymore.

A full discussion of multihoming is beyond the scope of this document. For more about multihoming, see *EVPN Multihoming Overview*.

## Verification

Confirm that routes are showing in mpls.0.

```
user@GW11> show route table mpls.0 protocol evpn | grep "Egress"
102                *[EVPN/7] 00:21:22, routing-instance evpn-vxlan, route-type Egress-MAC, vlan-
id 51, ESI 00:aa:bb:cc:dd:aa:bb:cc:dd:aa
103                *[EVPN/7] 00:21:22, remote-pe 10.200.22.22, routing-instance evpn-vxlan,
route-type Egress-MAC, vlan-id 51
104                *[EVPN/7] 00:21:22, routing-instance evpn-vxlan, route-type Egress-MAC, vlan-
id 52, ESI 00:aa:bb:cc:dd:aa:bb:cc:dd:aa
105                *[EVPN/7] 00:21:22, remote-pe 10.200.22.22, routing-instance evpn-vxlan,
route-type Egress-MAC, vlan-id 52
106                *[EVPN/7] 00:21:22, remote-pe 10.200.22.21, routing-instance evpn-vxlan,
route-type Egress-MAC, vlan-id 51
107                *[EVPN/7] 00:21:22, remote-pe 10.200.22.21, routing-instance evpn-vxlan,
route-type Egress-MAC, vlan-id 52
108                *[EVPN/7] 00:21:22, remote-pe 10.200.22.21, routing-instance evpn-vxlan,
route-type Egress-IM, vlan-id 51
109                *[EVPN/7] 00:21:22, remote-pe 10.200.22.21, routing-instance evpn-vxlan,
route-type Egress-IM, vlan-id 52
110                *[EVPN/7] 00:21:22, remote-pe 10.200.22.22, routing-instance evpn-vxlan,
route-type Egress-IM, vlan-id 51
111                *[EVPN/7] 00:21:22, remote-pe 10.200.22.22, routing-instance evpn-vxlan,
route-type Egress-IM, vlan-id 52

{master}[edit]
user@GW11> show route table mpls.0 protocol evpn | grep "Ingress"
99                 *[EVPN/7] 00:21:29, routing-instance evpn-vxlan, route-type Ingress-MAC, vlan-
id 51
                    [EVPN/7] 00:21:29, routing-instance evpn-vxlan, route-type Ingress-Aliasing,
vlan-id 51
100                *[EVPN/7] 00:21:29, routing-instance evpn-vxlan, route-type Ingress-MAC, vlan-
id 52
                    [EVPN/7] 00:21:29, routing-instance evpn-vxlan, route-type Ingress-Aliasing,
vlan-id 52
112                *[EVPN/7] 00:21:28, routing-instance evpn-vxlan, route-type Ingress-IM, vlan-
id 51
113                *[EVPN/7] 00:21:28, routing-instance evpn-vxlan, route-type Ingress-IM, vlan-
id 52
```

Confirm that VXLAN VNI's are populating in the EVPN database.

```
user@GW11> **show evpn database mac-address 00:00:11:11:51:01 extensive**
Instance: evpn-vxlan

VN Identifier: 501, MAC address: 00:00:11:11:51:01
  State: 0x0
  Source: 00:11:12:11:11:11:11:11:11:11, Rank: 1, Status: Active
    Remote origin: 10.11.1.11
    Remote state: <Mac-Only-Adv>
    Remote origin: 10.11.1.12
    Remote state: <Mac-Only-Adv>
    Mobility sequence number: 0 (minimum origin address 10.11.1.11)
    Timestamp: Jun 28 22:51:12.147619 (0x649c6c08)
    State: <Remote-To-Local-Adv-Done>
    MAC advertisement route status: Not created (no local state present)
    Interconn advertisement route status: DCI route created
    IP address: 10.100.51.1
      Remote origin: 10.11.1.11
      Remote state: <Sent-to-l2ald>
      Remote origin: 10.11.1.12
      Remote state: <Sent-to-l2ald>
      Interconn advertisement route status: DCI route created
    History db:
      Time                       Event
      Jun 28 22:51:09.533 2023   00:11:12:11:11:11:11:11:11:11 : Created
      Jun 28 22:51:09.541 2023   00:11:12:11:11:11:11:11:11:11 : Remote peer 10.11.1.12 created
      Jun 28 22:51:09.546 2023   Updating output state (change flags 0x1 <ESI-Added>)
      Jun 28 22:51:09.546 2023   Active ESI changing (not assigned ->
00:11:12:11:11:11:11:11:11:11)
      Jun 28 22:51:09.547 2023   00:11:12:11:11:11:11:11:11:11 : 10.100.51.1 Selected IRB
interface nexthop
      Jun 28 22:51:09.547 2023   00:11:12:11:11:11:11:11:11:11 : 10.100.51.1 Reject remote ip
host route 10.100.51.1 in L3 context VRF-100 since no remote-ip-host-routes configured
      Jun 28 22:51:09.733 2023   00:11:12:11:11:11:11:11:11:11 : 10.100.51.1 Selected IRB
interface nexthop
      Jun 28 22:51:09.733 2023   00:11:12:11:11:11:11:11:11:11 : 10.100.51.1 Reject remote ip
host route 10.100.51.1 in L3 context VRF-100 since no remote-ip-host-routes configured
      Jun 28 22:56:46.300 2023   00:11:12:11:11:11:11:11:11:11 : 10.100.51.1 Selected IRB
interface nexthop
```

```
      Jun 28 22:56:46.300 2023   00:11:12:11:11:11:11:11:11:11 : 10.100.51.1 Reject remote ip 
host route 10.100.51.1 in L3 context VRF-100 since no remote-ip-host-routes configured
```

Confirm MAC table entries for an IRB.

```
user@GW11> show ethernet-switching table 00:00:11:11:51:01

MAC flags (S - static MAC, D - dynamic MAC, L - locally learned, P - Persistent static
           SE - statistics enabled, NM - non configured MAC, R - remote PE MAC, O - ovsdb MAC,
           B - Blocked MAC)



Ethernet switching table : 33 entries, 33 learned
Routing instance : evpn-vxlan
   Vlan                MAC                 MAC       GBP    Logical                SVLBNH/      
Active
   name                address             flags     tag    interface              VENH Index   
source
   bd51                00:00:11:11:51:01   DR               esi.11802                           
00:11:12:11:11:11:11:11:11:11
```

```
user@GW11> show ethernet-switching mac-ip-table 00:00:11:11:51:01

MAC IP flags  (S - Static, D - Dynamic, L - Local , R - Remote, Lp - Local Proxy,
               Rp - Remote Proxy, K - Kernel, RT - Dest Route, (N)AD - (Not) Advt to remote,
               RE - Re-ARP/ND, RO - Router, OV - Override, Ur - Unresolved,
               RTS - Dest Route Skipped, RGw - Remote Gateway, GBP - Group Based Policy,
               RTF - Dest Route Forced, SC - Static Config, P - Probe, NLC - No Local Config)
 Routing instance : evpn-vxlan
 Bridging domain : bd51
   IP                            MAC                  Flags              GBP    
Logical            Active
   address                       address                                 Tag    
Interface          source
   10.100.51.1                    00:00:11:11:51:01    DR,K,RT                      
esi.11802          00:11:12:11:11:11:11:11:11:11

user@GW11> show route forwarding-table destination 00:00:11:11:51:01 vpn evpn-vxlan
Routing table: evpn-vxlan.vpls
VPLS:
Destination        Type RtRef Next hop           Type Index    NhRef Netif
```

```
00:00:11:11:51:01/48 user     0                 indr     11809      1 .local..56
                                                comp     11802      1
                                                comp     11795      1 vtep.32773
                                                indr      6323      1
                                                sftw     19002      1 et-0/0/1.0
                              10.11.11.1        ucst      1014      1 et-0/0/1.0
                                                comp     11796      1 vtep.32775
                                                indr      6324      1
                                                sftw     19004      1 et-0/0/3.0
                              10.12.11.1        ucst      1001      1 et-0/0/3.0
```

```
user@GW11> show arp no-resolve | grep 10.100.51.1
00:00:11:11:51:12 10.100.51.12     irb.51[ et-0/0/9.0 ]     permanent remote
```

## Static VXLAN Tunnels with Q-in-Q

**IN THIS SECTION**

For small MC-LAG networks, you can use static VXLAN to reduce the control plane complexity in your network. Configuring VTEPs on a static VXLAN is straightforward. Use this example to configure static VXLAN tunnels with Q-in-Q tagging (VLAN translation) between data centers. In this example, we focus on the following features:

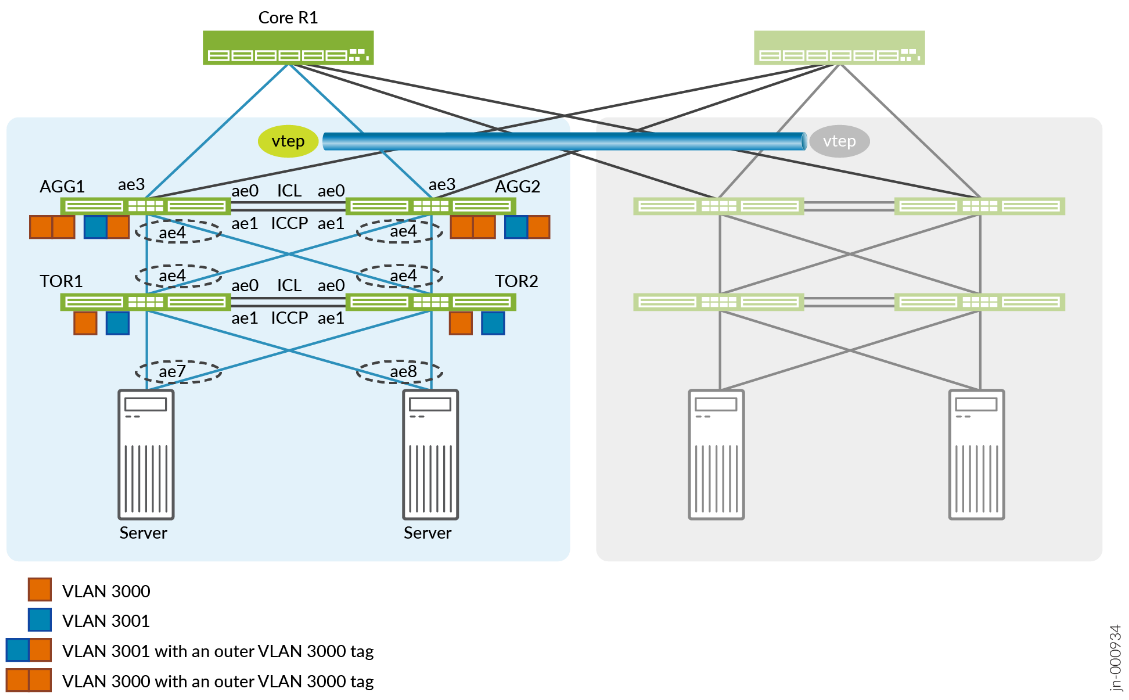
- Static VXLAN—Static VXLAN connects servers in different data centers by creating a Layer 2 path (tunnel). For more information about static VXLAN, see *Static VXLAN*.
- Q-in-Q tunnels—Q-in-Q tunnels segregates and bundles different customer VLAN (C-VLAN) traffic into a single service provider VLAN.

For more information about Q-in-Q tunnels, see *Configuring Q-in-Q Tunneling and VLAN Q-in-Q Tunneling and VLAN Translation*.

- MC-LAG—MC LAG provides redundancy and load balancing. We configure ICL and ICCP connection between two peer devices to create the MC-LAG. For more information about MC-LAG, see *Understanding Multichassis Link Aggregation Groups*

Figure 85 on page 457 shows a portion of a spine-leaf data center (POD). Within the POD, the TOR devices (TOR1 and TOR2) collect VLANs from the servers below and also manages the VLAN translations (Q-in-Q tunnels). The aggregators collect VLANs from different TOR devices and function as the gateway for the POD. We use a static VXLAN tunnel as a gateway between two PODs. We configure MC-LAG between the peer TOR devices and the peer aggregators. In our reference test environment, we tested a configuration with 64 pods. For this example, we describe how to configure the aggregators and TOR devices in a single pod.

**Figure 85: Data Center POD with Q-in-Q and Static VXLAN Tunnels**



This example is configured on top of an existing IP Fabric. See "IP Fabric Underlay Network Design and Implementation" on page 80.

## Configuring the Aggregators

The following section describes how to configure the aggregators.

1. Configure the aggregators to support aggregated Ethernet and MC-LAG.
   - Set the maximum number of aggregated Ethernet interfaces.
   - Set the service identifier (SID) for the LAG.
   - Configure a loopback address.
   - Configure a management port. We use the management interface as an "always up" port to support the keepalive communication between ICCP peers.

   AGG1 and AGG2

   ```
   set chassis aggregated-devices ethernet device-count 64
   set switch-options service-id 1
   ```

   AGG1

   ```
   set interfaces lo0 unit 0 family inet address 192.168.1.4/32 primary
   set interfaces em0 unit 0 family inet address 10.48.49.69/221
   ```

   AGG2

   ```
   set interfaces lo0 unit 0 family inet address 192.168.1.5/32 primary
   set nterfaces em0 unit 0 family inet address 10.48.49.117/22
   ```

2. Assign the aggregated Ethernet interfaces.
   - ae0 and ae1 forms the ICL and ICCP links between the aggregators.
   - ae3 connects the aggregator to the spine devices.
   - ae4 connects the aggregators to the TOR devices.

   AGG1 and AGG2

   ```
   set interfaces xe-0/0/49:0 ether-options 802.3ad ae0
   set interfaces xe-0/0/49:1 ether-options 802.3ad ae1
   set interfaces xe-0/0/48:0 ether-options 802.3ad ae3
   ```

```
set interfaces xe-0/0/50:0 ether-options 802.3ad ae4
set interfaces xe-0/0/50:1 ether-options 802.3ad ae4
```

3. Enable LACP on the aggregated Ethernet interfaces. Enable LACP with the fast periodic interval to send a packet every second.

   **AGG1 and AGG2**

```
set interfaces ae4 aggregated-ether-options minimum-links 1
set interfaces ae4 aggregated-ether-options lacp active
set interfaces ae4 aggregated-ether-options lacp periodic fast
set interfaces ae4 aggregated-ether-options lacp system-id 00:00:00:00:04:01
set interfaces ae4 aggregated-ether-options lacp admin-key 4
```

4. Configure the MC-LAG interfaces from the aggregators to the TOR devices and set it to active-active mode. Set a unique chassis ID for each peer.

   **AGG1 and AGG2**

```
set interfaces ae4 aggregated-ether-options mc-ae mc-ae-id 4
set interfaces ae4 aggregated-ether-options mc-ae redundancy-group 1
set interfaces ae4 aggregated-ether-options mc-ae mode active-active
set interfaces ae4 aggregated-ether-options mc-ae status-control active
set interfaces ae4 aggregated-ether-options mc-ae init-delay-time 300
```

   **AGG1**

```
set interfaces ae4 aggregated-ether-options mc-ae chassis-id 0
```

   **AGG2**

```
set interfaces ae4 aggregated-ether-options mc-ae chassis-id 1
```

5. Configure the ICCP peers (AGG1 and AGG2) across the ICL. We use the IP address of the management link when we configure `backup-liveness-detection` to exchange keepalive messages.

   **AGG1**

```
set interfaces ae0 description "ICCP link Connected to MCLAG peer"
set interfaces ae0 unit 0 family inet address 172.16.10.1/30
set multi-chassis multi-chassis-protection 172.16.10.2 interface ae1
set protocols iccp local-ip-addr 172.16.10.1
set protocols iccp peer 172.16.10.2 session-establishment-hold-time 600
set protocols iccp peer 172.16.10.2 redundancy-group-id-list 1
set protocols iccp peer 172.16.10.2 backup-liveness-detection backup-peer-ip 10.48.49.69
set protocols iccp peer 172.16.10.2 liveness-detection minimum-interval 1000
```

AGG2

```
set interfaces ae0 description "ICCP link Connected to MCLAG peer"
set interfaces ae0 unit 0 family inet address 172.16.10.2/30
set multi-chassis multi-chassis-protection 172.16.10.1 interface ae1
set protocols iccp local-ip-addr 172.16.10.2
set protocols iccp peer 172.16.10.1 session-establishment-hold-time 600
set protocols iccp peer 172.16.10.1 redundancy-group-id-list 1
set protocols iccp peer 172.16.10.1 backup-liveness-detection backup-peer-ip 10.48.49.117
set protocols iccp peer 172.16.10.1 liveness-detection minimum-interval 1000
```

6. Configure the interfaces to support VLANs.

   AGG1 and AGG2

```
set vlans SP-VLAN-3000 vlan-id 3000
set vlans SP-VLAN-3000 interface ae1.3000
set vlans SP-VLAN-3000 interface ae4.3000
set vlans SP-VLAN-3001 vlan-id 3001
set vlans SP-VLAN-3001 interface ae1.3001
set vlans SP-VLAN-3001 interface ae4.3001

set interfaces ae1 unit 3000 encapsulation vlan-bridge
set interfaces ae1 unit 3000 vlan-id 3000
set interfaces ae1 unit 3001 encapsulation vlan-bridge
set interfaces ae1 unit 3001 vlan-id 3001

set interfaces ae4 description "Connected to TOR1 TOR2"
set interfaces ae4 flexible-vlan-tagging
set interfaces ae4 encapsulation flexible-ethernet-services

set interfaces ae4 unit 3000 encapsulation vlan-bridge
set interfaces ae4 unit 3000 vlan-id 3000
set interfaces ae4 unit 3001 encapsulation vlan-bridge
set interfaces ae4 unit 3001 vlan-id 3001
```

7. Configure the interface to the Spine devices.

AGG1

```
set interfaces ae3 description "Connected to Spine-1"
set interfaces ae3 unit 0 family inet address 192.168.100.2/24
```

AGG2

```
set interfaces ae3 description "Connected to Spine-1"
set interfaces ae3 unit 0 family inet address 192.168.200.2/24
```

8. Enable static VXLAN by configuring the local and remote VTEP interfaces.

   AGG1 and AGG2

```
set switch-options vtep-source-interface lo0.0
set switch-options remote-vtep-list 192.168.1.6
```

9. Map the VLANs to the remote VTEP.

   AGG1 and AGG2

```
set vlans SP-VLAN-3000 vxlan vni 103000
set vlans SP-VLAN-3000 vxlan ingress-node-replication
set vlans SP-VLAN-3000 vxlan static-remote-vtep-list 192.168.1.6
set vlans SP-VLAN-3001 vxlan vni 103001
set vlans SP-VLAN-3001 vxlan ingress-node-replication
set vlans SP-VLAN-3001 vxlan static-remote-vtep-list 192.168.1.6
```

## Configuring the TOR Devices

The following section describes how to configure the TOR devices.

1. Configure the TOR device to support aggregated Ethernet and MC-LAG.

   - Set the maximum number of aggregated Ethernet interfaces.
   - Set the SID for the LAG.
   - Configure a loopback address.

- Configure a management port. We use the management interface as an "always up" port to support the keepalive communication between ICCP peers.

TOR1 and TOR2

```
set chassis aggregated-devices ethernet device-count 64
set switch-options service-id 1
```

TOR1

```
set interfaces lo0 unit 0 family inet address 192.168.1.8/32 primary
set interfaces em0 unit 0 family inet address 10.48.49.197/22
```

TOR2

```
set interfaces lo0 unit 0 family inet address 192.168.1.9/32 primary
set interfaces em0 unit 0 family inet address 10.48.49.196/22
```

2. Assign the aggregated Ethernet interfaces.

   - ae0 and ae1 form the ICL and ICCP link between the TOR devices.
   - ae4 connects the TOR devices to the aggregators.
   - ae7 and ae8 connect the TOR devices to the servers.

   TOR1 and TOR2

```
set interfaces xe-0/0/0:0 ether-options 802.3ad ae0
set interfaces xe-0/0/0:1 ether-options 802.3ad ae0
set interfaces xe-0/0/0:2 ether-options 802.3ad ae1
set interfaces xe-0/0/0:3 ether-options 802.3ad ae1
set interfaces xe-0/0/1:0 ether-options 802.3ad ae4
set interfaces xe-0/0/1:1 ether-options 802.3ad ae4
set interfaces xe-0/0/2:1 ether-options 802.3ad ae7
set interfaces xe-0/0/2:3 ether-options 802.3ad ae8
```

3. Enable LACP on the aggregated Ethernet interfaces. Enable LACP with a fast periodic interval to send a packet every second.

TOR1 and TOR2

```
set interfaces ae4 aggregated-ether-options minimum-links 1
set interfaces ae4 aggregated-ether-options lacp active
set interfaces ae4 aggregated-ether-options lacp periodic fast
set interfaces ae4 aggregated-ether-options lacp system-id 00:00:00:00:04:02
set interfaces ae4 aggregated-ether-options lacp admin-key 4

set interfaces ae7 aggregated-ether-options minimum-links 1
set interfaces ae7 aggregated-ether-options lacp active
set interfaces ae7 aggregated-ether-options lacp periodic fast
set interfaces ae7 aggregated-ether-options lacp system-id 00:00:00:00:07:01
set interfaces ae7 aggregated-ether-options lacp admin-key 7

set interfaces ae8 aggregated-ether-options minimum-links 1
set interfaces ae8 aggregated-ether-options lacp active
set interfaces ae8 aggregated-ether-options lacp periodic fast
set interfaces ae8 aggregated-ether-options lacp system-id 00:00:00:00:08:01
set interfaces ae8 aggregated-ether-options lacp admin-key 8
```

4. Configure the interfaces to support VLANs and Q-in-Q translation.

**NOTE**: When you configure Q-in-Q mapping, the device selects the lowest value in the VLAN ID range as the outer tag. For example, with a range of 3000-3001 in our vlan-id-list, our device uses the VLAN 3000 as the outer tag. When the device receives an outgoing packet with a VLAN in the 3000 to 3001 range, the device pushes an outer tag of with a VLAN ID of 3000. Conversely, the device strips the outer tag for incoming packets with a VLAN ID of 3000 in its outer tag.

TOR1 and TOR2

```
set interfaces ae1 description "ICL link Connected to MCLAG peer"
set interfaces ae1 flexible-vlan-tagging
set interfaces ae1 encapsulation encapsulation extended-vlan-bridge
set interfaces ae1 unit 3000 vlan-id 3000

set interfaces ae4 description "Connected to AGG1 AGG2"
set interfaces ae4 flexible-vlan-tagging
set interfaces ae4 encapsulation extended-vlan-bridge
set interfaces ae4 unit 3000 vlan-id-list 3000-3001
set interfaces ae4 unit 3000 input-vlan-map push
set interfaces ae4 unit 3000 output-vlan-map pop

set interfaces ae7 description "Connected to Server1"
set interfaces ae7 flexible-vlan-tagging
set interfaces ae7 encapsulation extended-vlan-bridge
set interfaces ae7 unit 3000 vlan-id-list 3000-3001
set interfaces ae7 unit 3000 input-vlan-map push
set interfaces ae7 unit 3000 output-vlan-map pop

set interfaces ae8 description "Connected to Server2"
set interfaces ae8 flexible-vlan-tagging
set interfaces ae8 encapsulation extended-vlan-bridge
set interfaces ae8 unit 3000 vlan-id-list 3000-3001
set interfaces ae8 unit 3000 input-vlan-map push
set interfaces ae8 unit 3000 output-vlan-map pop

set vlans SP-VLAN-3000 interface ae1.3000
set vlans SP-VLAN-3000 interface ae4.3000
set vlans SP-VLAN-3000 interface ae7.3000
set vlans SP-VLAN-3000 interface ae8.3000
set vlans SP-VLAN-3000 service-id 3000
set vlans SP-VLAN-3001 interface ae4.3001
set vlans SP-VLAN-3001 interface ae7.3001
set vlans SP-VLAN-3001 interface ae8.3001
set vlans SP-VLAN-3001 service-id 3001
```

5. Configure the MC-LAG interfaces from the TOR devices to the aggregators and servers and set it to active-active mode. Set a unique chassis ID for each peer.

   TOR1 and TOR2

```
set interfaces ae4 aggregated-ether-options mc-ae mc-ae-id 4
set interfaces ae4 aggregated-ether-options mc-ae redundancy-group 1
set interfaces ae4 aggregated-ether-options mc-ae mode active-active
set interfaces ae4 aggregated-ether-options mc-ae status-control active
set interfaces ae4 aggregated-ether-options mc-ae init-delay-time 300

set interfaces ae7 aggregated-ether-options mc-ae mc-ae-id 7
set interfaces ae7 aggregated-ether-options mc-ae redundancy-group 1
set interfaces ae7 aggregated-ether-options mc-ae mode active-active
set interfaces ae7 aggregated-ether-options mc-ae status-control active
set interfaces ae7 aggregated-ether-options mc-ae init-delay-time 300

set interfaces ae8 aggregated-ether-options mc-ae mc-ae-id 8
set interfaces ae8 aggregated-ether-options mc-ae redundancy-group 1
set interfaces ae8 aggregated-ether-options mc-ae mode active-active
set interfaces ae8 aggregated-ether-options mc-ae status-control active
set interfaces ae8 aggregated-ether-options mc-ae init-delay-time 300
```

TOR1

```
set interfaces ae4 aggregated-ether-options mc-ae chassis-id 0
set interfaces ae7 aggregated-ether-options mc-ae chassis-id 0
set interfaces ae8 aggregated-ether-options mc-ae chassis-id 0
```

TOR2

```
set interfaces ae4 aggregated-ether-options mc-ae chassis-id 1
set interfaces ae7 aggregated-ether-options mc-ae chassis-id 1
set interfaces ae8 aggregated-ether-options mc-ae chassis-id 1
```

6. Configure ICCP across the ICL between the two TOR peers (TOR1 and TOR2). We use the IP address of the management link when we configure `backup-liveness-detection` to exchange keepalive messages.

   TOR1

```
set interfaces ae0 description "ICCP link Connected to MCLAG peer"
set interfaces ae0 unit 0 family inet address 172.16.20.1/30
set multi-chassis multi-chassis-protection 172.16.20.2 interface ae1
set protocols iccp local-ip-addr 172.16.2.1
set protocols iccp peer 172.16.20.2 session-establishment-hold-time 600
set protocols iccp peer 172.16.20.2 redundancy-group-id-list 1
set protocols iccp peer 172.16.20.2 backup-liveness-detection backup-peer-ip 10.48.49.197
set protocols iccp peer 172.16.20.2 liveness-detection minimum-interval 1000
```

   TOR2

```
set interfaces ae0 description "ICCP link Connected to MCLAG peer"
set interfaces ae0 unit 0 family inet address 172.16.20.2/30
set multi-chassis multi-chassis-protection 172.16.20.1 interface ae1
set protocols iccp local-ip-addr 172.16.20.2
set protocols iccp peer 172.16.20.1 session-establishment-hold-time 600
set protocols iccp peer 172.16.20.1 redundancy-group-id-list 1
set protocols iccp peer 17.16.20.1 backup-liveness-detection backup-peer-ip 10.48.49.196
set protocols iccp peer 172.16.20.1 liveness-detection minimum-interval 1000
```

## Verify the Static VXLAN Tunnels with Q-in-Q on Aggregators

This section shows how to verify the operation of the aggregators as it manages VLANs through the static VXLAN tunnels. All commands are issued on AGG1.

1. Display the VLAN information.

```
user@agg1> show vlans
default-switch          SP-VLAN-3000          NA
                                                           ae1.3000*
                                                           ae4.3000*
                                                           vtep.32769*
```

2. Verify the operational status of the multichassis aggregated Ethernet link.

```
user@agg1> show interfaces mc-ae
 Member Link                    : ae4
 Current State Machine's State: mcae active state
 Configuration Error Status     : No Error
 Local Status                   : active
 Local State                    : up
 Peer Status                    : active
 Peer State                     : up
     Logical Interface          : ae4.3000
     Topology Type              : bridge
     Local State                : up
     Peer State                 : up
     Peer Ip/MCP/State          : 172.16.10.2  ae1.3000 up
```

3. Verify the MC-LAG status between AGG1 and AGG2.

```
user@agg1> show iccp
```

```
Redundancy Group Information for peer 172.16.10.2
  TCP Connection       : Established
  Liveliness Detection : Up
  Backup liveness peer status: Up
  Redundancy Group ID          Status
    1                           Up

Client Application: lacpd
  Redundancy Group IDs Joined: 1

Client Application: l2ald_iccpd_client
  Redundancy Group IDs Joined: 1
```

4. Verify the LACP status on the aggregated Ethernet interface.

```
user@agg1> show lacp interfaces

Aggregated interface: ae4
    LACP state:           Role   Exp   Def  Dist  Col  Syn  Aggr  Timeout  Activity
      xe-0/0/50:0        Actor    No    No   Yes  Yes  Yes   Yes     Fast    Active
      xe-0/0/50:0      Partner    No    No   Yes  Yes  Yes   Yes     Fast    Active
      xe-0/0/50:1        Actor    No    No   Yes  Yes  Yes   Yes     Fast    Active
      xe-0/0/50:1      Partner    No    No   Yes  Yes  Yes   Yes     Fast    Active
    LACP protocol:        Receive State  Transmit State          Mux State
      xe-0/0/50:0               Current   Fast periodic Collecting distributing
      xe-0/0/50:1               Current   Fast periodic Collecting distributing
```

5. Verify the local and remote VTEP interfaces for the Static VXLAN is operational.

```
user@agg1> show ethernet-switching vxlan-tunnel-end-point source
Logical System Name       Id  SVTEP-IP         IFL   L3-Idx    SVTEP-Mode    ELP-SVTEP-IP
<default>                 0   192.168.1.4      lo0.0    0
    L2-RTT                   Bridge Domain              VNID     Translation-VNID    MC-Group-
IP    Interface
    default-switch           SP-VLAN-3000               103000
0.0.0.0             vtep.32768




user@agg1> show ethernet-switching vxlan-tunnel-end-point remote
Logical System Name       Id  SVTEP-IP         IFL   L3-Idx    SVTEP-Mode    ELP-SVTEP-IP
<default>                 0   192.168.1.4      lo0.0    0
```

```
 RVTEP-IP         L2-RTT                    IFL-Idx    Interface     NH-Id    RVTEP-Mode   ELP-
IP         Flags
 192.168.1.5      default-switch            825        vtep.32769    1784     RNVE
    VNID          MC-Group-IP
    103000        0.0.0.0
```

## Verify Q-in-Q Tunnels on the TOR Devices

This section shows how you verify the operation of VLANs on one of the TOR Device. All commands are issued on TOR1

1. Display the VLAN information.

```
user@tor1> show vlans
default-switch          SP-VLAN-3000          NA
                                                           ae1.3000*
                                                           ae4.3000*
                                                           ae7.3000*
                                                           ae8.3000*
```

2. Verify the operational status of the multichassis aggregated Ethernet link.

```
user@tor1> show interfaces mc-ae
 Member Link                  : ae4
 Current State Machine's State: mcae active state
 Configuration Error Status   : No Error
 Local Status                 : active
 Local State                  : up
 Peer Status                  : active
 Peer State                   : up
     Logical Interface        : ae4.3000
     Topology Type            : bridge
     Local State              : up
     Peer State               : up
     Peer Ip/MCP/State        : 172.16.20.2  ae1.3000 up

Member Link                  : ae7
```

```
 Current State Machine's State: mcae active state
 Configuration Error Status   : No Error
 Local Status                 : active
Peer State                   : up
     Logical Interface        : ae7.3000
     Topology Type            : bridge
     Local State              : up
     Peer State               : up
     Peer Ip/MCP/State        : 172.16.20.2  ae1.3000 up

 Member Link                  : ae8
 Current State Machine's State: mcae active state
 Configuration Error Status   : No Error
 Local Status                 : active
 Local State                  : up
 Peer Status                  : active
 Peer State                   : up
     Logical Interface        : ae8.3000
     Topology Type            : bridge
     Local State              : up
     Peer State               : up
     Peer Ip/MCP/State        : 172.16.20.1 ae1.3000 up
```

3. Verify the LACP status on the aggregated Ethernet interface.

```
user@tor1> show lacp interfaces
Aggregated interface: ae4
    LACP state:           Role   Exp   Def  Dist  Col  Syn  Aggr  Timeout  Activity
      xe-0/0/1:0         Actor    No    No   Yes  Yes  Yes   Yes     Fast    Active
      xe-0/0/1:0       Partner    No    No   Yes  Yes  Yes   Yes     Fast    Active
      xe-0/0/1:1         Actor    No    No   Yes  Yes  Yes   Yes     Fast    Active
      xe-0/0/1:1       Partner    No    No   Yes  Yes  Yes   Yes     Fast    Active
    LACP protocol:        Receive State  Transmit State          Mux State
      xe-0/0/1:0                Current   Fast periodic Collecting distributing
      xe-0/0/1:1                Current   Fast periodic Collecting distributing

Aggregated interface: ae7
    LACP state:           Role   Exp   Def  Dist  Col  Syn  Aggr  Timeout  Activity
      xe-0/0/2:1         Actor    No    No   Yes  Yes  Yes   Yes     Fast    Active
      xe-0/0/2:1       Partner    No    No   Yes  Yes  Yes   Yes     Fast    Active
    LACP protocol:        Receive State  Transmit State          Mux State
      xe-0/0/2:1                Current   Fast periodic Collecting distributing
```

```
Aggregated interface: ae8
    LACP state:           Role   Exp   Def  Dist  Col  Syn  Aggr  Timeout  Activity
      xe-0/0/2:3         Actor    No    No   Yes  Yes  Yes   Yes     Fast    Active
      xe-0/0/2:3       Partner    No    No   Yes  Yes  Yes   Yes     Fast    Active
    LACP protocol:        Receive State  Transmit State          Mux State
      xe-0/0/2:3                Current   Fast periodic Collecting distributing
```

4. Verify the MC-LAG status between TOR1 and TOR2.

```
user@tor1> show iccp

Redundancy Group Information for peer 172.16.20.2
  TCP Connection       : Established
  Liveliness Detection : Up
  Backup liveness peer status: Up
  Redundancy Group ID          Status
    1                           Up

Client Application: l2ald_iccpd_client
  Redundancy Group IDs Joined: 1

Client Application: lacpd
  Redundancy Group IDs Joined: 1
```

# Service Chaining Design and Implementation

**IN THIS SECTION**

For an overview of service chaining, see the Service Chaining section in "Data Center Fabric Blueprint Architecture Components" on page 9.

The following sections show how to implement service chaining and service chaining of multicast in a EVPN VXLAN network.

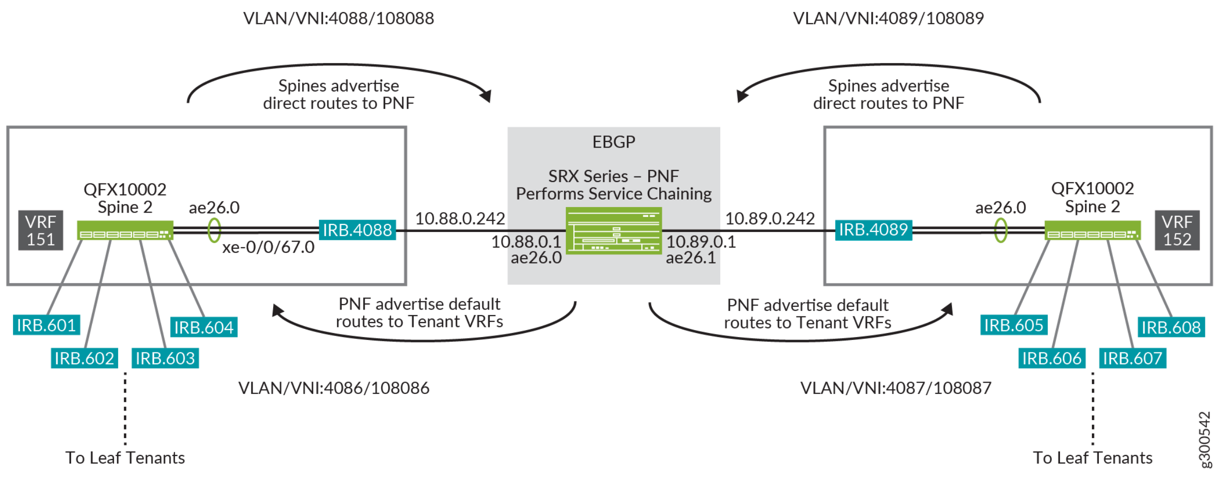
## Service Chaining

**IN THIS SECTION**

### Service Chaining Design

Figure 86 on page 471 shows a logical view of the service chaining configuration. It shows the configuration of one spine with a left and right side VRF configuration. The SRX Series router is the PNF, and is performing the Service chaining.

**Figure 86: Service Chaining Logical View**



The flow of traffic for the spine:

1. Traffic from the leaf tenants enters tenant VRF-151, and is destined to a network attached to tenant VRF-152.

2. Because there is no route from VRF-151 to VRF-152, a default route lookup is performed using routes received from the PNF.

3. After it receives the default route from the PNF, VRF-151 routes traffic toward the PNF via IRB.4088.

4. Traffic reaches the PNF, and the PNF performs the service chaining. Traffic is forwarded out ae26.1 using routes received from EBGP.

5. Traffic now reaches tenant VRF-152, and follows regular route lookup and forwarding action towards the leaf tenants.

## Configuring Service Chaining

This section shows how to configure the spine for service chaining as shown in Figure 86 on page 471. This configuration is based on the "Centrally-Routed Bridging Overlay Design and Implementation" on page 142 configuration.

1. Configure the ESI connection towards the PNF.

```
set interfaces xe-0/0/67:0 gigether-options 802.3ad ae26
set interfaces ae26 esi 00:88:88:88:88:88:88:88:88:88
set interfaces ae26 esi all-active
set interfaces ae26 aggregated-ether-options lacp active
set interfaces ae26 aggregated-ether-options lacp periodic fast
set interfaces ae26 aggregated-ether-options lacp system-id 00:00:00:00:00:22
set interfaces ae26 unit 0 family ethernet-switching interface-mode trunk
```

2. Configure the Layer 3 connection for VRF-151 towards the PNF.

```
set interfaces irb unit 4088 virtual-gateway-accept-data
set interfaces irb unit 4088 family inet address 10.88.0.242/24 preferred
set interfaces irb unit 4088 family inet address 10.88.0.242/24 virtual-gateway-address
10.88.0.254
set interfaces irb unit 4088 family inet6 address 2001:db8::10:58:0:242/112 preferred
set interfaces irb unit 4088 family inet6 address 2001:db8::10:58:0:242/112 virtual-gateway-
address 2001:db8::10:58:0:254
set interfaces irb unit 4088 virtual-gateway-v4-mac 00:00:5e:00:00:04
set interfaces ae26 unit 0 family ethernet-switching vlan members 4088
set vlans 4088 vlan-id 4088
set vlans 4088 l3-interface irb.4088
set vlans 4088 vxlan vni 104088
```

3. Configure the Layer 3 connection for VRF-152 towards the PNF.

```
set interfaces irb unit 4089 virtual-gateway-accept-data
set interfaces irb unit 4089 family inet address 10.89.0.242/24 preferred
set interfaces irb unit 4089 family inet address 10.89.0.242/24 virtual-gateway-address
10.89.0.254
set interfaces irb unit 4089 family inet6 address 2001:db8::10:59:0:242/112 preferred
set interfaces irb unit 4089 family inet6 address 2001:db8::10:59:0:242/112 virtual-gateway-
address 2001:db8::10:59:0:254
set interfaces irb unit 4089 virtual-gateway-v4-mac 00:00:5e:00:00:04
set interfaces ae26 unit 0 family ethernet-switching vlan members 4089
set vlans 4089 vlan-id 4089
set vlans 4089 l3-interface irb.4089
set vlans 4089 vxlan vni 104089
```

4. Configure a policy to export local routes from VRF-151 and VRF-152 to the PNF.

```
set policy-options policy-statement DIRECT-EXPORT-TO-PNF term 10 from protocol direct
set policy-options policy-statement DIRECT-EXPORT-TO-PNF term 10 then accept
set policy-options policy-statement DIRECT-EXPORT-TO-PNF term 20 from protocol bgp
set policy-options policy-statement DIRECT-EXPORT-TO-PNF term 20 then accept
set policy-options policy-statement DIRECT-EXPORT-TO-PNF term 30 then reject
```

5. Configure the VRF-151 routing instance.

```
set routing-instances VRF-151 instance-type vrf
set routing-instances VRF-151 interface irb.601
set routing-instances VRF-151 interface irb.602
set routing-instances VRF-151 interface irb.603
set routing-instances VRF-151 interface irb.604
set routing-instances VRF-151 interface lo0.151
set routing-instances VRF-151 interface irb.4088
set routing-instances VRF-151 route-distinguisher 192.186.0.2:151
set routing-instances VRF-151 vrf-target target:100:151
set routing-instances VRF-151 protocols bgp group to-srx type external
set routing-instances VRF-151 protocols bgp group to-srx export DIRECT-EXPORT-TO-PNF
set routing-instances VRF-151 protocols bgp group to-srx local-as 4200000002
set routing-instances VRF-151 protocols bgp group to-srx neighbor 10.88.0.1 family inet
unicast
set routing-instances VRF-151 protocols bgp group to-srx neighbor 10.88.0.1 peer-as 4000000001
set routing-instances VRF-151 protocols bgp group to-srx neighbor 2001:db8::10:58:0:1 family
```

```
inet6 unicast
set routing-instances VRF-151 protocols bgp group to-srx neighbor 2001:db8::10:58:0:1 peer-as
4000000001
```

6. Configure the VRF-152 routing instance.

```
set routing-instances VRF-152 instance-type vrf
set routing-instances VRF-152 interface irb.605
set routing-instances VRF-152 interface irb.606
set routing-instances VRF-152 interface irb.607
set routing-instances VRF-152 interface irb.608
set routing-instances VRF-152 interface lo0.152
set routing-instances VRF-152 interface irb.4089
set routing-instances VRF-152 route-distinguisher 192.186.0.2:152
set routing-instances VRF-152 vrf-target target:100:152
set routing-instances VRF-152 protocols bgp group to-srx type external
set routing-instances VRF-152 protocols bgp group to-srx export DIRECT-EXPORT-TO-PNF
set routing-instances VRF-152 protocols bgp group to-srx local-as 4200000002
set routing-instances VRF-152 protocols bgp group to-srx neighbor 10.89.0.1 family inet
unicast
set routing-instances VRF-152 protocols bgp group to-srx neighbor 10.89.0.1 peer-as 4000000001
set routing-instances VRF-152 protocols bgp group to-srx neighbor 2001:db8::10:59:0:1 family
inet6 unicast
set routing-instances VRF-152 protocols bgp group to-srx neighbor 2001:db8::10:59:0:1 peer-as
4000000001
```

7. Repeat this procedure for every spine in your topology.

## Verifying Service Chaining

To verify that service chaining is working:

1. On the spine, display the local routes in VRF-152.

```
host@SPINE-2> show route table VRF-152.inet.0 protocol direct
VRF-152.inet.0: 61 destinations, 62 routes (61 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

10.2.93.0/24       *[Direct/0] 1d 15:30:29
                    >  via irb.605
10.2.94.0/24       *[Direct/0] 1d 15:30:29
                    >  via irb.606
10.2.95.0/24       *[Direct/0] 1d 15:30:29
```

```
                    >  via irb.607
10.2.96.0/24       *[Direct/0] 1d 15:30:29
                    >  via irb.608
10.87.0.0/24       *[Direct/0] 1d 15:30:37
                    >  via irb.4087
10.89.0.0/24       *[Direct/0] 1d 15:30:37
                    >  via irb.4089
```

2. On the spine, display the local routes in VRF-151.

```
host@SPINE2> **show route table VRF-151.inet.0 protocol direct**
VRF-151.inet.0: 63 destinations, 64 routes (63 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

10.2.89.0/24       *[Direct/0] 1d 15:30:36
                    >  via irb.601
10.2.90.0/24       *[Direct/0] 1d 15:30:36
                    >  via irb.602
10.2.91.0/24       *[Direct/0] 1d 15:30:36
                    >  via irb.603
10.2.92.0/24       *[Direct/0] 1d 15:30:36
                    >  via irb.604
10.86.0.0/24       *[Direct/0] 1d 15:30:44
                    >  via irb.4086
10.88.0.0/24       *[Direct/0] 1d 15:30:44
                    >  via irb.4088
```

3. On the spine, check that a route lookup for VRF-151 points to the default route received from PNF.

```
host@SPINE2> **show route table VRF-151.inet.0 10.2.93.0**
VRF-151.inet.0: 63 destinations, 64 routes (63 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

0.0.0.0/0          *[BGP/170] 1d 15:33:08, localpref 100
                      AS path: 4000000001 I, validation-state: unverified
                    >  to 10.88.0.1 via irb.4088
```

4. On the PNF device, check the route lookup toward the tenant VRF-152 destination.

```
host@PNF> show route 10.2.93.0
inet.0: 20 destinations, 52 routes (20 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

10.2.93.0/24       *[BGP/170] 2d 00:39:44, localpref 100
                      AS path: 4200000002 I, validation-state: unverified
                    > to 10.89.0.242 via ae26.1
```

5. On the PNF device, check that it is advertising only default routes to the EBGP peer on tenant VRF-152.

```
host@PNF> show route advertising-protocol bgp 10.89.0.242
inet.0: 20 destinations, 52 routes (20 active, 0 holddown, 0 hidden)
  Prefix                  Nexthop              MED     Lclpref    AS path
* 0.0.0.0/0               Self                                    I
```

6. On the PNF device, check that it is advertising only default routes to the EBGP peer on tenant VRF-151.

```
host@PNF> show route advertising-protocol bgp 10.88.0.242
inet.0: 20 destinations, 52 routes (20 active, 0 holddown, 0 hidden)
  Prefix                  Nexthop              MED     Lclpref    AS path
* 0.0.0.0/0               Self                                    I
```

7. On the spine, verify that the spine is advertising local VRF-152 routes to the PNF.

```
host@SPINE2> show route advertising-protocol bgp 10.89.0.1
VRF-152.inet.0: 52 destinations, 53 routes (52 active, 0 holddown, 0 hidden)
  Prefix                  Nexthop              MED     Lclpref    AS path
* 10.2.93.0/24            Self                                    I
* 10.2.94.0/24            Self                                    I
* 10.2.95.0/24            Self                                    I
* 10.2.96.0/24            Self                                    I
* 10.87.0.0/24            Self                                    I
* 10.89.0.0/24            Self                                    I
* 192.68.152.1/32         Self                                    4100000000 I
```

8. On the spine, verify that the spine is advertising local VRF-151 routes to the PNF.

```
host@SPINE2> show route advertising-protocol bgp 10.88.0.1
VRF-151.inet.0: 53 destinations, 54 routes (53 active, 0 holddown, 0 hidden)
  Prefix                  Nexthop              MED     Lclpref    AS path
* 10.2.89.0/24            Self                                    I
* 10.2.90.0/24            Self                                    I
* 10.2.91.0/24            Self                                    I
* 10.2.92.0/24            Self                                    I
* 10.86.0.0/24            Self                                    I
* 10.88.0.0/24            Self                                    I
* 192.68.151.1/32         Self                                    4100000000 I
```

## Service Chaining with Multicast

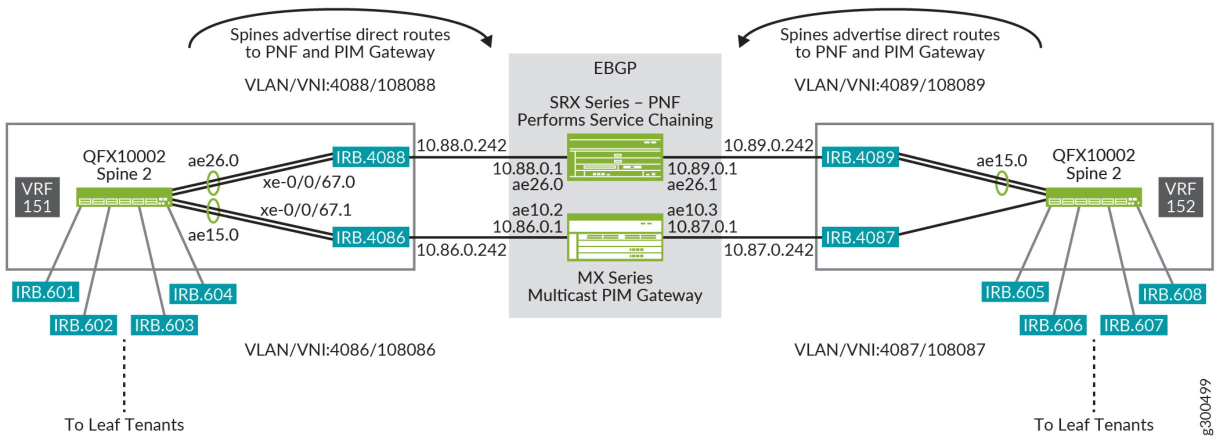
**IN THIS SECTION**

### Service Chaining with Multicast Design

Figure 87 on page 478 shows the logical view of the service chaining configuration. The VRF routing instances and IRB interfaces are all configured on the same spine.

In this design, we are performing service chaining with an SRX Series router as the PNF and with a PIM gateway.

**Figure 87: Service Chaining Logical View**



The flow of traffic for Spine 2 when multicast receivers start sending IGMP reports:

1. The leaf devices snoop the IGMP report and advertise EVPN Type 6 routes to the spines to notify the spines of the interested Multicast receiver.

2. The spine receives the EVPN Type 6 routes based on the VNI mapping, and creates a PIM join (*.G) entry in VRF-152.

   The spine configuration includes the address of the RP on the PIM gateway. However, on the VRF-152 routing instance, there is only a default route toward the RP via the PNF.

3. The PIM designated router (DR) on the receiver side IRBs (irb.605, irb.606 irb.607, irb.608) sends a PIM join (*.G ) towards the PNF device on irb.4089.

4. The PNF device, creates a PIM (*.G) entry with ae26.1 as the outgoing interface.

   The PNF is configured with the RP.

   On the PNF, the RP is also configured on the PNF and the lookup to the PIM RP points toward interface ae26.0 -> interface towards VRF-151 on spines

5. The PIM join arrives on VRF-151 on irb.4088 and creates a PIM ( *,G ) state with irb.4088 as the outgoing interface and lookup towards the RP points to irb.4086.

6. The Spine sends the PIM join entry on irb.4086 towards the PIM gateway.

7. The PIM gateway receives the PIM (*,G ) join on ae10.2.

The flow of traffic for Spine 2 when the multicast source on VRF-151 starts sending packets:

1. VRF-151 creates a PIM (*,G ) entry with irb.4088 as the outgoing interface and also the RP reachable via irb.4086.

2. Spine 2 sends two packets—one toward the PNF on irb.4088 and one toward the PIM gateway on irb.4086.

   When the PNF receives the packet:

   a. The PNF forwards it based on the PIM ( *.G ) entry it created with outgoing interface ae26.1

   b. Multicast traffic arrives on irb.4089 in VRF-512 and is forwarded to the receivers on the leafs.

   When the PIM gateway receives the packet, it forwards the packet based on its (*,G) PIM entry with the outgoing interface as ae10.2.

3. When traffic arrives at irb.4086 from the PIM gateway on Spine 2, Spine 2 prunes the RPT/shared tree.

## Configuring Service Chaining With Multicast

This section shows how to configure the spine for service chaining as shown in Figure 87 on page 478.

1. Configure the ESI connection towards the PNF.

```
set interfaces xe-0/0/67:0 gigether-options 802.3ad ae26
set interfaces ae26 esi 00:88:88:88:88:88:88:88:88:88
set interfaces ae26 esi all-active
set interfaces ae26 aggregated-ether-options lacp active
set interfaces ae26 aggregated-ether-options lacp periodic fast
set interfaces ae26 aggregated-ether-options lacp system-id 00:00:00:00:00:22
set interfaces ae26 unit 0 family ethernet-switching interface-mode trunk
```

2. Configure the Layer 3 connection for VRF-151 towards the PNF.

```
set interfaces irb unit 4088 virtual-gateway-accept-data
set interfaces irb unit 4088 family inet address 10.88.0.242/24 preferred
set interfaces irb unit 4088 family inet address 10.88.0.242/24 virtual-gateway-address
10.88.0.254
set interfaces irb unit 4088 family inet6 address 2001:db8::40:58:0:242/112 preferred
set interfaces irb unit 4088 family inet6 address 2001:db8::40:58:0:242/112 virtual-gateway-
address 2001:db8::40:58:0:254
set interfaces irb unit 4088 virtual-gateway-v4-mac 00:00:5e:00:00:04
set interfaces ae26 unit 0 family ethernet-switching vlan members 4088
set vlans 4088 vlan-id 4088
set vlans 4088 l3-interface irb.4088
set vlans 4088 vxlan vni 104088
```

3. Configure the Layer 3 connection for VRF-152 towards the PNF.

```
set interfaces irb unit 4089 proxy-macip-advertisement
set interfaces irb unit 4089 virtual-gateway-accept-data
set interfaces irb unit 4089 family inet address 10.89.0.242/24 preferred
set interfaces irb unit 4089 family inet address 10.89.0.242/24 virtual-gateway-address
10.89.0.254
set interfaces irb unit 4089 family inet6 address 2001:db8::10:59:0:242/112 preferred
set interfaces irb unit 4089 family inet6 address 2001:db8::10:59:0:242/112 virtual-gateway-
address 2001:db8::10:59:0:254
set interfaces irb unit 4089 virtual-gateway-v4-mac 00:00:5e:00:00:04
set interfaces ae26 unit 0 family ethernet-switching vlan members 4089
set vlans 4089 vlan-id 4089
set vlans 4089 l3-interface irb.4089
set vlans 4089 vxlan vni 104089
```

4. Configure the ESI connection towards the PIM gateway.

```
set interfaces xe-0/0/67:1 gigether-options 802.3ad ae15
set interfaces ae15 esi 00:22:22:22:22:22:22:22:22:22
set interfaces ae15 esi all-active
set interfaces ae15 aggregated-ether-options lacp active
set interfaces ae15 aggregated-ether-options lacp periodic fast
set interfaces ae15 aggregated-ether-options lacp system-id 00:00:00:00:00:22
set interfaces ae15 unit 0 family ethernet-switching interface-mode trunk
```

5. Configure the Layer 3 link for VRF-151 towards PIM gateway.

```
set interfaces irb unit 4086 proxy-macip-advertisement
set interfaces irb unit 4086 virtual-gateway-accept-data
set interfaces irb unit 4086 family inet address 10.86.0.242/24 preferred
set interfaces irb unit 4086 family inet address 10.86.0.242/24 virtual-gateway-address
10.86.0.254
set interfaces irb unit 4086 virtual-gateway-v4-mac 00:00:5e:00:00:04
set interfaces ae15 unit 0 family ethernet-switching vlan members 4086
set vlans 4086 vlan-id 4086
set vlans 4086 l3-interface irb.4086
set vlans 4086 vxlan vni 104086
```

6. Configure the Layer 3 link for VRF-152 towards the PIM gateway.

```
set interfaces irb unit 4087 proxy-macip-advertisement
set interfaces irb unit 4087 virtual-gateway-accept-data
set interfaces irb unit 4087 family inet address 10.87.0.242/24 preferred
set interfaces irb unit 4087 family inet address 10.87.0.242/24 virtual-gateway-address
10.87.0.254
set interfaces irb unit 4087 virtual-gateway-v4-mac 00:00:5e:00:00:04
set interfaces ae15 unit 0 family ethernet-switching vlan members 4087
set vlans 4087 vlan-id 4087
set vlans 4087 l3-interface irb.4087
set vlans 4087 vxlan vni 104087
```

7. Configure a policy to export local routes from VRF-151 and VRF-152 to the PNF.

```
set policy-options policy-statement DIRECT-EXPORT-TO-SRX term 10 from protocol direct
set policy-options policy-statement DIRECT-EXPORT-TO-SRX term 10 then accept
set policy-options policy-statement DIRECT-EXPORT-TO-SRX term 20 from protocol bgp
set policy-options policy-statement DIRECT-EXPORT-TO-SRX term 20 then accept
set policy-options policy-statement DIRECT-EXPORT-TO-SRX term 30 then reject
```

8. Configure a policy to export local routes from VRF-151 and VRF-152 to the PIM gateway.

```
set policy-options policy-statement DIRECT-EXPORT-TO-PIM term 10 from protocol direct
set policy-options policy-statement DIRECT-EXPORT-TO-PIM term 10 then accept
set policy-options policy-statement DIRECT-EXPORT-TO-PIM term 20 then reject
```

9. Configure a routing instance for VRF 151.

```
set routing-instances VRF-151 instance-type vrf
set routing-instances VRF-151 interface irb.601
set routing-instances VRF-151 interface irb.602
set routing-instances VRF-151 interface irb.603
set routing-instances VRF-151 interface irb.604
set routing-instances VRF-151 interface lo0.151
set routing-instances VRF-151 interface irb.4088
set routing-instances VRF-151 interface irb.4086
set routing-instances VRF-151 route-distinguisher 192.186.0.2:151
set routing-instances VRF-151 vrf-target target:100:151
set routing-instances VRF-151 protocols bgp group to-srx type external
set routing-instances VRF-151 protocols bgp group to-srx export DIRECT-EXPORT-TO-SRX
set routing-instances VRF-151 protocols bgp group to-srx local-as 4200000002
set routing-instances VRF-151 protocols bgp group to-srx neighbor 10.88.0.1 family inet
unicast
set routing-instances VRF-151 protocols bgp group to-srx neighbor 10.88.0.1 peer-as
4000000001
set routing-instances VRF-151 protocols bgp group to-srx neighbor 2001:db8::40:58:0:1
family inet6 unicast
set routing-instances VRF-151 protocols bgp group to-srx neighbor 2001:db8::40:58:0:1 peer-
as 4000000001
set routing-instances VRF-151 protocols bgp group to-pim type external
set routing-instances VRF-151 protocols bgp group to-pim export DIRECT-EXPORT-TO-PIM
set routing-instances VRF-151 protocols bgp group to-pim local-as 4200000002
set routing-instances VRF-151 protocols bgp group to-pim neighbor 10.86.0.1 family inet
unicast
set routing-instances VRF-151 protocols bgp group to-pim neighbor 10.86.0.1 peer-as
4100000000
set routing-instances VRF-151 protocols pim rp static address 192.68.151.1 group-ranges
225.0.4.0/24
set routing-instances VRF-151 protocols pim rp static address 192.68.152.1 group-ranges
225.0.5.0/24
set routing-instances VRF-151 protocols pim interface irb.601 family inet
set routing-instances VRF-151 protocols pim interface irb.601 mode sparse-dense
set routing-instances VRF-151 protocols pim interface irb.602 family inet
set routing-instances VRF-151 protocols pim interface irb.602 mode sparse-dense
set routing-instances VRF-151 protocols pim interface irb.604 family inet
set routing-instances VRF-151 protocols pim interface irb.604 mode sparse-dense
set routing-instances VRF-151 protocols pim interface irb.603 family inet
set routing-instances VRF-151 protocols pim interface irb.603 mode sparse-dense
set routing-instances VRF-151 protocols pim interface irb.4086 family inet
set routing-instances VRF-151 protocols pim interface irb.4086 mode sparse-dense
set routing-instances VRF-151 protocols pim interface irb.4088 family inet
set routing-instances VRF-151 protocols pim interface irb.4088 mode sparse-dense
```

10. Configure a routing instance for VRF 152.

```
set routing-instances VRF-152 instance-type vrf
set routing-instances VRF-152 interface irb.605
set routing-instances VRF-152 interface irb.606
set routing-instances VRF-152 interface irb.607
set routing-instances VRF-152 interface irb.608
set routing-instances VRF-152 interface lo0.152
set routing-instances VRF-152 interface irb.4089
set routing-instances VRF-152 interface irb.4087
set routing-instances VRF-152 route-distinguisher 192.186.0.2:152
set routing-instances VRF-152 vrf-target target:100:152
set routing-instances VRF-152 protocols bgp group to-srx type external
set routing-instances VRF-152 protocols bgp group to-srx export DIRECT-EXPORT-TO-SRX
set routing-instances VRF-152 protocols bgp group to-srx local-as 4200000002
set routing-instances VRF-152 protocols bgp group to-srx neighbor 10.89.0.1 family inet
unicast
set routing-instances VRF-152 protocols bgp group to-srx neighbor 10.89.0.1 peer-as
4000000001
set routing-instances VRF-152 protocols bgp group to-srx neighbor 2001:db8::10:59:0:1
family inet6 unicast
set routing-instances VRF-152 protocols bgp group to-srx neighbor 2001:db8::10:59:0:1 peer-
as 4000000001
set routing-instances VRF-152 protocols bgp group to-pim type external
set routing-instances VRF-152 protocols bgp group to-pim export DIRECT-EXPORT-TO-PIM
set routing-instances VRF-152 protocols bgp group to-pim local-as 4200000002
set routing-instances VRF-152 protocols bgp group to-pim neighbor 10.87.0.1 family inet
unicast
set routing-instances VRF-152 protocols bgp group to-pim neighbor 10.87.0.1 peer-as
4100000000
set routing-instances VRF-152 protocols pim rp static address 192.68.151.1 group-ranges
225.0.4.0/24
set routing-instances VRF-152 protocols pim rp static address 192.68.152.1 group-ranges
225.0.5.0/24
set routing-instances VRF-152 protocols pim interface irb.605 family inet
set routing-instances VRF-152 protocols pim interface irb.605 mode sparse-dense
set routing-instances VRF-152 protocols pim interface irb.606 family inet
set routing-instances VRF-152 protocols pim interface irb.606 mode sparse-dense
set routing-instances VRF-152 protocols pim interface irb.607 family inet
set routing-instances VRF-152 protocols pim interface irb.607 mode sparse-dense
set routing-instances VRF-152 protocols pim interface irb.608 family inet
set routing-instances VRF-152 protocols pim interface irb.608 mode sparse-dense
set routing-instances VRF-152 protocols pim interface irb.4087 family inet
set routing-instances VRF-152 protocols pim interface irb.4087 mode sparse-dense
set routing-instances VRF-152 protocols pim interface irb.4089 family inet
set routing-instances VRF-152 protocols pim interface irb.4089 mode sparse-dense
```

11. Repeat this procedure for every spine in your topology.

## Verifying Service Chaining with Multicast

This section shows how to configure service chaining with Multicast. Note the following:

- The multicast source is on VRF-151 - VLAN 601
- The multicast receivers are on VRF-152 - VLAN 606

1. Display the interfaces on VRF-152 that are configured for PIM. This command shows the interfaces that are DRs.

```
host@SPINE5> show pim interfaces instance VRF-152
Stat = Status, V = Version, NbrCnt = Neighbor Count,
S = Sparse, D = Dense, B = Bidirectional,
DR = Designated Router, DDR = Dual DR, DistDR = Distributed DR,
P2P = Point-to-point link, P2MP = Point-to-Multipoint,
Active = Bidirectional is active, NotCap = Not Bidirectional Capable

Name                Stat Mode IP V State        NbrCnt JoinCnt(sg/*g) DR address
irb.4087            Up   SD    4 2 DR,NotCap        4 0/0            10.87.0.245
irb.4089            Up   SD    4 2 DR,NotCap        4 50/50          10.89.0.245
irb.605             Up   SD    4 2 DR,NotCap        3 0/0            10.2.93.245
irb.606             Up   SD    4 2 DR,NotCap        3 0/0            10.2.94.245
irb.607             Up   SD    4 2 DR,NotCap        3 0/0            10.2.95.245
irb.608             Up   SD    4 2 DR,NotCap        3 0/0            10.2.96.245
pime.32780          Up   S     4 2 P2P,NotCap       0 0/0
pime.32781          Up   S     4 2 P2P,NotCap       0 0/0
```

2. Display PIM groups on the Upstream interface towards RP and Source are pointing to irb.4089 – towards the PNF

```
host@SPINE5> show pim join extensive instance VRF-152 225.0.4.1
Instance: PIM.VRF-152 Family: INET
R = Rendezvous Point Tree, S = Sparse, W = Wildcard

Group: 225.0.4.1
    Source: *
    RP: 192.68.151.1
    Flags: sparse,rptree,wildcard
    Upstream interface: irb.4089
    Upstream neighbor: 10.89.0.1
    Upstream state: Join to RP
    Uptime: 00:14:30
    Downstream neighbors:
        Interface: irb.606
            10.2.94.245 State: Join Flags: SRW  Timeout: Infinity
            Uptime: 00:14:30 Time since last Join: 00:14:30
    Number of downstream interfaces: 1
    Number of downstream neighbors: 1

Group: 225.0.4.1
    Source: 10.2.89.5
    Flags: sparse,spt
    Upstream interface: irb.4089
    Upstream neighbor: 10.89.0.1
    Upstream state: Join to Source, No Prune to RP
    Keepalive timeout: 317
    Uptime: 00:13:51
    Downstream neighbors:
        Interface: irb.606
            10.2.94.245 State: Join Flags: S   Timeout: Infinity
            Uptime: 00:13:51 Time since last Join: 00:13:51
    Number of downstream interfaces: 1
    Number of downstream neighbors: 1
```

## Service Chaining— Release History

Table 15 on page 485 provides a history of all of the features in this section and their support within this reference design.

**Table 15: Service Chaining Release History**

| Release | Description |
|---|---|
| 19.1R2 | QFX10002-60C and QFX5120-32C switches running Junos OS Release 19.1R2 and later releases in the same release train support all features documented in this section. |
| 18.4R2 | QFX5120-48Y switches running Junos OS Release 18.4R2 and later releases in the same release train support all features documented in this section. |

**Table 15: Service Chaining Release History *(Continued)***

| Release | Description |
|---|---|
| 18.1R3-S3 | All devices in the reference design that support Junos OS Release 18.1R3-S3 and later releases in the same release train also support all features documented in this section. |

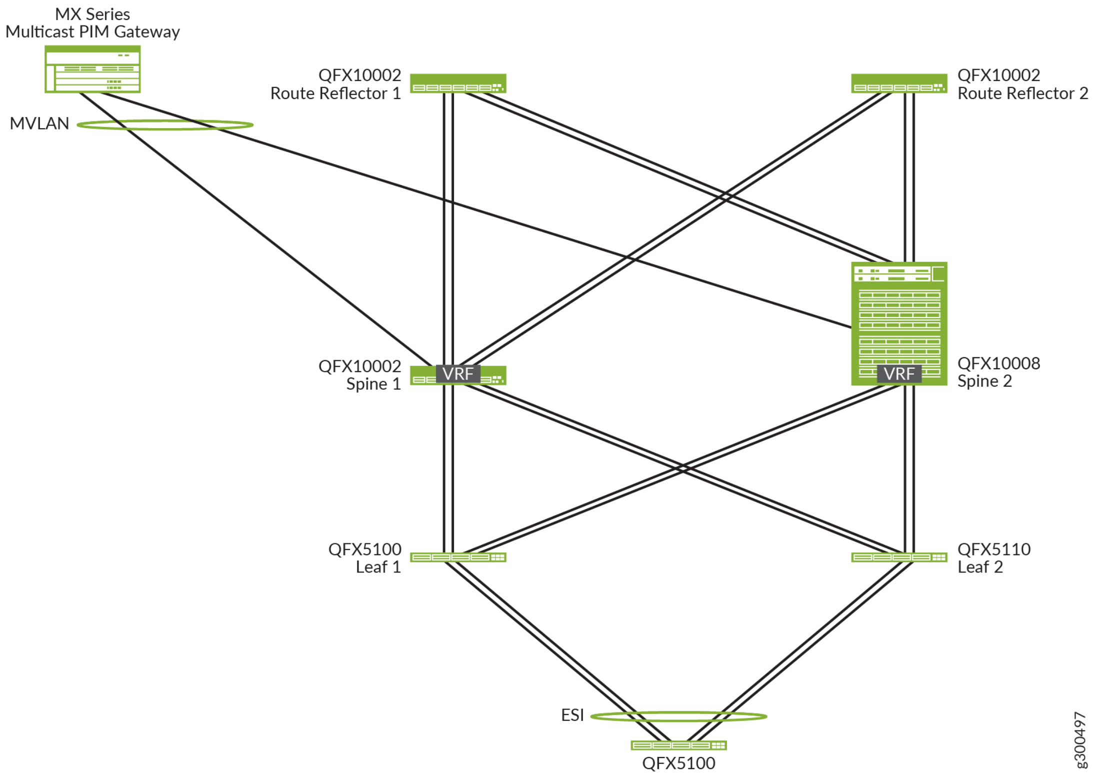
## Multicast IGMP Snooping and PIM Design and Implementation

**IN THIS SECTION**

Use this design to configure Internet Group Management Protocol (IGMP) and Protocol Independent Multicast (PIM) in your fabric to improve multicast replication. IGMP snooping preserves bandwidth because multicast traffic is forwarded only on interfaces where there are IGMP listeners. For instance, every leaf device does not need to receive every instance of multicast traffic.

For an overview of multicast, see "Multicast Optimizations" on page 35.

In this design, we are using an external PIM gateway, which extends multicast beyond the data center, and is useful in DCI implementations.



The next sections show how to configure and verify multicast.

## Configuring IGMP Snooping

In this design, we are using IGMP snooping to constrain multicast traffic in a broadcast domain to interested receivers and multicast devices.

To configure IGMP snooping:

Configure IGMP snooping on all VXLAN enabled VLANs on the leafs. The current implementation does not support IGMP snooping on selected VXLAN enabled VLANs.

```
set protocols igmp-snooping vlan BD-3 proxy
set vlans BD-3 vlan-id 3
set vlans BD-3 vxlan vni 100003
```

## Verifying IGMP Snooping

Enter the following CLI commands to verify IGMP snooping:

1. Verify the local IGMP snooping state on the leaf.

```
user@leaf-2> show igmp snooping membership vlan BD-3
Instance: default-switch

Vlan: BD-3

Learning-Domain: default
Interface: ae11.0, Groups: 2
    Group: 225.0.1.1
        Group mode: Exclude
        Source: 0.0.0.0
        Last reported by: 10.0.3.7
        Group timeout:     215 Type: Dynamic
    Group: 225.0.1.2
        Group mode: Exclude
        Source: 0.0.0.0
        Last reported by: 10.0.3.7
        Group timeout:     207 Type: Dynamic
```

```
user@leaf-2> show evpn igmp-snooping database l2-domain-id 100003
Instance: default-switch
    VN Identifier: 100003
        Group IP: 225.0.1.1, Source IP: 0.0.0.0, Access OIF Count: 1
        Group IP: 225.0.1.2, Source IP: 0.0.0.0, Access OIF Count: 1
```

2. Verify that the leaf is advertising the EVPN Type 7 route where IGMP is snooped.

```
user@leaf-2> show route table __default_evpn__.evpn.0 match-prefix 7:*100003*225.0.1.[12]*
__default_evpn__.evpn.0: 215 destinations, 318 routes (215 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

7:192.168.1.4:10::99080706050403::100003::225.0.1.1::192.168.1.4/600
                   *[EVPN/170] 04:55:32
                       Indirect
7:192.168.1.4:10::99080706050403::100003::225.0.1.2::192.168.1.4/600
```

```
                   *[EVPN/170] 04:55:30
                       Indirect
```

3. Verify that the leaf and its multihomed ESI peer device are both advertising the EVPN Type 6 route for the multicast group.

```
user@leaf-2> show route table default-switch.evpn.0 match-prefix 6:*100003*225.0.1.[12]*
default-switch.evpn.0: 100334 destinations, 198153 routes (100334 active, 0 holddown, 0
hidden)
+ = Active Route, - = Last Active, * = Both

6:192.168.1.3:10::100003::225.0.1.1::192.168.1.3/520
                   *[BGP/170] 04:55:09, localpref 100, from 192.168.0.4
                      AS path: I, validation-state: unverified
                    >  to 172.16.4.1 via ae1.0
                       to 172.16.4.3 via ae2.0
                    [BGP/170] 04:55:11, localpref 100, from 192.168.0.5
                      AS path: I, validation-state: unverified
                    >  to 172.16.4.1 via ae1.0
                       to 172.16.4.3 via ae2.0
6:192.168.1.3:10::100003::225.0.1.2::192.168.1.3/520
                   *[BGP/170] 04:55:07, localpref 100, from 192.168.0.4
                      AS path: I, validation-state: unverified
                    >  to 172.16.4.1 via ae1.0
                       to 172.16.4.3 via ae2.0
                    [BGP/170] 04:55:10, localpref 100, from 192.168.0.5
                      AS path: I, validation-state: unverified
                    >  to 172.16.4.1 via ae1.0
                       to 172.16.4.3 via ae2.0
6:192.168.1.4:10::100003::225.0.1.1::192.168.1.4/520
                   *[EVPN/170] 04:56:16
                       Indirect
6:192.168.1.4:10::100003::225.0.1.2::192.168.1.4/520
                   *[EVPN/170] 04:56:14
                       Indirect
```

## Configuring PIM

To configure PIM:

1. To configure inter-VNI multicast routing at the spine, create a routing instance for a tenant (a leaf device) named VRF-1. Configure the following in the routing instance:

   - Add the IRB interfaces to the leaf devices.
   - Enable PIM and configure the local address for this spine as the rendezvous point (RP).
   - Enable PIM on the IRB interfaces.

   Spine 1:

   ```
   set routing-instances VRF-1 instance-type vrf
   set routing-instances VRF-1 interface irb.1
   set routing-instances VRF-1 interface irb.2
   set routing-instances VRF-1 interface irb.3
   set routing-instances VRF-1 interface irb.4
   set routing-instances VRF-1 interface lo0.10
   set routing-instances VRF-1 route-distinguisher 192.186.0.2:1
   set routing-instances VRF-1 vrf-target target:100:1
   set routing-instances VRF-1 protocols pim rp local address 10.0.1.242
   set routing-instances VRF-1 protocols pim interface irb.1 family inet
   set routing-instances VRF-1 protocols pim interface irb.1 mode sparse-dense
   set routing-instances VRF-1 protocols pim interface irb.2 family inet
   set routing-instances VRF-1 protocols pim interface irb.2 mode sparse-dense
   set routing-instances VRF-1 protocols pim interface irb.3 family inet
   set routing-instances VRF-1 protocols pim interface irb.3 mode sparse-dense
   set routing-instances VRF-1 protocols pim interface irb.4 family inet
   set routing-instances VRF-1 protocols pim interface irb.4 mode sparse-dense
   ```

2. Configure multicast routing on another spine. Configure a corresponding VRF routing instance for the same tenant as in step 6.

   - Add the IRB interfaces toward the leaf devices.
   - Enable PIM and configure the RP address on spine 1 as the static RP.
   - Enable PIM on the IRB interfaces.

   Spine 2:

   ```
   set routing-instances VRF-1 instance-type vrf
   set routing-instances VRF-1 interface irb.1
   set routing-instances VRF-1 interface irb.2
   set routing-instances VRF-1 interface irb.3
   set routing-instances VRF-1 interface irb.4
   ```

```
set routing-instances VRF-1 interface lo0.10
set routing-instances VRF-1 route-distinguisher 192.168.0.3:1
set routing-instances VRF-1 vrf-target target:100:1
set routing-instances VRF-1 protocols pim rp static address 10.0.1.242
set routing-instances VRF-1 protocols pim interface irb.1 family inet
set routing-instances VRF-1 protocols pim interface irb.1 mode sparse-dense
set routing-instances VRF-1 protocols pim interface irb.2 family inet
set routing-instances VRF-1 protocols pim interface irb.2 mode sparse-dense
set routing-instances VRF-1 protocols pim interface irb.3 family inet
set routing-instances VRF-1 protocols pim interface irb.3 mode sparse-dense
set routing-instances VRF-1 protocols pim interface irb.4 family inet
set routing-instances VRF-1 protocols pim interface irb.4 mode sparse-dense
```

## Verifying PIM

Enter the following commands to verify PIM:

1. On spine 1, check the PIM control plane on the RP, and verify that:

   - PIM joins are created from Type 6 routes that are generated by leaf devices.
   - IGMP reports are coming from non-IGMP snooping capable leaf devices.

```
user@spine-1> show pim join instance VRF-1 225.0.1.0/30 extensive
Instance: PIM.VRF-1 Family: INET
R = Rendezvous Point Tree, S = Sparse, W = Wildcard

Group: 225.0.1.1
    Source: *
    RP: 10.0.1.242
    Flags: sparse,rptree,wildcard
    Upstream interface: Local
    Upstream neighbor: Local
    Upstream state: Local RP
    Uptime: 15:18:24
    Downstream neighbors:
        Interface: irb.3
            10.0.3.242 State: Join Flags: SRW  Timeout: Infinity
            Uptime: 15:17:51 Time since last Join: 15:17:51
        Interface: irb.2
            10.0.2.242 State: Join Flags: SRW  Timeout: Infinity
            Uptime: 14:51:29 Time since last Join: 14:51:29
        Interface: irb.1
            10.0.1.242 State: Join Flags: SRW  Timeout: Infinity
            Uptime: 05:31:09 Time since last Join: 05:31:09
            10.0.1.245 State: Join Flags: SRW Timeout: 199
            Uptime: 15:17:28 Time since last Join: 00:00:11
    Number of downstream interfaces: 3
    Number of downstream neighbors: 4

Group: 225.0.1.2
    Source: *
    RP: 10.0.1.242
    Flags: sparse,rptree,wildcard
    Upstream interface: Local
    Upstream neighbor: Local
    Upstream state: Local RP
    Uptime: 15:18:24
    Downstream neighbors:
        Interface: irb.3
            10.0.3.242 State: Join Flags: SRW  Timeout: Infinity
            Uptime: 15:17:51 Time since last Join: 15:17:51
        Interface: irb.2
            10.0.2.242 State: Join Flags: SRW  Timeout: Infinity
            Uptime: 14:51:29 Time since last Join: 14:51:29
        Interface: irb.1
            10.0.1.242 State: Join Flags: SRW  Timeout: Infinity
            Uptime: 05:31:09 Time since last Join: 05:31:09
            10.0.1.245 State: Join Flags: SRW Timeout: 199
            Uptime: 15:17:28 Time since last Join: 00:00:11
    Number of downstream interfaces: 3
    Number of downstream neighbors: 4
```

2. On the spine that is configured as the PIM DR, verify the multicast forwarding state from the spine to the tenant VRF. To do so:

   a. Enter **show pim interfaces instance** on all spines, and check the State column to see which IRB interface is a DR.

```
user@spine-1> show pim interfaces instance VRF-1
Stat = Status, V = Version, NbrCnt = Neighbor Count,
S = Sparse, D = Dense, B = Bidirectional,
DR = Designated Router, DDR = Dual DR, DistDR = Distributed DR,
P2P = Point-to-point link, P2MP = Point-to-Multipoint,
Active = Bidirectional is active, NotCap = Not Bidirectional Capable

Name                Stat Mode IP V State        NbrCnt JoinCnt(sg/*g) DR address
irb.1               Up   SD    4 2 DR,NotCap         3 2/100          10.0.1.245
irb.2               Up   SD    4 2 DR,NotCap         3 0/0            10.0.2.245
irb.3               Up   SD    4 2 DR,NotCap         3 0/0            10.0.3.245
irb.4               Up   SD    4 2 DR,NotCap         3 98/0           10.0.4.245
pime.32769          Up   S     4 2 P2P,NotCap        0 0/0
irb.1               Up   SD    6 2 DR,NotCap         2 0/0            fe80:db8:10:0:1::254
irb.2               Up   SD    6 2 DR,NotCap         2 0/0            fe80:db8:10:0:2::254
irb.3               Up   SD    6 2 DR,NotCap         2 0/0            fe80:db8:10:0:3::254
irb.4               Up   SD    6 2 DR,NotCap         2 0/0            fe80:db8:10:0:4::254
```

b. On the PIM DR, display the multicast forwarding state.

```
user@spine-2> show multicast route extensive instance VRF-1 group 225.0.1.0/30
Instance: VRF-1 Family: INET

Group: 225.0.1.1
    Source: 10.0.1.5/32
    Upstream interface: irb.1
    Downstream interface list:
        irb.3 irb.2
    Number of outgoing interfaces: 3
    Session description: Unknown
    Statistics: 69 kBps, 559 pps, 309165 packets
    Next-hop ID: 2109194
    Upstream protocol: PIM
    Route state: Active
    Forwarding state: Forwarding
    Cache lifetime/timeout: 360 seconds
    Wrong incoming interface notifications: 0
    Uptime: 00:09:13

Group: 225.0.1.2
    Source: 10.0.1.5/32
    Upstream interface: irb.1
    Downstream interface list:
        irb.3 irb.2
    Number of outgoing interfaces: 3
    Session description: Unknown
    Statistics: 68 kBps, 554 pps, 307024 packets
    Next-hop ID: 2109194
    Upstream protocol: PIM
    Route state: Active
    Forwarding state: Forwarding
    Cache lifetime/timeout: 360 seconds
    Wrong incoming interface notifications: 0
    Uptime: 00:09:13
```

## Multicast — Feature Summary

Table 16 on page 494 provides a history of the features described in this section and their support within this reference design.

**Table 16: Multicast Feature Summary**

| Hardware | IGMPv2 Snooping | EVPN Type 6 SMET Routes | Inter-VNI Multicast with PIM Gateway | PIM to External Rendezvous Point (From Border) |
|---|---|---|---|---|
| QFX5100[1] | Not supported | Not supported | Not supported | Not supported |
| QFX5110-32Q, QFX5110-48S | 18.1R3-S3 | 18.4R2 | Not supported | Not supported |
| QFX5120-48Y | 18.4R2 | 18.4R2 | Not supported | Not supported |
| QFX5120-32C | 19.1R2 | 19.1R2 | Not supported | Not supported |
| QFX5200-32C[1], QFX5200-48Y[1] | Not supported | Not supported | Not supported | Not supported |
| QFX10002-36Q/72Q, QFX10008, QFX10016 | 18.1R3-S3 | 18.4R2 | 18.1R3-S3 | 17.3R3-S1 |

**Table 16: Multicast Feature Summary *(Continued)***

| Hardware | IGMPv2 Snooping | EVPN Type 6 SMET Routes | Inter-VNI Multicast with PIM Gateway | PIM to External Rendezvous Point (From Border) |
|---|---|---|---|---|
| QFX10002-60C[2] | 20.2R2 | 20.2R2 | 20.2R2 | 20.2R2 |
| MX204; MX240, MX480, MX960 with MPC7E; MX10003; | Not supported | Not supported | Not supported | Not supported |

[1]Make sure that IGMP snooping is not enabled on these QFX switches. If IGMP snooping is inadvertently enabled, these switches might process EVPN Type 6 routes that are reflected to them.

[2]The QFX10002-60C switch supports multicast at a lower scale than the QFX10002-36Q/72Q switches.

## RELATED DOCUMENTATION

*Overview of Multicast Forwarding with IGMP Snooping or MLD Snooping in an EVPN-VXLAN Environment*

*Multicast Support in EVPN-VXLAN Overlay Networks*

# Multicast Optimization Design and Implementation

**IN THIS SECTION**

Juniper Networks supports the multicast optimization features discussed in this section in both centrally-routed bridging (CRB) and edge-routed bridging (ERB) overlays.

This design assumes that an EVPN-VXLAN ERB overlay is already running for IPv4 unicast traffic. (See "Edge-Routed Bridging Overlay Design and Implementation" on page 206 for information about configuring edge-routed bridging.) However, the multicast optimization features uses a centrally routed approach.

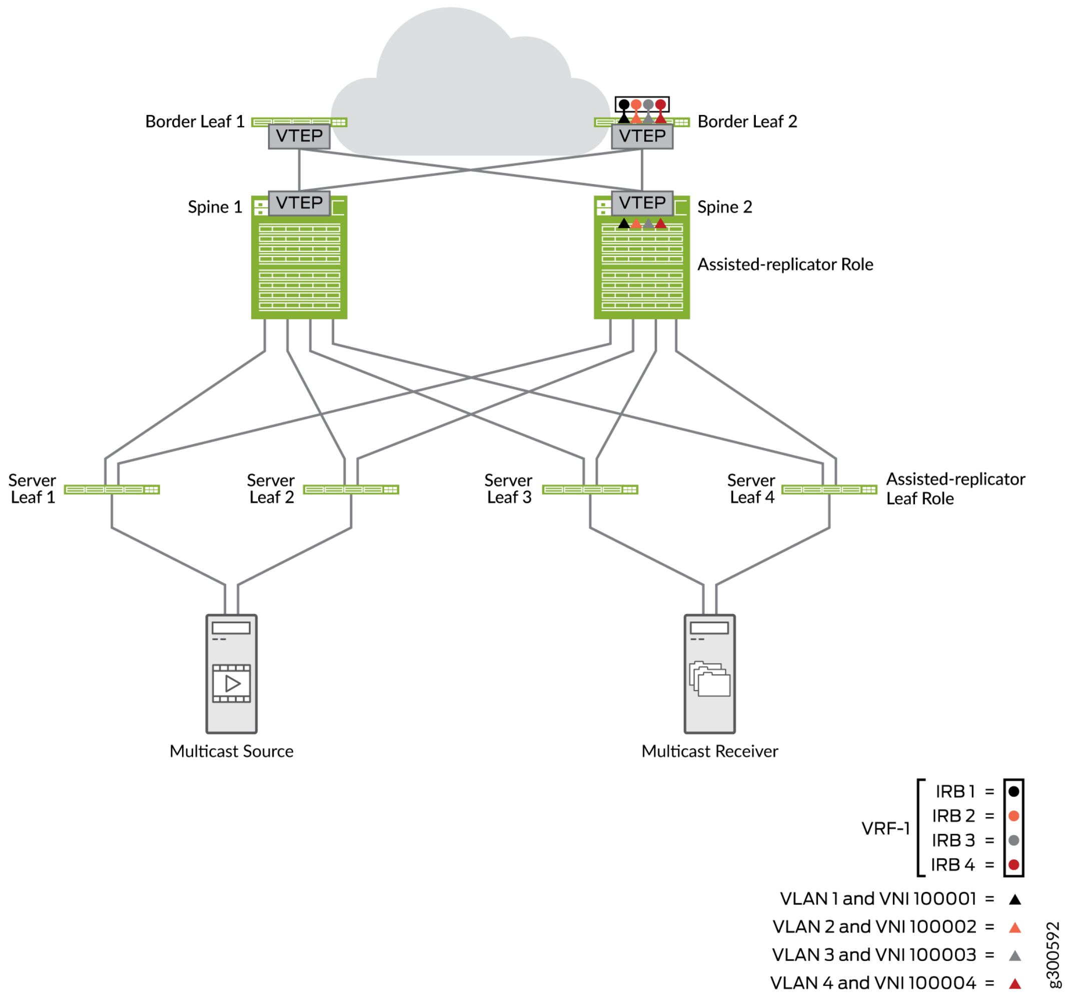
**NOTE**: Starting in Junos OS and Junos OS Evolved Release 22.2R2, we recommend deploying the optimized intersubnet multicast (OISM) solution for ERB overlay unicast EVPN-VXLAN networks that include multicast traffic. OISM combines the best aspects of ERB and CRB overlay designs together to provide the most efficient multicast traffic flow in ERB overlay fabrics.

We describe the OISM configuration that we validated in our ERB overlay reference architecture here:

- "Optimized Intersubnet Multicast (OISM) with Assisted Replication (AR) for Edge-Routed Bridging Overlays" on page 509

This section shows how to add centrally-routed multicast optimizations to the edge-routed bridging topology shown in Figure 88 on page 497.

**Figure 88: Topology for Multicast Optimizations in an Edge-Routed Bridging Overlay**



Multicast is configured as follows:

- Server leaf devices are set up in AR leaf role and for IGMP snooping.
- Spine devices are set up in AR replicator role.
- Border leaf devices are set up for multicast routing.

**NOTE**: If your multicast environment requires assisted replication to handle large multicast flows and multicast routing, we recommend any of the QFX10000 line of switches for the border leaf and border spine roles. However, note that the QFX10002-60C switch supports multicast at a lower scale than the QFX10002-36Q/72Q switches. Also, we do not recommend any of the MX Series routers included in this reference design as a border leaf in a multicast environment with large multicast flows.

For an overview of multicast optimizations, see the Multicast Optimization section in "Data Center Fabric Blueprint Architecture Components" on page 9.

The following sections show how to configure and verify multicast assisted replication:

## Configuring the Server Leaf

We are configuring AR and IGMP snooping on the server leaf. When IGMP snooping is enabled on a device, SMET is also enabled on the device by default.

1. Enable IGMP snooping.

   ```
   set protocols igmp-snooping vlan BD-1
   set protocols igmp-snooping vlan BD-2
   set protocols igmp-snooping vlan BD-3
   set protocols igmp-snooping vlan BD-4
   ```

2. Enable AR in the leaf role. This causes the server leaf to only forward one copy of multicast traffic to the spine, which then performs replication of the multicast traffic.

   The `replicator-activation-delay` is the time, in seconds, the leaf waits before sending the replication to the AR replicator after receiving the AR replicator route from the replicator.

   ```
   set protocols evpn assisted-replication leaf replicator-activation-delay 10
   ```

## Configuring the Spine

We are configuring the spine as AR replicator device.

1. Configure IP addressing for the loopback interfaces. One address is used for the AR replicator role (192.168.102.2). The other address (192.168.2.2) is used for the VTEP tunnel.

```
set interfaces lo0 unit 0 family inet address 192.168.2.2/32 preferred
set interfaces lo0 unit 0 family inet address 192.168.102.2/32
```

2. Configure the spine to act as the AR replicator device.

```
set protocols evpn assisted-replication replicator inet 192.168.102.2
set protocols evpn assisted-replication replicator vxlan-encapsulation-source-ip ingress-
replication-ip
```

3. Configure the loopback interface that is used in the VRF routing instance.

```
set interfaces lo0 unit 1 family inet
```

4. Configure a VRF routing instance.

```
set routing-instances VRF-1 instance-type vrf
set routing-instances VRF-1 interface lo0.1
set routing-instances VRF-1 route-distinguisher 192.168.2.2:1
set routing-instances VRF-1 vrf-target target:100:1
```

5. Configure VLANs to the border leaf.

```
set vlans BD-1 vlan-id 1
set vlans BD-1 vxlan vni 100001
set vlans BD-2 vlan-id 2
set vlans BD-2 vxlan vni 100002
set vlans BD-3 vlan-id 3
set vlans BD-3 vxlan vni 100003
set vlans BD-4 vlan-id 4
set vlans BD-4 vxlan vni 100004
```

6. Configure the EVPN protocol with VXLAN encapsulation.

```
set protocols evpn encapsulation vxlan
set protocols evpn default-gateway no-gateway-community
set protocols evpn extended-vni-list all
```

7. Configure the switch options, and specify that the loopback interface is the VTEP source interface.

```
set switch-options vtep-source-interface lo0.0
set switch-options vrf-target target:10458:0
set switch-options vrf-target auto
```

## Configuring the Border Leaf

This section describes how to set up multicast routing on the border leafs.

**NOTE**: We do not configure AR on the border leafs. In this network design, the two border leafs share a multihomed ESI, and one of the border leaf devices supports AR but the other does not. In this situation, we do not recommend configuring AR on the border leaf that supports this feature. However, if your network includes two border leafs that share a multihomed ESI, and both border leaf devices support AR, we support the configuration of AR on both border leafs.

1. Configure VLANs

```
set vlans BD-1 vlan-id 1
set vlans BD-1 l3-interface irb.1
set vlans BD-1 vxlan vni 100001
set vlans BD-2 vlan-id 2
set vlans BD-2 l3-interface irb.2
set vlans BD-2 vxlan vni 100002
set vlans BD-3 vlan-id 3
set vlans BD-3 l3-interface irb.3
set vlans BD-3 vxlan vni 100003
set vlans BD-4 vlan-id 4
set vlans BD-4 l3-interface irb.4
set vlans BD-4 vxlan vni 100004
```

2. Configure the EVPN protocol with VXLAN encapsulation.

```
set protocols evpn encapsulation vxlan
set protocols evpn default-gateway no-gateway-community
set protocols evpn extended-vni-list all
```

**NOTE**: On QFX5130 and QFX5700 switches, also include the `host-profile` unified forwarding profile option to support an EVPN-VXLAN environment (see *Layer 2 Forwarding Tables* for details):

```
set system packet-forwarding-options forwarding-profile host-profile
```

3. Configure the switch options and specify that the loopback interface is the VTEP source interface.

```
set switch-options vtep-source-interface lo0.0
set switch-options vrf-target target:10458:0
set switch-options vrf-target auto
```

4. Configure the IRBs.

```
set interfaces irb unit 1 virtual-gateway-accept-data
set interfaces irb unit 1 family inet address 10.0.1.239/24 preferred
set interfaces irb unit 1 family inet address 10.0.1.239/24 virtual-gateway-address 10.0.1.254
set interfaces irb unit 1 family inet6 nd6-stale-time 1200
set interfaces irb unit 1 family inet6 address 2001:db8::10:0:1:239/112 preferred
set interfaces irb unit 1 family inet6 address 2001:db8::10:0:1:239/112 virtual-gateway-
address 2001:db8::10:0:1:254
set interfaces irb unit 1 family inet6 address fe80:10:0:1::239/64 virtual-gateway-address
fe80:10:0:1::254
set interfaces irb unit 1 virtual-gateway-v4-mac 00:00:5e:00:00:04
set interfaces irb unit 1 virtual-gateway-v6-mac 00:00:5e:00:00:04
set interfaces irb unit 2 virtual-gateway-accept-data
set interfaces irb unit 2 family inet address 10.0.2.239/24 preferred
set interfaces irb unit 2 family inet address 10.0.2.239/24 virtual-gateway-address 10.0.2.254
set interfaces irb unit 2 family inet6 nd6-stale-time 1200
set interfaces irb unit 2 family inet6 address 2001:db8::10:0:2:239/112 preferred
set interfaces irb unit 2 family inet6 address 2001:db8::10:0:2:239/112 virtual-gateway-
address 2001:db8::10:0:2:254
set interfaces irb unit 2 family inet6 address fe80:10:0:2::239/64 virtual-gateway-address
fe80:10:0:2::254
set interfaces irb unit 2 virtual-gateway-v4-mac 00:00:5e:00:00:04
set interfaces irb unit 2 virtual-gateway-v6-mac 00:00:5e:00:00:04
set interfaces irb unit 3 virtual-gateway-accept-data
set interfaces irb unit 3 family inet address 10.0.3.239/24 preferred
set interfaces irb unit 3 family inet address 10.0.3.239/24 virtual-gateway-address 10.0.3.254
set interfaces irb unit 3 family inet6 nd6-stale-time 1200
set interfaces irb unit 3 family inet6 address 2001:db8::10:0:3:239/112 preferred
set interfaces irb unit 3 family inet6 address 2001:db8::10:0:3:239/112 virtual-gateway-
address 2001:db8::10:0:3:254
set interfaces irb unit 3 family inet6 address fe80:10:0:3::239/64 virtual-gateway-address
fe80:10:0:3::254
set interfaces irb unit 3 virtual-gateway-v4-mac 00:00:5e:00:00:04
set interfaces irb unit 3 virtual-gateway-v6-mac 00:00:5e:00:00:04
set interfaces irb unit 4 virtual-gateway-accept-data
set interfaces irb unit 4 family inet address 10.0.4.239/24 preferred
set interfaces irb unit 4 family inet address 10.0.4.239/24 virtual-gateway-address 10.0.4.254
set interfaces irb unit 4 family inet6 nd6-stale-time 1200
set interfaces irb unit 4 family inet6 address 2001:db8::10:0:4:239/112 preferred
set interfaces irb unit 4 family inet6 address 2001:db8::10:0:4:239/112 virtual-gateway-
address 2001:db8::10:0:4:254
set interfaces irb unit 4 family inet6 address fe80:10:0:4::239/64 virtual-gateway-address
fe80:10:0:4::254
set interfaces irb unit 4 virtual-gateway-v4-mac 00:00:5e:00:00:04
set interfaces irb unit 4 virtual-gateway-v6-mac 00:00:5e:00:00:04
```

5. Configure a VRF routing instance.

```
set routing-instances VRF-1 instance-type vrf
set routing-instances VRF-1 interface lo0.1
set routing-instances VRF-1 interface irb.1
set routing-instances VRF-1 interface irb.2
set routing-instances VRF-1 interface irb.3
set routing-instances VRF-1 interface irb.4
set routing-instances VRF-1 route-distinguisher 192.186.0.1:1
set routing-instances VRF-1 vrf-target target:100:1
```

6. Configure PIM for multicast routing at the border leaf devices.

```
set routing-instances VRF-1 protocols pim rp local address 10.0.1.239
set routing-instances VRF-1 protocols pim interface irb.1 family inet
set routing-instances VRF-1 protocols pim interface irb.1 mode sparse-dense
set routing-instances VRF-1 protocols pim interface irb.2 family inet
set routing-instances VRF-1 protocols pim interface irb.2 mode sparse-dense
set routing-instances VRF-1 protocols pim interface irb.3 family inet
set routing-instances VRF-1 protocols pim interface irb.3 mode sparse-dense
set routing-instances VRF-1 protocols pim interface irb.4 family inet
set routing-instances VRF-1 protocols pim interface irb.4 mode sparse-dense
```

## Verifying Assisted Replication on the Server Leaf

The server leaf is in the role of AR leaf device. This means that it does not perform ingress replication. Instead, it forwards one copy of multicast traffic to the spine, which is configured as the AR replicator device.

1. Verify that the spines are in the role of Assisted Replicator and are receiving Type 3 routes. Address 192.168.102.1 is Spine 1, and address 192.168.102.2 is Spine 2.

```
user@server-leaf> show route table bgp.evpn.0 match-prefix 3:*100001*192.168.102.* extensive
| match "3:192.168.2|ASSISTED-REPLICATION"
3:192.168.2.1:10000::100001::192.168.102.1/248 IM (2 entries, 0 announced)
                PMSI: Flags 0x8: Label 6250: Type ASSISTED-REPLICATION 192.168.102.1 Role AR-
REPLICATOR
                PMSI: Flags 0x8: Label 6250: Type ASSISTED-REPLICATION 192.168.102.1 Role AR-
REPLICATOR
3:192.168.2.2:10000::100001::192.168.102.2/248 IM (2 entries, 0 announced)
                PMSI: Flags 0x8: Label 6250: Type ASSISTED-REPLICATION 192.168.102.2 Role AR-
REPLICATOR
                PMSI: Flags 0x8: Label 6250: Type ASSISTED-REPLICATION 192.168.102.2 Role AR-
REPLICATOR
```

```
user@server-leaf>  show route table bgp.evpn.0 match-prefix 3:*100001*192.168.102.1* extensive
bgp.evpn.0: 156388 destinations, 299429 routes (156388 active, 0 holddown, 0 hidden)
3:192.168.2.1:10000::100001::192.168.102.1/248 IM (2 entries, 0 announced)
        *BGP    Preference: 170/-101
                Route Distinguisher: 192.168.2.1:10000
                PMSI: Flags 0x8: Label 6250: Type ASSISTED-REPLICATION 192.168.102.1 Role AR-
REPLICATOR
                Next hop type: Indirect, Next hop index: 0
                Address: 0x1a6b08f0
                Next-hop reference count: 4000
                Source: 192.168.2.1
                Protocol next hop: 192.168.102.1
                Indirect next hop: 0x2 no-forward INH Session ID: 0x0
                State: <Active Int Ext>
                Local AS: 4210000001 Peer AS: 4210000001
                Age: 4:58:08    Metric2: 0
                Validation State: unverified
                Task: BGP_4210000001.192.168.2.1
                AS path: I
                Communities: target:32897:268535457 encapsulation:vxlan(0x8) evpn-mcast-
flags:0x4:extended-MH-AR
                Import Accepted
                Localpref: 100
                Router ID: 192.168.2.1
                Secondary Tables: default-switch.evpn.0
                Indirect next hops: 1
                        Protocol next hop: 192.168.102.1
                        Indirect next hop: 0x2 no-forward INH Session ID: 0x0
                        Indirect path forwarding next hops: 1
                                Next hop type: Router
                                Next hop: 172.16.109.0 via ae3.0
                                Session Id: 0x0
                                192.168.102.1/32 Originating RIB: inet.0
                                  Node path count: 1
                                  Forwarding nexthops: 1
                                        Nexthop: 172.16.109.0 via ae3.0
                                        Session Id: 0
         BGP    Preference: 170/-101
                Route Distinguisher: 192.168.2.1:10000
                PMSI: Flags 0x8: Label 6250: Type ASSISTED-REPLICATION 192.168.102.1 Role AR-
REPLICATOR
                Next hop type: Indirect, Next hop index: 0
                Address: 0x1a6b08f0
                Next-hop reference count: 4000
                Source: 192.168.2.2
                Protocol next hop: 192.168.102.1
                Indirect next hop: 0x2 no-forward INH Session ID: 0x0
                State: <NotBest Int Ex>
                Inactive reason: Not Best in its group - Cluster list length
                Local AS: 4210000001 Peer AS: 4210000001
                Age: 5:14:41    Metric2: 0
                Validation State: unverified
                Task: BGP_4210000001.192.168.2.2
                AS path: I  (Originator)
                Cluster list:  192.168.2.10
                Originator ID: 192.168.2.1
                Communities: target:32897:268535457 encapsulation:vxlan(0x8) evpn-mcast-
flags:0x4:extended-MH-AR
                Import Accepted
                Localpref: 100
                Router ID: 192.168.2.2
```

```
                  Secondary Tables: default-switch.evpn.0
                  Indirect next hops: 1
                          Protocol next hop: 192.168.102.1
                          Indirect next hop: 0x2 no-forward INH Session ID: 0x0
                          Indirect path forwarding next hops: 1
                                  Next hop type: Router
                                  Next hop: 172.16.109.0 via ae3.0
                                  Session Id: 0x0
                                  192.168.102.1/32 Originating RIB: inet.0
                                    Node path count: 1
                                    Forwarding nexthops: 1
                                          Nexthop: 172.16.109.0 via ae3.0
                                          Session Id: 0
```

2. Verify the spine that is set up as the AR device for the VLAN. 192.168.102.2 is the address of the AR device.

```
user@server-leaf> show evpn multicast-snooping assisted-replication next-hops l2-domain-id
100001
Instance: default-switch
AR Role: AR Leaf

  VN Identifier: 100001
    Load Balance Nexthop Index: 134135
      Load balance to:
      Nexthop Index    Interface        AR IP
      23119            vtep.32881       192.168.102.1
      21753            vtep.32770       192.168.102.2 (Designated Node)
```

## Verifying Assisted Replication on the Spine

Verify that server leaf devices 1 through 4 are AR leaf devices. (The loopback addresses of server leaf devices 1 through 4 are 192.168.0.1, 192.168.0.2, 192.168.0.3, and 192.168.0.4, respectively.) The border leaf devices are not set up for assisted replication.

```
user@spine> show route table bgp.evpn.0 match-prefix 3:*100001*192.168.0.* extensive | match
"3:192.168.0.|LEAF"| except "PMSI|Path"
3:192.168.0.1:10000::100001::192.168.0.1/248 IM (1 entry, 1 announced)
     PMSI: Flags 0x10: Label 6250: Type INGRESS-REPLICATION 192.168.0.1 AR-LEAF  ## Leaf 1
     PMSI: Flags 0x10: Label 6250: Type INGRESS-REPLICATION 192.168.0.1 AR-LEAF
3:192.168.0.2:10::100001::192.168.0.2/248 IM (1 entry, 1 announced)
     PMSI: Flags 0x10: Label 6250: Type INGRESS-REPLICATION 192.168.0.2 AR-LEAF  ## Leaf 2
     PMSI: Flags 0x10: Label 6250: Type INGRESS-REPLICATION 192.168.0.2 AR-LEAF
3:192.168.0.3:10000::100001::192.168.0.3/248 IM (1 entry, 1 announced)
     PMSI: Flags 0x10: Label 6250: Type INGRESS-REPLICATION 192.168.0.3 AR-LEAF  ## Leaf 3
     PMSI: Flags 0x10: Label 6250: Type INGRESS-REPLICATION 192.168.0.3 AR-LEAF
3:192.168.0.4:10000::100001::192.168.0.4/248 IM (2 entries, 1 announced)
     PMSI: Flags 0x10: Label 6250: Type INGRESS-REPLICATION 192.168.0.4 AR-LEAF  ## Leaf 4
     PMSI: Flags 0x10: Label 6250: Type INGRESS-REPLICATION 192.168.0.4 AR-LEAF
3:192.168.0.10:10000::100001::192.168.0.10/248 IM (1 entry, 1 announced)  ## Border Leaf 1
3:192.168.0.11:10000::100001::192.168.0.11/248 IM (1 entry, 1 announced)  ## Border Leaf 2
```

```
user@spine>  show route table bgp.evpn.0 match-prefix 3:*100001*192.168.0.1 extensive
bgp.evpn.0: 362179 destinations, 504791 routes (347873 active, 14306 holddown, 0 hidden)
3:192.168.0.1:10000::100001::192.168.0.1/248 IM (1 entry, 1 announced)
TSI:
Page 0 idx 0, (group overlay-bgp-rr type Internal) Type 1 val 0x1af46804 (adv_entry)
   Advertised metrics:
     Nexthop: 192.168.0.1
     Localpref: 100
     AS path: [4210000001] I
     Communities: target:32897:268535457 encapsulation:vxlan(0x8) evpn-mcast-flags:0x1:snooping-
enabled
     PMSI: Flags 0x10: Label 6250: Type INGRESS-REPLICATION 192.168.0.1 AR-LEAF
     Cluster ID: 192.168.2.10
     Originator ID: 192.168.0.1
Page 0 idx 1, (group overlay-bgp type Internal) Type 1 val 0x1af46510 (adv_entry)
   Advertised metrics:
     Nexthop: 192.168.0.1
     Localpref: 100
     AS path: [4210000001] I
     Communities: target:32897:268535457 encapsulation:vxlan(0x8) evpn-mcast-flags:0x1:snooping-
enabled
     PMSI: Flags 0x10: Label 6250: Type INGRESS-REPLICATION 192.168.0.1 AR-LEAF
     Cluster ID: 192.168.2.10
     Originator ID: 192.168.0.1
    Advertise: 0000001e
Path 3:192.168.0.1:10000::100001::192.168.0.1
from 192.168.0.1
```

```
Vector len 4.  Val: 0 1
        *BGP    Preference: 170/-101
                Route Distinguisher: 192.168.0.1:10000
                PMSI: Flags 0x10: Label 6250: Type INGRESS-REPLICATION 192.168.0.1 AR-LEAF
                Next hop type: Indirect, Next hop index: 0
                Address: 0x11bd0d90
                Next-hop reference count: 35023
                Source: 192.168.0.1
                Protocol next hop: 192.168.0.1
                Indirect next hop: 0x2 no-forward INH Session ID: 0x0
                State: <Active Int Ext>
                Local AS: 4210000001 Peer AS: 4210000001
                Age: 18:34:04   Metric2: 0
                Validation State: unverified
                Task: BGP_4210000001.192.168.0.1
                Announcement bits (1): 1-BGP_RT_Background
                AS path: I
                Communities: target:32897:268535457 encapsulation:vxlan(0x8) evpn-mcast-
flags:0x1:snooping-enabled
                Import Accepted
                Localpref: 100
                Router ID: 192.168.0.1
                Secondary Tables: default-switch.evpn.0
                Indirect next hops: 1
                        Protocol next hop: 192.168.0.1
                        Indirect next hop: 0x2 no-forward INH Session ID: 0x0
                        Indirect path forwarding next hops: 1
                                Next hop type: Router
                                Next hop: 172.16.101.1 via ae1.0
                                Session Id: 0x0
                                192.168.0.1/32 Originating RIB: inet.0
                                  Node path count: 1
                                  Forwarding nexthops: 1
                                        Nexthop: 172.16.101.1 via ae1.0
                                        Session Id: 0
```

## Multicast Optimization with a Centrally Routed Multicast Design—Feature Summary

Table 17 on page 508 provides a history of the features described in this section and their support within this reference design.

**Table 17: Multicast Optimization Feature Summary (Centrally Routed Multicast Design)**

| Hardware | IGMPv2 Snooping | EVPN Type 6 SMET Routes | Inter-VNI Multicast with PIM Gateway | Assisted Replication | PIM to External Rendezvous Point (From Border) |
|---|---|---|---|---|---|
| QFX5100[1] | Not supported | Not supported | Not supported | Not supported | Not supported |
| QFX5110-32Q, QFX5110-48S | 18.1R3-S3 | 18.4R2 | Not supported | Not supported | Not supported |
| QFX5120-48Y | 18.4R2 | 18.4R2 | Not supported | Not supported | Not supported |
| QFX5120-32C | 19.1R2 | 19.1R2 | Not supported | Not supported | Not supported |
| QFX5200-32C[1], QFX5200-48Y[1] | Not supported | Not supported | Not supported | Not supported | Not supported |
| QFX10002-36Q/72Q, QFX10008, QFX10016 | 18.1R3-S3 | 18.4R2 | 18.1R3-S3 | 18.4R2 | 17.3R3-S1 |
| QFX10002-60C[2] | 20.2R2 | 20.2R2 | 20.2R2 | 20.2R2 | 20.2R2 |
| MX204; MX240, MX480, MX960 with MPC7E; MX10003; | Not supported | Not supported | Not supported | Not supported | Not supported |

[1]Make sure that IGMP snooping is not enabled on these QFX switches. If IGMP snooping is inadvertently enabled, these switches might process EVPN Type 6 routes that are reflected to them.

[2]The QFX10002-60C switch supports multicast at a lower scale than the QFX10002-36Q/72Q switches.

RELATED DOCUMENTATION

*Multicast Support in EVPN-VXLAN Overlay Networks*

# Optimized Intersubnet Multicast (OISM) with Assisted Replication (AR) for Edge-Routed Bridging Overlays

IN THIS SECTION

This example shows how to configure our original optimized intersubnet multicast (OISM) implementation with assisted replication (AR) in a large EVPN-VXLAN edge-routed bridging (ERB) overlay fabric.

In EVPN ERB overlay fabric designs, the leaf devices route traffic between tenant VLANs as well as forwarding traffic within tenant VLANs. To support efficient multicast traffic flow in a scaled ERB overlay fabric with both internal and external multicast sources and receivers, we provide a multicast configuration model based on the IETF draft specification draft-ietf-bess-evpn-irb-mcast, EVPN

Optimized Inter-Subnet Multicast (OISM) Forwarding. OISM combines the best aspects of ERB and CRB overlay designs for multicast traffic together to provide the most efficient multicast traffic flow in ERB overlay fabrics, especially in scaled environments.

OISM enables ERB overlay fabrics to:

- Support multicast traffic with sources and receiver both inside and outside the fabric.
- Minimize multicast control and data traffic flow in the EVPN core to optimize performance in scaled environments.

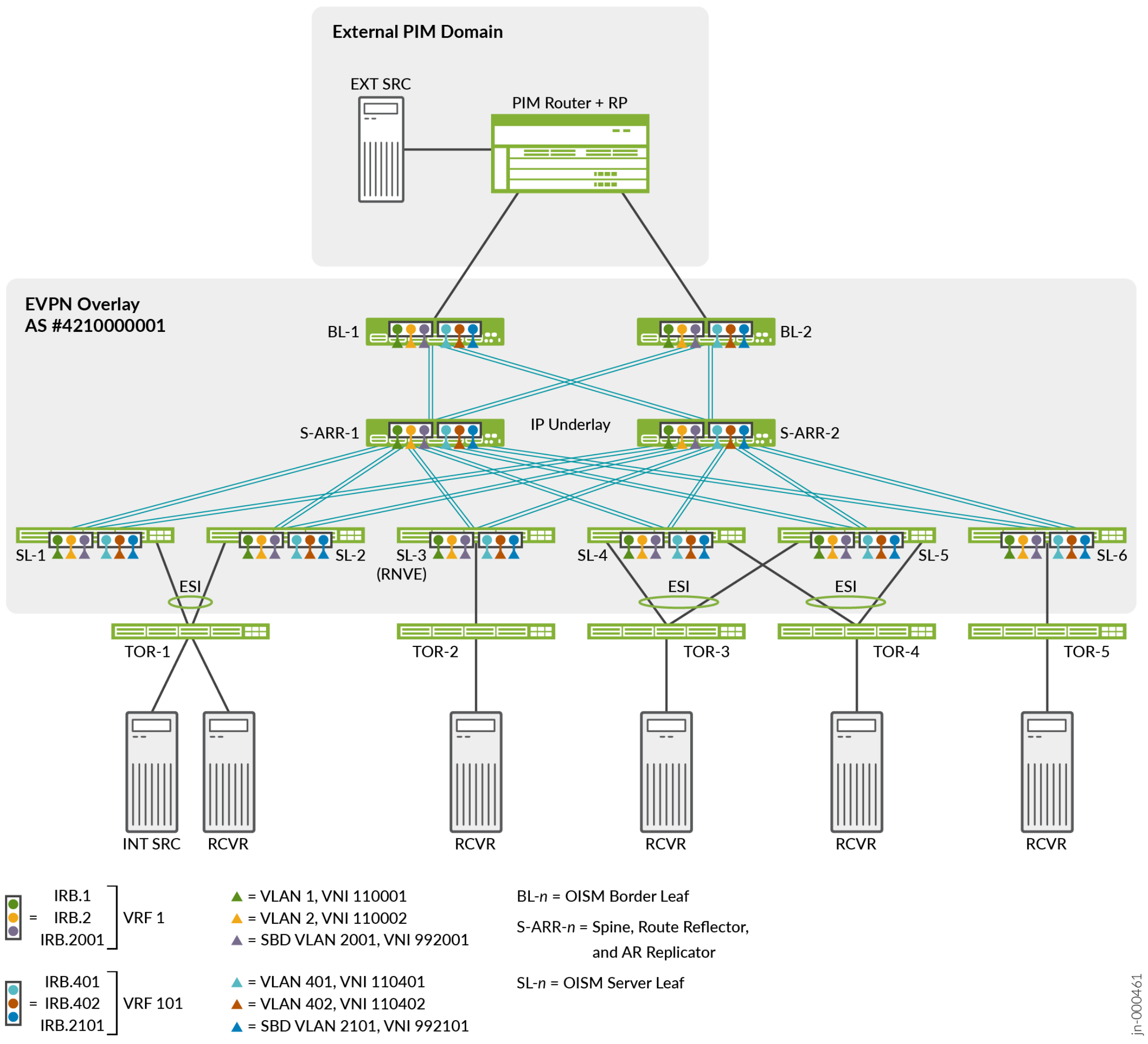
Our original OISM implementation, called *regular OISM*, uses a symmetric bridge domains model. With this model, you configure all tenant VLANs in the fabric symmetrically on all OISM devices. This example shows a regular OISM configuration.

On some platforms we also support an enhanced version of OISM that uses an asymmetric bridge domains model in which you don't need to configure all tenant VLANs symmetrically on all OISM devices. The OISM configuration elements and steps are almost the same for regular and enhanced OISM. The main difference, besides setting either OISM mode, is how you configure the tenant VLANs with each mode. Enhanced OISM also has some important operational differences to support the asymmetric bridge domains model. See "Enhanced Optimized Intersubnet Multicast (OISM) Implementation" on page 576 for an enhanced OISM configuration.

Regular OISM can also interoperate with the assisted replication (AR) feature to offload and load-balance multicast replication in the fabric to the devices better equipped to handle the load. See Assisted Replication Multicast Optimization in EVPN Networks for details on how AR works, and "Optimized Intersubnet Multicast for ERB Overlay Networks" on page 39 earlier in this guide for a short summary of how OISM works with AR. This example includes configuring AR with regular OISM.

Figure 89 on page 511 shows the ERB overlay reference architecture in which we validated OISM and AR on supported devices in this example.

**Figure 89: Edge-Routed Bridging Overlay Fabric with OISM and AR**



Here is a summary of OISM components, configuration elements, and operation in this environment. For full details on how OISM works in different scenarios and available OISM support on different platforms, see Optimized Intersubnet Multicast in EVPN Networks.

- In this example, the OISM devices take one of these device roles:
  - Server leaf (SL)—Leaf devices that link to the access side (internal) top-of-rack (TOR) devices that host the multicast servers and receivers inside the fabric. The SL devices can act as AR leaf devices.
  - Border Leaf (BL)—Leaf devices that link to an external PIM domain to manage multicast flow to and from external multicast sources and receivers. The BL devices can also act as AR leaf devices.

  - AR Replicator Spine (S-ARR)—IP fabric transit devices that serve as route reflectors in the ERB overlay fabric and also as the AR replicator devices working with OISM. When the spine devices in an ERB overlay act as AR replicators, they must run EVPN-VXLAN and no longer function simply as lean spines.

- In this example, you configure OISM with a MAC-VRF EVPN instance with the VLAN-aware service type (supports multiple VLANs in the MAC-VRF instance) on all SL, BL, and S-ARR devices. You don't need to configure an EVPN instance on the external PIM router.

- This example configures regular OISM, which uses a symmetric bridge domains model. With this model, you configure all tenant VLANs (also called *revenue bridge domains* or *revenue VLANs*) and virtual routing and forwarding (VRF) instances in the fabric on all OISM leaf devices. If you configure OISM with AR, you also configure these elements on the spine devices that act as AR replicators.

- OISM leaf devices do intrasubnet bridging, and use a local routing model for intersubnet (Layer 3 [L3]) multicast traffic to conserve bandwidth and avoid hairpinning in the EVPN core. See Local Routing on OISM Devices for details.

  - SL devices forward multicast source traffic into the EVPN core only on the source VLAN.

  - BL devices forward traffic from external multicast sources into the EVPN core toward internal receivers only on a *supplemental bridge domain* called the *SBD*. The SBD design enables the local routing model and solves other issues with externally sourced traffic. For each tenant VRF instance, you assign a VLAN and a corresponding IRB interface for the SBD.

  - OISM SL devices receive multicast traffic from internal sources on the source VLAN, or from external sources through the BL devices on the SBD. For internally sourced traffic, the SL devices locally bridge the traffic to receivers on the source VLAN, and use IRB interfaces to locally route the traffic to receivers on other VLANs. Upon receiving traffic from outside the fabric, the SL devices use IRB interfaces to locally route the traffic from the SBD to the tenant VLANs and then to their locally attached receivers.

- We support OISM with IGMPv2 (any-source multicast [ASM] reports only) or IGMPv3 (source-specific multicast [SSM] reports only). OISM requires that you enable IGMP snooping with either IGMP version. We use Protocol Independent Multicast (PIM) in sparse mode for multicast routing with different options on SL and BL devices according to their functions.

**NOTE**: To support both IGMPv2 and IGMPv3 receivers on the same device, you must:

- Use different tenant VRF instances to support the receivers for each IGMP version.

- Configure different VLANs and corresponding IRB interfaces that support the receivers for each IGMP version.

- Associate the IRB interfaces for each version with the corresponding tenant VRF instance.

  See Considerations for OISM Configurations for details on the required configuration considerations. The configuration we tested here accommodates receivers for both versions on the same device.

- With IGMP snooping, OISM also optimizes multicast traffic using EVPN Type 6 routes for selective multicast Ethernet tag (SMET) forwarding. With SMET, OISM devices only forward traffic for a multicast group to other devices in the fabric with receivers that show interest in receiving that traffic. (Multicast receivers send IGMP join messages to request traffic for a multicast group.)

  In the regular OISM model used here, OISM devices advertise EVPN Type 6 routes only on the SBD.

- OISM supports EVPN multihoming with multicast traffic. The fabric can include receivers behind TOR devices that are multihomed in an Ethernet segment (ES) to more than one OISM leaf device. You configure an ES identifier (ESI) for the links in the ES.

  OISM devices use EVPN Type 7 (Join Sync) and Type 8 (Leave Sync) routes to synchronize the multicast state among the multihoming peer devices that serve an ES.

In this environment, we validate OISM and AR together at scale with the AR replicator role on the spine devices. "Configure AR Replicator Role on OISM Spine Devices and AR Leaf Role on OISM Leaf Devices" on page 540 explains more about how AR works in this example. When the AR replicator role is not collocated with an OISM border leaf role on the same device, as in this example, we say the AR replicator operates in standalone AR replicator mode. The OISM SL and BL devices act as AR leaf devices.

We call devices that don't support AR *regular network virtualization edge (RNVE) devices*. The test environment includes an SL device (see SL-3 in Figure 89 on page 511) on which we don't configure the AR leaf role to simulate an RNVE device. With RNVE devices in the fabric:

- The RNVE devices use ingress replication to forward multicast traffic to other leaf devices in the fabric.

- The AR replicators use ingress replication instead of AR to forward multicast source data to the RNVE devices.

In this chapter, we show configuration and verification for a small subset of the scaled environment in which we validate OISM and AR together. Although the scaled test environment includes more devices, configured elements, multicast sources, and subscribed receivers, in this example we show configuration and verification output for the following elements:

- One EVPN instance, MACVRF-1, which is a MAC-VRF instance with VLAN-aware service type and VXLAN encapsulation.

- Multicast stream use cases that encompass:

- IGMPv2 or IGMPv3 traffic.
- Internal or external multicast sources.

- Two tenant VRF instances, one for IGMPv3 receivers and one for IGMPv2 receivers.

  For each tenant VRF instance, we define:

  - Four tenant VLANs with VXLAN tunnel network identifier (VNI) mappings, and corresponding IRB interfaces in the tenant VRF instance.

    In the OISM design, we refer to the tenant VLANs as *revenue bridge domains* or *revenue VLANs*.

  - One SBD VLAN mapped to a VNI, and a corresponding IRB interface in the tenant VRF instance.

- One multicast source inside the data center, and one multicast source outside the data center in the external PIM domain.

  You configure the BL devices to act as PIM EVPN gateway (PEG) devices for the EVPN fabric. In this example, we connect PEG devices through classic L3 interfaces to an external PIM router and PIM rendezvous point (RP). The L3 interfaces on each of the BL PEG devices link to the external PIM router on different subnets.

- Multicast receivers that subscribe to one or more multicast groups.

**NOTE**: Each multicast stream has multiple receivers subscribed to traffic from each source. The multicast traffic verification commands in this example focus on the first receiver device in the Receivers column in Table 18 on page 514.

See Table 18 on page 514 for a summary of these elements and their values. Figure 89 on page 511 illustrates the device roles and the first two corresponding IRB interfaces, VLANs, and VNI mappings for each of the tenant VRFs in the table.

**Table 18: OISM Streams and Elements in this Example**

| Multicast Stream | Tenant VRF | VLANs, IRB Interfaces, and VNI Mappings | | Source | Receivers | Multicast Groups |
|---|---|---|---|---|---|---|
| Internal source, internal receivers with IGMPv3 | VRF-1 | VLAN-1, irb.1 | VNI 110001 | TOR-1 on VLAN-1 (Multihomed to SL-1 and SL-2) | TOR-4 (Multihomed to SL-4 and SL-5)<br><br>Other receivers: | 233.252.0.21 through 233.252.0.23<br><br>233.252.0.121 through |
| | | VLAN-2, irb.2 | VNI 110002 | | | |

**Table 18: OISM Streams and Elements in this Example *(Continued)***

| Multicast Stream | Tenant VRF | VLANs, IRB Interfaces, and VNI Mappings | | Source | Receivers | Multicast Groups |
|---|---|---|---|---|---|---|
| —SSM reports only | | VLAN-3, irb.3 | VNI 110003 | | TOR-2 (Single-homed to SL-3)<br><br>TOR-3 (Multihomed to SL-4 and SL-5)<br><br>TOR-5 (Single-homed to SL-6) | 233.252.0.123 |
| | | VLAN-4, irb.4 | VNI 110004 | | | |
| | | (SBD) VLAN-2001, irb.2001 | VNI 992001 | | | |
| External source, internal receivers with IGMPv2—ASM reports only | VRF-101 | VLAN-401, irb.401 | VNI 110401 | External Source (In External PIM Domain) | TOR-1 on VLAN-1 (Multihomed to SL-1 and SL2)<br><br>Other receivers:<br><br>TOR-2 (Single-homed to SL-3)<br><br>TOR-3 (Multihomed to SL-4 and SL-5)<br><br>TOR-4 (Multihomed to SL-4 and SL-5)<br><br>TOR-5 (Single- | 233.252.0.1 through 233.252.0.3<br><br>233.252.0.101 through 233.252.0.103 |
| | | VLAN-402, irb.402 | VNI 110402 | | | |
| | | VLAN-403, irb.403 | VNI 110403 | | | |
| | | VLAN-404, irb.404 | VNI 110404 | | | |

**Table 18: OISM Streams and Elements in this Example *(Continued)***

| Multicast Stream | Tenant VRF | VLANs, IRB Interfaces, and VNI Mappings | | Source | Receivers | Multicast Groups |
|---|---|---|---|---|---|---|
| | | (SBD) VLAN-2101, irb.2101 | VNI 992101 | | homed to SL-6) | |

See Table 19 on page 516 for a summary of the BL device and external PIM router L3 connection parameters. In this example, the BL devices both use aggregated Ethernet (AE) interface ae3 for the external L3 connection, with different subnets per BL device. In the scaled-out test environment, the configuration uses a range of logical units on the ae3 interface with corresponding VLANs per tenant VRF, starting with unit 0 and VLAN-3001 for VRF-1. We focus on tenant VRF instances VRF-1 and VRF-101 in this example.

**Table 19: External Multicast L3 Interface Parameters**

| BL Device | Tenant VRF Instance | External L3 Interface Logical Unit | Associated VLAN | BL L3 Logical Interface IP Address | PIM Router Logical Interface and IP Address | PIM RP Logical Unit and IP Address |
|---|---|---|---|---|---|---|
| BL-1 | VRF-1 | unit 0: ae3.0 | VLAN-3001 | 172.30.0.1 | ae1.0: 172.30.0.0 | lo0.1: 172.22.2.1 |
| | VRF-101 | unit 100: ae3.100 | VLAN-3101 | 172.30.100.1 | ae1.100: 172.30.100.0 | lo0.101: 172.22.102.1 |
| BL-2 | VRF-1 | unit 0: ae3.0 | VLAN-3001 | 172.31.0.1 | ae2.0: 172.31.0.0 | lo0.1: 172.22.2.1 |
| | VRF-101 | unit 100: ae3.100 | VLAN-3101 | 172.31.100.1 | ae2.100: 172.31.100.0 | lo0.101: 172.22.102.1 |

You configure these parameters in "Configure the Border Leaf Devices for External Multicast Connectivity, PIM EVPN Gateway Role, and PIM Options" on page 534.

We divide the configuration into several sections.

## Configure the Underlay (with EBGP) and the Overlay (with IBGP)

We use eBGP for the underlay and iBGP for the overlay following the reference architectures in "IP Fabric Underlay Network Design and Implementation" on page 80 and "Configure IBGP for the Overlay" on page 96.

This example uses AE interfaces with one or two member links each for all of the connections for redundancy.

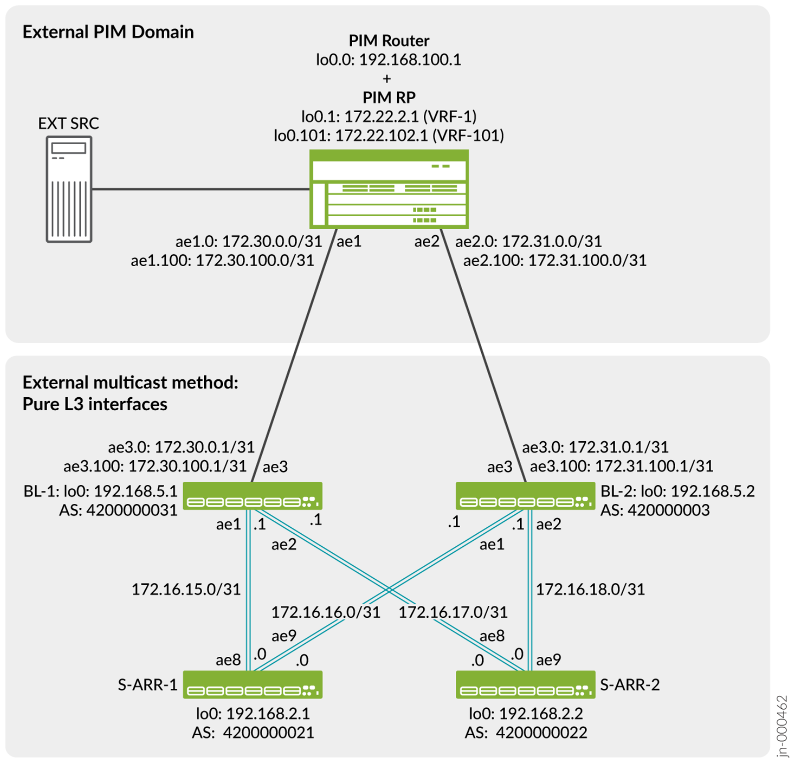
- Figure 90 on page 518 shows the AE interface IP addresses for the links between the S-ARR devices and the BL devices.

  **NOTE**: You configure the L3 interfaces from the BL devices to the external PIM router in "Configure the Border Leaf Devices for External Multicast Connectivity, PIM EVPN Gateway Role, and PIM Options" on page 534

- Figure 91 on page 519 shows the AE interface IP addresses for the links between the S-ARR devices and the SL devices.

  **NOTE**: You configure the links from the SL devices to the TOR devices in "Configure Server Leaf to TOR Interfaces and Ethernet Segment Identifiers (ESIs) for EVPN Multihoming" on page 529

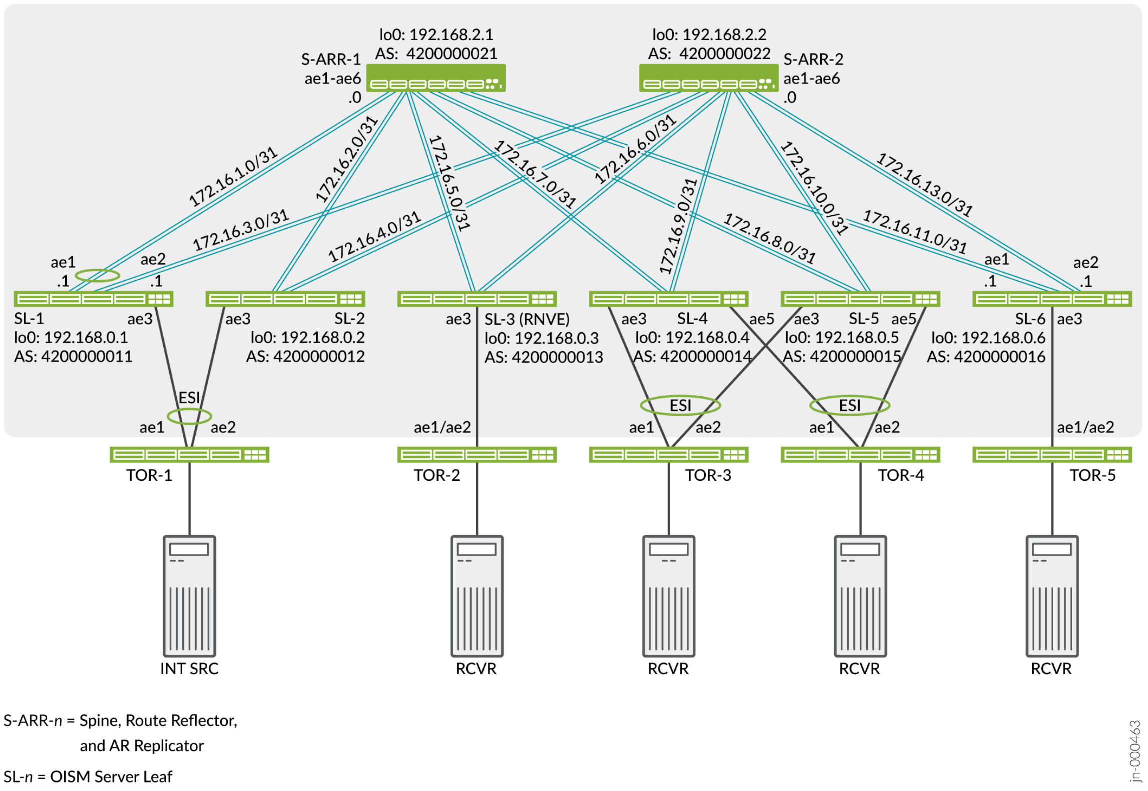
**Figure 90: OISM BL Device Links to S-ARR Devices and External PIM Router**



S-ARR-*n* = Spine, Route Reflector,
and AR Replicator

BL-n = OISM Border Leaf

**Figure 91: OISM SL Device Links to S-ARR Devices and TOR Devices**



1. Configure the approximate number of aggregated Ethernet (AE) interfaces in the configuration.

   For example:

   *All SL and BL Devices*:

   ```
   set chassis aggregated-devices ethernet device-count 10
   ```

   *S-ARR Devices*:

   ```
   set chassis aggregated-devices ethernet device-count 20
   ```

2. Follow the procedure in "Configuring the Aggregated Ethernet Interfaces Connecting Spine Devices to Leaf Devices" on page 83 to configure the AE interfaces for the underlay on all the BL, SL, and S-ARR devices.

   See Figure 90 on page 518 and Figure 91 on page 519 for the device IP addresses, AS numbers, and the AE interface names and IP addresses in this example.

3. Configure the eBGP underlay on the BL and SL devices. On each BL and SL device, set S-ARR-1 (AS: 4200000021) and S-ARR-2 (AS: 4200000022) as BGP neighbors. Similarly, on each S-ARR device, set each of the BL and SL devices as BGP neighbors. See "IP Fabric Underlay Network Design and Implementation" on page 80 for details.

   See Figure 90 on page 518 and Figure 91 on page 519 for the device IP addresses, AS numbers, and the AE interface names and IP addresses in this example.

   For example:

   *SL-1 (device AS: 4200000011)*:

```
set policy-options policy-statement underlay-clos-export term loopback from interface lo0.0
set policy-options policy-statement underlay-clos-export term loopback then accept
set protocols bgp group underlay-bgp type external
set protocols bgp group underlay-bgp export underlay-clos-export
set protocols bgp group underlay-bgp local-as 4200000011
set protocols bgp group underlay-bgp multipath multiple-as
set protocols bgp group underlay-bgp bfd-liveness-detection minimum-interval 1200
set protocols bgp group underlay-bgp bfd-liveness-detection multiplier 3
set protocols bgp group underlay-bgp bfd-liveness-detection session-mode automatic
set protocols bgp group underlay-bgp neighbor 172.16.1.0 peer-as 4200000021
set protocols bgp group underlay-bgp neighbor 172.16.3.0 peer-as 4200000022
set protocols bgp log-updown
```

4. Configure the iBGP overlay with an overlay AS number and EVPN signaling on the SL, BL, and S-ARR devices. See "Configure IBGP for the Overlay" on page 96 for details.

   On each SL and BL device, set the spine devices S-ARR-1 (lo0:192.168.2.1) and S-ARR-2 (192.168.2.2) in the overlay AS as BGP neighbors. For example:

   *SL-1 (device lo0: 192.168.0.1)*:

```
set routing-options autonomous-system 4210000001
set protocols bgp group cluster-rr type internal
set protocols bgp group cluster-rr local-address 192.168.0.1
set protocols bgp group cluster-rr mtu-discovery
set protocols bgp group cluster-rr family evpn signaling delay-route-advertisements minimum-
delay routing-uptime 400
set protocols bgp group cluster-rr multipath
set protocols bgp group cluster-rr bfd-liveness-detection minimum-interval 4000
set protocols bgp group cluster-rr bfd-liveness-detection multiplier 3
set protocols bgp group cluster-rr bfd-liveness-detection session-mode automatic
set protocols bgp group cluster-rr neighbor 192.168.2.1
set protocols bgp group cluster-rr neighbor 192.168.2.2
set protocols bgp group cluster-rr vpn-apply-export
```

Repeat this step for each SL and BL device, substituting that device loopback IP address for the `local-address` option. See Figure 90 on page 518 and Figure 91 on page 519.

On each S-ARR device, set all of the SL and BL devices in the overlay AS as BGP neighbors. For example:

*S-ARR-1 (device lo0: 192.168.2.1)*:

```
set routing-options autonomous-system 4210000001
set protocols bgp group cluster-rr type internal
set protocols bgp group cluster-rr local-address 192.168.2.1
set protocols bgp group cluster-rr mtu-discovery
set protocols bgp group cluster-rr family evpn signaling delay-route-advertisements minimum-
delay routing-uptime 400
set protocols bgp group cluster-rr cluster 192.168.2.1
set protocols bgp group cluster-rr multipath
set protocols bgp group cluster-rr bfd-liveness-detection minimum-interval 4000
set protocols bgp group cluster-rr bfd-liveness-detection multiplier 3
set protocols bgp group cluster-rr bfd-liveness-detection session-mode automatic
set protocols bgp group cluster-rr neighbor 192.168.5.1
set protocols bgp group cluster-rr neighbor 192.168.5.2
set protocols bgp group cluster-rr neighbor 192.168.0.1
set protocols bgp group cluster-rr neighbor 192.168.0.2
set protocols bgp group cluster-rr neighbor 192.168.0.3
set protocols bgp group cluster-rr neighbor 192.168.0.4
set protocols bgp group cluster-rr neighbor 192.168.0.5
set protocols bgp group cluster-rr neighbor 192.168.0.6
set protocols bgp group cluster-rr vpn-apply-export
```

5. Set MAC aging, Address Resolution Protocol (ARP), and Neighbor Discovery Protocol (NDP) timer values to support MAC address learning and IPv4 and IPv6 route learning. These parameters are common settings in EVPN-VXLAN fabrics and are not specific to the OISM or AR feature configurations.

Set these values on all SL and BL devices.

```
set protocols l2-learning global-mac-ip-table-aging-time 3600
set protocols l2-learning global-mac-table-aging-time 4200
set system arp aging-timer 60
```

Set the NDP stale timer on all IRB interfaces for IPv6 neighbor reachability confirmation on all SL and BL devices for IPv6 unicast traffic.

**NOTE**: You apply this setting to all IRB interfaces that might handle IPv6 unicast traffic, so you usually add this statement to a configuration group of common statements and apply the group to all interfaces.

```
set interfaces irb unit unit-number family inet6 nd6-stale-time 3600
```

## Configure an OISM-Enabled EVPN MAC-VRF Instance

The scaled test environment includes multiple MAC-VRF EVPN instances. We show one instance called MACVRF-1 here that we use for OISM and AR traffic.

**NOTE**: We require that you enable the shared tunnels feature on the QFX5000 line of switches running Junos OS with a MAC-VRF instance configuration. This feature prevents problems with VTEP scaling on the device when the configuration uses multiple MAC-VRF instances. When you configure shared tunnels, the device minimizes the number of next-hop entries to reach remote VTEPs. You globally enable shared VXLAN tunnels on the device using the `shared-tunnels` statement at the `[edit forwarding-options evpn-vxlan]` hierarchy level. You must reboot the device for this setting to take effect.

This statement is optional on the QFX10000 line of switches running Junos OS, which can handle higher VTEP scaling than the QFX5000 line of switches. On devices running Junos OS Evolved in EVPN-VXLAN fabrics, shared tunnels are enabled by default.

Configure the elements in these steps on all SL, BL, and S-ARR devices.

**NOTE**: This example includes the AR multicast optimization with OISM. The spine devices (S-ARR-1 and S-ARR-2) in the fabric serve as standalone AR replicator devices.

For AR to work with the regular OISM symmetric bridge domains model, you must also configure all the common OISM SL and BL elements on the standalone AR replicator devices, such as the MAC-VRF instance, VLANs, tenant VRFs, and IRB interfaces.

If the spine devices don’t run as AR replicators, you don’t need to configure these elements on the spine devices.

1. Configure MACVRF-1, an EVPN MAC-VRF instance with the VLAN-aware service type. Specify the VXLAN tunnel endpoint (VTEP) source interface as the device loopback interface. Set a route distinguisher and route target (with automatic route target derivation) for the instance. Also, enable all VNIs in the instance to extend into the EVPN BGP domain.

   For example:

   *SL-1*:

   ```
   set routing-instances MACVRF-1 instance-type mac-vrf
   set routing-instances MACVRF-1 protocols evpn encapsulation vxlan
   set routing-instances MACVRF-1 protocols evpn default-gateway no-gateway-community
   set routing-instances MACVRF-1 protocols evpn extended-vni-list all
   set routing-instances MACVRF-1 vtep-source-interface lo0.0
   set routing-instances MACVRF-1 service-type vlan-aware
   set routing-instances MACVRF-1 route-distinguisher 192.168.0.1:33301
   set routing-instances MACVRF-1 vrf-target target:33300:1
   set routing-instances MACVRF-1 vrf-target auto
   ```

   Repeat this step on the remaining SL devices, BL-1, BL-2, S-ARR-1, and S-ARR-2. In the configuration on each device, substitute that device’s IP address as part of the `route-distinguisher` statement value so these values are unique across the devices.

2. Configure platform-specific settings required in scaled EVPN-VXLAN environments. You configure these options on the devices in the fabric that operate in any OISM or AR role.

   a. On Junos OS devices in the QFX5000 line of switches , include the global shared tunnels configuration statement. This setting supports VXLAN tunneling with MAC-VRF instances on those devices in scaled environments:

      ```
      set forwarding-options evpn-vxlan shared-tunnels
      ```

**NOTE**: If you need to configure the shared tunnels setting here, after you commit the configuration, you must reboot the device for the shared tunnels setting to take affect.

The shared tunnels feature is set by default on devices that run Junos OS Evolved.

b. On QFX5130 and QFX5700 switches, configure the `host-profile` unified forwarding profile option to support an EVPN-VXLAN environment (see *Layer 2 Forwarding Tables* for details):

```
set system packet-forwarding-options forwarding-profile host-profile
```

c. On QFX5110 and QFX5120 switches, configure next hop settings for scaled environments to ensure next hops table size is proportional to the device capacity, as follows:

*QFX5110 switches:*

```
set forwarding-options vxlan-routing next-hop 32768
set forwarding-options vxlan-routing interface-num 8192
```

*QFX5120 switches:*

```
set forwarding-options vxlan-routing next-hop 45056
set forwarding-options vxlan-routing interface-num 8192
```

3. Configure the OISM revenue VLANs and SBD VLANs in the EVPN MACVRF-1 instance. Map each VLAN to a VXLAN VNI value. Also, create IRB interfaces for each VLAN. See Table 18 on page 514 for the VLANs and VNI values we use in this example. You configure one SBD and corresponding IRB for each VRF.

   Configure these elements on all SL, BL, and S-ARR devices.

```
set routing-instances MACVRF-1 vlans VLAN-1 vlan-id 1
set routing-instances MACVRF-1 vlans VLAN-1 l3-interface irb.1
set routing-instances MACVRF-1 vlans VLAN-1 vxlan vni 110001
set routing-instances MACVRF-1 vlans VLAN-2 vlan-id 2
set routing-instances MACVRF-1 vlans VLAN-2 l3-interface irb.2
set routing-instances MACVRF-1 vlans VLAN-2 vxlan vni 110002
set routing-instances MACVRF-1 vlans VLAN-3 vlan-id 3
set routing-instances MACVRF-1 vlans VLAN-3 l3-interface irb.3
set routing-instances MACVRF-1 vlans VLAN-3 vxlan vni 110003
set routing-instances MACVRF-1 vlans VLAN-4 vlan-id 4
set routing-instances MACVRF-1 vlans VLAN-4 l3-interface irb.4
set routing-instances MACVRF-1 vlans VLAN-4 vxlan vni 110004

set routing-instances MACVRF-1 vlans VLAN-2001 vlan-id 2001
set routing-instances MACVRF-1 vlans VLAN-2001 l3-interface irb.2001
set routing-instances MACVRF-1 vlans VLAN-2001 vxlan vni 992001

set routing-instances MACVRF-1 vlans VLAN-401 vlan-id 401
set routing-instances MACVRF-1 vlans VLAN-401 l3-interface irb.401
set routing-instances MACVRF-1 vlans VLAN-401 vxlan vni 110401
set routing-instances MACVRF-1 vlans VLAN-402 vlan-id 402
set routing-instances MACVRF-1 vlans VLAN-402 l3-interface irb.402
set routing-instances MACVRF-1 vlans VLAN-402 vxlan vni 110402
set routing-instances MACVRF-1 vlans VLAN-403 vlan-id 403
set routing-instances MACVRF-1 vlans VLAN-403 l3-interface irb.403
set routing-instances MACVRF-1 vlans VLAN-404 vxlan vni 110403
set routing-instances MACVRF-1 vlans VLAN-404 vlan-id 404
set routing-instances MACVRF-1 vlans VLAN-404 l3-interface irb.404
set routing-instances MACVRF-1 vlans VLAN-404 vxlan vni 110404

set routing-instances MACVRF-1 vlans VLAN-2101 vlan-id 2101
set routing-instances MACVRF-1 vlans VLAN-2101 l3-interface irb.2101
set routing-instances MACVRF-1 vlans VLAN-2101 vxlan vni 992101
```

4. Enable OISM globally on all SL, BL, and S-ARR devices.

   **NOTE**: The S-ARR spine devices in this example serve as standalone AR replicator devices, so you must enable OISM on them too. If the spine devices don't run as AR replicators, you don't need to enable OISM on those devices.

   ```
   set forwarding-options multicast-replication evpn irb oism
   ```

5. Enable IGMP snooping in the MAC-VRF instance for all OISM tenant (revenue) VLANs and SBD VLANs on all SL, BL, and S-ARR devices.

   **NOTE**: You enable IGMP snooping on the S-ARR devices because they act as AR replicator devices. AR replicators use IGMP snooping to optimize traffic forwarding.

In EVPN-VXLAN fabrics, we support IGMPv2 traffic with ASM reports only. We support IGMPv3 traffic with SSM reports only. When you enable IGMP snooping for IGMPv3 traffic, include the SSM-specific option evpn-ssm-reports-only configuration option as shown below. See Supported IGMP or MLD Versions and Group Membership Report Modes for more on ASM and SSM support with EVPN-VXLAN.

*IGMP snooping with IGMPv2 on all VLANs*:

```
set routing-instances MACVRF-1 protocols igmp-snooping vlan all proxy
```

*IGMP snooping on VLANs with IGMPv3 receivers (VLAN-1 through VLAN-4, according to* Table 18 on page 514):

```
set routing-instances MACVRF-1 protocols igmp-snooping vlan VLAN-1 evpn-ssm-reports-only
set routing-instances MACVRF-1 protocols igmp-snooping vlan VLAN-2 evpn-ssm-reports-only
set routing-instances MACVRF-1 protocols igmp-snooping vlan VLAN-3 evpn-ssm-reports-only
set routing-instances MACVRF-1 protocols igmp-snooping vlan VLAN-4 evpn-ssm-reports-only
```

6. Enable IGMP on the IRB interfaces that handle multicast traffic on the SL, BL, and S-ARR devices.

   IGMPv2 is enabled by default when you configure PIM, but you need to explicitly enable IGMPv3 on the IRB interfaces that handle IGMPv3 traffic. According to Table 18 on page 514, the IGMPv3 IRB interfaces are irb.1, irb.2, irb.3, and irb.4, which are associated with VRF-1:

```
set protocols igmp interface irb.1 version 3
set protocols igmp interface irb.2 version 3
set protocols igmp interface irb.3 version 3
set protocols igmp interface irb.4 version 3
```

7. (Required on any devices configured as AR replicators with OISM, and on the QFX10000 line of switches and the PTX10000 line of routers in any OISM role only in releases prior to Junos OS or Junos OS Evolved Release 23.4R1) To avoid traffic loss at the onset of multicast flows, set the `install-star-g-routes` option at the `[edit routing-instances` *`name`* `multicast-snooping-options oism]` hierarchy. You set this parameter in the MAC-VRF instance. With this option, the Routing Engine installs (*,G) multicast routes on the Packet Forwarding Engine for all of the OISM revenue VLANs in the routing instance immediately upon learning about any interested receiver. See *Latency and Scaling Trade-Offs for Installing Multicast Routes with OISM (install-star-g-routes Option)* for more on using this option.

   For example, in this test environment, S-ARR-1 and S-ARR-2 are AR replicators, so we configure this statement in the MAC-VRF routing instance on those devices. Also, SL-4 and SL-5 are switches in the

QFX10000 line switches, so if those devices are running earlier releases than Junos OS or Junos OS Evolved Release 23.4R1, you would also configure this statement on those devices.

```
set routing-instances MACVRF-1 multicast-snooping-options oism install-star-g-routes
```

## Configure the Tenant VRF Instances for IGMPv2 and IGMPv3 Multicast Receivers

The scaled test environment includes many tenant L3 VRF instances. We show the VRF instances for the two multicast use cases in Table 18 on page 514:

- VRF-1 for IGMPv3 traffic pulled from an internal source.
- VRF-101 for IGMPv2 traffic pulled from an external source.

Configure the elements in these steps in those VRF instances on all SL, BL, and S-ARR devices.

**NOTE**: The S-ARR spine devices in this example also serve as standalone AR replicator devices, so you must configure all the tenant VRF settings on them too. If the spine devices don't run as AR replicators, you don't need to include these steps on those devices.

You also configure different PIM options in the tenant VRF instances for SL devices compared to BL devices. See "Configure OSPF and PIM on the Server Leaf Devices" on page 533 and "Configure the Border Leaf Devices for External Multicast Connectivity, PIM EVPN Gateway Role, and PIM Options" on page 534 for those configuration steps. You don't need to configure PIM on the S-ARR devices.

1. Configure the tenant VRF instances.

   Include the IRB interfaces associated with each tenant VRF instance. Set a route distinguisher based on the device IP address, and a route target for the instance.

   For example:

   *SL-1*:

```
set routing-instances VRF-1 instance-type vrf
set routing-instances VRF-1 route-distinguisher 192.168.0.1:1
set routing-instances VRF-1 vrf-target target:100:1

set routing-instances VRF-101 instance-type vrf
set routing-instances VRF-101 route-distinguisher 192.168.0.1:101
set routing-instances VRF-101 vrf-target target:100:101
```

Repeat this step on the remaining SL devices, BL-1, BL-2, S-ARR-1, and S-ARR-2. On each device, configure the `route-distinguisher` to be unique across the devices and tenant VRFs. Substitute the device IP address from Figure 90 on page 518 or Figure 91 on page 519, and the VRF instance number for each tenant VRF, as follows:

- On SL-2, use `route-distinguisher` 192.168.0.2:1 for VRF-1, and 192.168.0.2:101 for VRF-101.

  On SL-3, use `route-distinguisher` 192.168.0.3:1 for VRF-1, and 192.168.0.3:101 for VRF-101.

  And so on for the remaining SL devices.

- On BL-1, use `route-distinguisher` 192.168.5.1:1 for VRF-1, and 192.168.5.1:101 for VRF-101.

  On BL-2, use `route-distinguisher` 192.168.5.2:1 for VRF-1, and 192.168.5.2:101 for VRF-101.

- On S-ARR-1, use `route-distinguisher` 192.168.2.1:1 for VRF-1, and 192.168.2.1:101 for VRF-101.

  On S-ARR-2, use `route-distinguisher` 192.168.2.2:1 for VRF-1, and 192.168.2.2:101 for VRF-101.

2. In each VRF instance, include the corresponding IRB interfaces for the OISM revenue VLANs and the SBD associated with that instance, according to Table 18 on page 514:

```
set routing-instances VRF-1 interface irb.1
set routing-instances VRF-1 interface irb.2
set routing-instances VRF-1 interface irb.3
set routing-instances VRF-1 interface irb.4
set routing-instances VRF-1 interface irb.2001

set routing-instances VRF-101 interface irb.401
set routing-instances VRF-101 interface irb.402
set routing-instances VRF-101 interface irb.403
set routing-instances VRF-101 interface irb.404
set routing-instances VRF-101 interface irb.2101
```

   Repeat the same configuration all SL, BL, and S-ARR devices.

3. Identify the IRB interface for the OISM SBD associated with each tenant VRF instance, as outlined in Table 18 on page 514:

```
set routing-instances VRF-1 protocols evpn oism supplemental-bridge-domain-irb irb.2001
set routing-instances VRF-101 protocols evpn oism supplemental-bridge-domain-irb irb.2101
```

Repeat the same configuration all SL, BL, and S-ARR devices.

4. On any SL, BL, and S-ARR devices that have a single Routing Engine, configure the graceful restart feature for each tenant VRF:

```
set routing-instances VRF-1 routing-options graceful-restart
set routing-instances VRF-101 routing-options graceful-restart
```

## Configure Server Leaf to TOR Interfaces and Ethernet Segment Identifiers (ESIs) for EVPN Multihoming

The TOR devices host the multicast sources and receivers inside the fabric. These devices have single-homed or multihomed connections to the SL devices in the EVPN core. See Figure 91 on page 519 for the topology in this example. TOR-1, TOR-3, and TOR-4 are each multihomed to two SL devices, and TOR-2 and TOR-5 are single-homed. Figure 91 on page 519 shows that:

- The SL devices all use interface ae3 to connect to the TOR devices.
- SL4 and SL-5 use interfaces ae3 and ae5 for redundant connections to TOR-3 and TOR-4, respectively.
- Each multihomed TOR device uses interfaces ae1 and ae2 to connect to its peer SL devices.
- For consistency in the configuration, the single-homed TORs (TOR-2 and TOR-5) also use interfaces ae1 and ae2 but as redundant links to a single SL device.

Also, starting in Junos OS and Junos OS Evolved Release 23.2R2, you can additionally configure the network isolation feature on the multihomed TOR-facing interfaces on the SL devices to help mitigate traffic loss during core isolation events on those interfaces. In this example, Step 4 shows how to configure the network isolation feature on the interfaces from SL-1 and SL-2 to multihomed device TOR-1.

**NOTE**: Although this example doesn't show any multihomed server-facing or TOR-facing interfaces on the BL devices, you can similarly configure the network isolation feature for any such interfaces on BL devices.

1. Configure the server leaf to TOR interfaces and Ethernet segment identifiers (ESIs) for EVPN multihoming.

   On multihoming peer SL devices, configure the same ESI on the AE interface links to the multihomed TOR device. Also enable LACP and LACP hold timers on the interfaces. See "Multihoming an

Ethernet-Connected End System Design and Implementation" on page 199 for more details on configuring these interfaces and the corresponding Ethernet segment identifiers (ESIs).

**NOTE**: See Step 4 in this section to add configuring the device to detect and take action upon network isolation events on these interfaces. If you include that configuration, make sure the "up" hold timer configured here in this step is greater than the "up" hold timer in the network isolation group profile in that step.

For example:

*On SL-1 for the multihoming interfaces to TOR-1*:

```
set interfaces xe-0/0/6:0 ether-options 802.3ad ae3
set interfaces xe-0/0/6:0 hold-time up 300000
set interfaces xe-0/0/6:0 hold-time down 1000
set interfaces ae3 esi 00:00:00:ff:00:01:00:01:00:3
set interfaces ae3 esi all-active
set interfaces ae3 aggregated-ether-options lacp active
set interfaces ae3 aggregated-ether-options lacp periodic fast
set interfaces ae3 aggregated-ether-options lacp system-id 00:00:00:99:99:01
set interfaces ae3 aggregated-ether-options lacp hold-time up 300
```

Repeat this configuration on SL-2, the multihoming peer to SL-1. Use the corresponding interface for the link on SL-2 to TOR-1, and the same ESI.

*On SL-2 for the multihoming interfaces to TOR-1*:

```
set interfaces xe-0/0/12 ether-options 802.3ad ae3
set interfaces xe-0/0/12 hold-time up 300000
set interfaces xe-0/0/12 hold-time down 1000
set interfaces ae3 esi 00:00:00:ff:00:01:00:01:00:03
set interfaces ae3 esi all-active
set interfaces ae3 aggregated-ether-options lacp active
set interfaces ae3 aggregated-ether-options lacp periodic fast
set interfaces ae3 aggregated-ether-options lacp system-id 00:00:00:99:99:01
set interfaces ae3 aggregated-ether-options lacp hold-time up 300
```

2. Repeat the configuration in Step 1 above on the other SL devices that serve multihomed TOR devices.

   Use a different ESI for each corresponding multihoming peer SL device pair. See Figure 91 on page 519. In this example, all SL devices use ae3 to link to the TOR devices. SL-4 and SL-5 use ae3 to link to TOR-3, and additionally use ae5 to link to TOR-4. As a result, you set one ESI for the ae3 links and a different ESI for the ae5 links on both of those SL devices.

Here we assign ESI values with the following conventions:

- The 6th segment from the right in the ESI value matches the number of the multihomed TOR.
- The last segment of the ESI value matches the AE interface number.

For example:

- *On SL-4 and SL-5 for the interfaces to multihomed TOR-3 (link is ae3 on both SL devices)*:

  ```
  set interfaces ae3 esi 00:00:00:ff:00:03:00:01:00:03
  ```

- *On SL-4 and SL-5 for the interfaces to multihomed TOR-4 (link is ae5 on both SL devices)*:

  ```
  set interfaces ae3 esi 00:00:00:ff:00:04:00:01:00:05
  ```

TOR-2 is single-homed to SL-3, so you don’t configure an ESI on the ae3 interface on SL-3.

Similarly, TOR-5 is single-homed to SL-6, so you don’t configure an ESI on the ae3 interface on SL-6.

**NOTE**: The test environment also includes separate AE interfaces (ae4 on each SL device) and ESIs (for the multihomed TOR devices) that the fabric uses for unicast traffic. We only show configuring the multicast ESIs here.

3. Include the interfaces to the TOR devices in the MAC-VRF instance on each SL device—ae3 on all SL devices, and also ae5 on SL-4 and SL-5.
   For example:

   *On SL-1 for the interface to TOR-1 (and repeat this configuration on SL-2 for the interface to TOR-1, SL-3 for the interface to TOR-2, and SL-6 for the interface to TOR-5)*:

   ```
   set routing-instances MACVRF-1 interface ae3.0
   ```

   *On SL-4 and SL-5 for the interfaces to TOR-3 and TOR-4*:

   ```
   set routing-instances MACVRF-1 interface ae3.0
   set routing-instances MACVRF-1 interface ae5.0
   ```

4. (Optional in releases starting with Junos OS and Junos OS Evolved Release 23.2R2 with this OISM configuration) Configure the network isolation service-tracking feature on SL-1 and SL-2 for interface ae3 to TOR-1. With this feature, the device changes the interface status upon detecting and recovering from a core isolation event. See *Layer 2 Interface Status Tracking and Shutdown Actions for EVPN Core Isolation Conditions*, *network-isolation*, and *network-isolation-profile* for details on how this feature works. To enable this feature, we:

   - Set up a network isolation group `net-isolation-grp-1` with core isolation service tracking using the *network-isolation* statement.

   - Include an "up" hold time (in ms) that the device will wait upon detecting recovery from network isolation before acting to bring the interface up again.

     **NOTE**: The "up" hold time you configure here for core isolation event detection should be less than the general "up" hold timer on the interfaces for the ESI in Step 1. For example, in that step, the interface "up" hold timer is 300000 ms; here, we set the "up" hold timer for the network isolation group to 100000 ms.

   - Configure the device to set either `lacp-out-of-sync` or `link-down` status on the interface as the core isolation action. For example, here we set the `lacp-out-of-sync` action.

   - Apply the network isolation group to the interface using the *network-isolation-profile* statement.

   - Because core isolation service tracking improves the convergence time for the ESI routes, we don't need to include a delay in the overlay BGP route advertisements. As a result, on devices where we configure the network isolation feature on TOR-facing interfaces, we remove the `delay-route-advertisements` settings from Step 4 in "Configure the Underlay (with EBGP) and the Overlay (with IBGP)" on page 517.

   For example:

   *On SL-1 for the multihoming interface(s) to TOR-1*:

```
set protocols network-isolation group net-isolation-grp-1 detection hold-time up 100000
set protocols network-isolation group net-isolation-grp-1 detection service-tracking core-
isolation
set protocols network-isolation group net-isolation-grp-1 service-tracking-action lacp-out-of-
sync

set interfaces ae3 network-isolation-profile net-isolation-grp-1

delete protocols bgp group cluster-rr family evpn signaling delay-route-advertisements
```

Repeat this configuration on SL-2 for ae3 with the same network isolation group profile parameters.

You can set up the same or similar network isolation group profiles on the other multihoming peer SL devices, SL-4 and SL-5, which use ae3 for the ESI to TOR-3 and ae5 for the ESI to TOR-4.

Also, when we enable the network isolation feature on the SL device interfaces, with that configuration we remove the overlay route advertisement delay setting on the device. We still need to retain the overlay `delay-route-advertisements minimum-delay routing-uptime` setting on S-ARR-1 and S-ARR-2 from Step 4 in "Configure the Underlay (with EBGP) and the Overlay (with IBGP)" on page 517. However, with the network isolation feature configured in the network, we observed better results testing with a higher setting on the S-ARR devices than the setting in that step. As a result, in this case we recommend you also change the following:

*On S-ARR-1 and S-ARR-2 from Step* 4 *in* "Configure the Underlay (with EBGP) and the Overlay (with IBGP)" on page 517:

```
...
set protocols bgp group cluster-rr family evpn signaling delay-route-advertisements minimum-
delay routing-uptime 400
...
```

*change that configuration to:*

```
...
set protocols bgp group cluster-rr family evpn signaling delay-route-advertisements minimum-
delay routing-uptime 500
...
```

## Configure OSPF and PIM on the Server Leaf Devices

In this procedure, you configure the OISM elements specific to server leaf functions, such as PIM, in the tenant VRF instances in this example—VRF-1 and VRF-101. Configure these steps on all SL devices.

1. (Optional) Configure an OSPF area for each tenant VRF, but with all interfaces on the SL devices in OSPF passive mode so the devices can advertise routes but don't form OSPF adjacencies.

```
set routing-instances VRF-1 protocols ospf interface all passive

set routing-instances VRF-101 protocols ospf interface all passive
```

2. Configure PIM in passive mode on the SL devices for all interfaces in each of the VRF routing instances.

```
set routing-instances VRF-1 protocols pim passive
set routing-instances VRF-1 protocols pim interface all

set routing-instances VRF-101 protocols pim passive
set routing-instances VRF-101 protocols pim interface all
```

3. Set the PIM `accept-remote-source` option to enable the SL devices to accept multicast traffic from the SBD IRB interface as the source interface. With the regular OISM symmetric bridge domains model, any multicast traffic coming from external sources arrives at the SL devices on the SBD.

```
set routing-instances VRF-1 protocols pim interface irb.2001 accept-remote-source
set routing-instances VRF-101 protocols pim interface irb.2101 accept-remote-source
```

## Configure the Border Leaf Devices for External Multicast Connectivity, PIM EVPN Gateway Role, and PIM Options

In this procedure, you configure the OISM elements specific to border leaf functions, including the steps to connect to the external PIM domain. Configure statements in this procedure on each BL device.

**NOTE**: This example connects to the external PIM router using classic L3 interface links. OISM supports additional methods to connect to the external domain depending on the platform of the BL device. See External Multicast Connection Methods in the EVPN User Guide for a list of supported external multicast methods per platform.

1. Configure the L3 interfaces that connect to the external PIM router.

   In this example, both BL devices use interface ae3 for this purpose with a different subnet for each BL device connection per tenant VRF. The physical interfaces in the AE interface bundle might differ across the BL devices. Figure 90 on page 518 shows the BL device and link information for this example.

   The external L3 multicast connection uses the ae3 interface with VLAN tagging enabled. In the scaled environment, we configure logical units on the ae3 interface and corresponding VLANs per tenant VRF starting with unit 0 and VLAN-3001 for VRF-1. In this example we show configuring ae3.0 in VRF-1 and ae3.100 in VRF-101 on both BL devices. See Table 19 on page 516 for a summary of the external L3 interface connection configuration parameters.

*BL-1*:

```
set interfaces et-0/0/0:0 ether-options 802.3ad ae3
set interfaces ae3 vlan-tagging
set interfaces ae3 mtu 9192
set interfaces ae3 aggregated-ether-options minimum-links 1
set interfaces ae3 aggregated-ether-options lacp active
set interfaces ae3 aggregated-ether-options lacp periodic fast

set interfaces ae3 unit 0 vlan-id 3001
set interfaces ae3 unit 0 family inet address 172.30.0.1/31 preferred
set interfaces ae3 unit 100 vlan-id 3101
set interfaces ae3 unit 100 family inet address 172.30.100.1/31 preferred
```

*BL-2*:

```
set interfaces xe-0/0/14:0 ether-options 802.3ad ae3
set interfaces ae3 vlan-tagging
set interfaces ae3 mtu 9192
set interfaces ae3 aggregated-ether-options minimum-links 1
set interfaces ae3 aggregated-ether-options lacp active
set interfaces ae3 aggregated-ether-options lacp periodic fast

set interfaces ae3 unit 0 vlan-id 3001
set interfaces ae3 unit 0 family inet address 172.31.0.1/31 preferred
set interfaces ae3 unit 100 vlan-id 3101
set interfaces ae3 unit 100 family inet address 172.31.100.1/31 preferred
```

2. Include the logical L3 interfaces in the tenant VRF instances. Both BL devices use ae3 for the external multicast connection, so use the same configuration on both devices.

   *BL-1 and BL-2*:

```
set routing-instances VRF-1 interface ae3.0
set routing-instances VRF-101 interface ae3.100
```

3. Configure each of the BL devices with the OISM PEG role in each tenant VRF.

*BL-1 and BL-2*:

```
set routing-instances VRF-1 protocols evpn oism pim-evpn-gateway
set routing-instances VRF-101 protocols evpn oism pim-evpn-gateway
```

4. Configure PIM in the tenant VRF instances, including the following for each tenant VRF in this example:

   - In this environment, set a static PIM RP address corresponding to the external PIM router and RP router (one for each logical unit and associated tenant VRF).

   - Configure PIM on the OISM revenue VLAN IRB interfaces and include the PIM `distributed-dr` option.

   - Configure classic PIM (no `distributed-dr` option) on the SBD IRB interface, the logical unit L3 interface and logical loopback interface.

     With PIM on the SBD IRB interface, include the `accept-remote-source` option to enable the BL devices to accept multicast traffic from the SBD IRB interface as the source interface. This option handles situations where the BL devices might send source traffic to each other on the SBD. See Multicast Traffic from an External Source to Receivers Inside the EVPN Data Center—L3 Interface Method or Non-EVPN IRB Method in the EVPN User Guide for more information on those situations.

   - With PIM on the SBD IRB interface, enable Bidirectional Forwarding Detection (BFD) and the *stickydr* option. The BFD settings improve convergence time with interface issues to help avoid traffic loss. The `stickydr` option eliminates designated router switchover convergence delay during reboot events.

   Include the same configuration on BL-1 and BL-2.

   *For VRF-1*:

```
set routing-instances VRF-1 protocols pim interface ae3.0
set routing-instances VRF-1 protocols pim rp static address 172.22.2.1
set routing-instances VRF-1 protocols pim interface irb.1 distributed-dr
set routing-instances VRF-1 protocols pim interface irb.2 distributed-dr
set routing-instances VRF-1 protocols pim interface irb.3 distributed-dr
set routing-instances VRF-1 protocols pim interface irb.4 distributed-dr
set routing-instances VRF-1 protocols pim interface lo0.1
set routing-instances VRF-1 protocols pim interface irb.2001 accept-remote-source
set routing-instances VRF-1 protocols pim interface irb.2001 family inet bfd-liveness-
detection minimum-interval 1000
set routing-instances VRF-1 protocols pim interface irb.2001 family inet bfd-liveness-
```

```
detection multiplier 3
set routing-instances VRF-1 protocols pim interface irb.2001 stickydr
```

*For VRF-101*:

```
set routing-instances VRF-101 protocols pim interface ae3.100
set routing-instances VRF-101 protocols pim rp static address 172.22.102.1
set routing-instances VRF-101 protocols pim interface irb.401 distributed-dr
set routing-instances VRF-101 protocols pim interface irb.402 distributed-dr
set routing-instances VRF-101 protocols pim interface irb.403 distributed-dr
set routing-instances VRF-101 protocols pim interface irb.404 distributed-dr
set routing-instances VRF-101 protocols pim interface lo0.101
set routing-instances VRF-101 protocols pim interface irb.2101 accept-remote-source
set routing-instances VRF-101 protocols pim interface irb.2101 family inet bfd-liveness-
detection minimum-interval 1000
set routing-instances VRF-101 protocols pim interface irb.2101 family inet bfd-liveness-
detection multiplier 3
set routing-instances VRF-1 protocols pim interface irb.2101 stickydr
```

See OISM Components in the EVPN User Guide for details on why we configure these PIM options on the BL devices.

5. Configure an OSPF area for each tenant VRF instance that includes the external multicast logical L3 interface and the SBD IRB interface, both in OSPF active mode. This step establishes an OSPF routing domain with these interfaces as neighbors in the tenant VRF to support routing among them. Include the other interfaces on the device in the OSPF area but in OSPF passive mode, so they can advertise routes but don't form OSPF adjacencies.

   You also define and include the `export-direct` export policy, which exports addresses of the directly-connected IRB interfaces on a particular BL device into the OSPF routing protocol.

   The configuration is the same on both BL devices.

   *BL-1 and BL-2*:

```
set policy-options policy-statement export-direct term t1 from protocol direct
set policy-options policy-statement export-direct term t1 then accept
set routing-instances VRF-1 protocols ospf area 1.1.1.1 interface ae3.0
set routing-instances VRF-1 protocols ospf area 1.1.1.1 interface irb.2001
set routing-instances VRF-1 protocols ospf area 1.1.1.1 interface all passive
set routing-instances VRF-1 protocols ospf export export-direct

set routing-instances VRF-101 protocols ospf area 1.1.1.1 interface ae3.100
set routing-instances VRF-101 protocols ospf area 1.1.1.1 interface irb.2101
set routing-instances VRF-101 protocols ospf area 1.1.1.1 interface all passive
set routing-instances VRF-101 protocols ospf export export-direct
```

## Configure External Multicast PIM Router and PIM RP Router

In this example, an MX Series router acts as the external PIM domain router and PIM RP device. In this procedure, we include the configuration on this device that matches the BL device configuration in "Configure the Border Leaf Devices for External Multicast Connectivity, PIM EVPN Gateway Role, and PIM Options" on page 534. This information helps you interpret show command output to verify the setup, connectivity, and group memberships established for the OISM and AR devices in the fabric.

The PIM router and RP router configuration includes:

- Connections to BL-1 on interface ae1 and to BL-2 on interface ae2, with VLAN-tagging enabled.
- Routing instances of type `virtual-router` (PIM-GW-VR-*n*) that correspond to each tenant VRF-*n* in the OISM configuration on the BL devices.
- Logical units on ae1 and ae2 with corresponding VLANs per virtual router VRF, starting with unit 0 and VLAN-3001 for VRF-1.
- A PIM RP IP address for each virtual router VRF instance.

This procedure shows configuring the following on the PIM router and RP router, as listed in Table 19 on page 516:

- PIM-GW-VR-1 (corresponding to VRF-1) and VLAN 3001 with:
    - Interface ae1.0 to BL-1.
    - Interface ae2.0 to BL-2.
- PIM-GW-VR-101 (corresponding to VRF-101) and VLAN-3101 with:
    - Interface ae1.100 to BL-1.
    - Interface ae2.100 to BL-2.

1. Configure the L3 interfaces on the PIM router that connect to the BL devices (ae1 to BL-1 and ae2 to BL-2).

```
set interfaces xe-1/3/0 gigether-options 802.3ad ae1
set interfaces ae1 vlan-tagging
set interfaces ae1 aggregated-ether-options minimum-links 1
set interfaces ae1 aggregated-ether-options lacp active
set interfaces ae1 aggregated-ether-options lacp periodic fast
set interfaces ae1 unit 0 vlan-id 3001
set interfaces ae1 unit 0 family inet address 172.30.0.0/31
set interfaces ae1 unit 1 vlan-id 3101
set interfaces ae1 unit 1 family inet address 172.30.100.0/31

set interfaces xe-2/1/2 gigether-options 802.3ad ae2
set interfaces ae2 vlan-tagging
set interfaces ae2 aggregated-ether-options minimum-links 1
set interfaces ae2 aggregated-ether-options lacp active
set interfaces ae2 aggregated-ether-options lacp periodic fast
set interfaces ae2 unit 0 vlan-id 3001
set interfaces ae2 unit 0 family inet address 172.31.0.0/31
set interfaces ae2 unit 1 vlan-id 3101
set interfaces ae2 unit 1 family inet address 172.31.100.0/31
```

2. Configure virtual router routing instances corresponding to the OISM tenant VRFs and VLANs in this example.

   Include the L3 interfaces to BL-1 and BL-2 in the routing instances, as well as the device logical loopback interface and logical interfaces that connect to the external source for each routing instance.

```
set routing-instances PIM-GW-VR-1 instance-type virtual-router
set routing-instances PIM-GW-VR-1 interface ae1.0
set routing-instances PIM-GW-VR-1 interface ae2.0
set routing-instances PIM-GW-VR-1 interface lo0.1
set routing-instances PIM-GW-VR-1 interface xe-1/3/3.3001

set routing-instances PIM-GW-VR-101 instance-type virtual-router
set routing-instances PIM-GW-VR-101 interface ae1.100
set routing-instances PIM-GW-VR-101 interface ae2.100
set routing-instances PIM-GW-VR-101 interface lo0.101
set routing-instances PIM-GW-VR-101 interface xe-1/3/3.3101
```

3. In each of the routing instances, configure PIM, a static IP address for the PIM RP, and an OSPF area for PIM for the interfaces.

   See Table 19 on page 516 for the PIM RP static addresses we use in this example.

```
set routing-instances PIM-GW-VR-1 protocols ospf area 1.1.1.1 interface all
set routing-instances PIM-GW-VR-1 protocols pim rp local address 172.22.2.1
set routing-instances PIM-GW-VR-1 protocols pim interface all

set routing-instances PIM-GW-VR-101 protocols ospf area 1.1.1.1 interface all
set routing-instances PIM-GW-VR-101 protocols pim rp local address 172.22.102.1
set routing-instances PIM-GW-VR-101 protocols pim interface all
```

## Configure AR Replicator Role on OISM Spine Devices and AR Leaf Role on OISM Leaf Devices

In an ERB overlay fabric, you can enable OISM with AR. You can assign the AR replicator role to one or more spine devices in the fabric. When a spine device runs as an AR replicator, the AR replicator operates in standalone AR replicator mode. This means the AR replicator role is not collocated with the OISM border leaf role on the device.

When an ingress AR leaf device needs to forward multicast traffic to other AR leaf devices, it uses an AR overlay VXLAN tunnel to send only one copy of the traffic to an available AR replicator device instead. Then, also using AR overlay VXLAN tunnels, the AR replicator device replicates and forwards the traffic to the other AR leaf devices with receivers that subscribed to the multicast stream. AR replicators use ingress replication instead of AR to forward multicast traffic directly to leaf devices that don’t support AR (what we call RNVE devices).

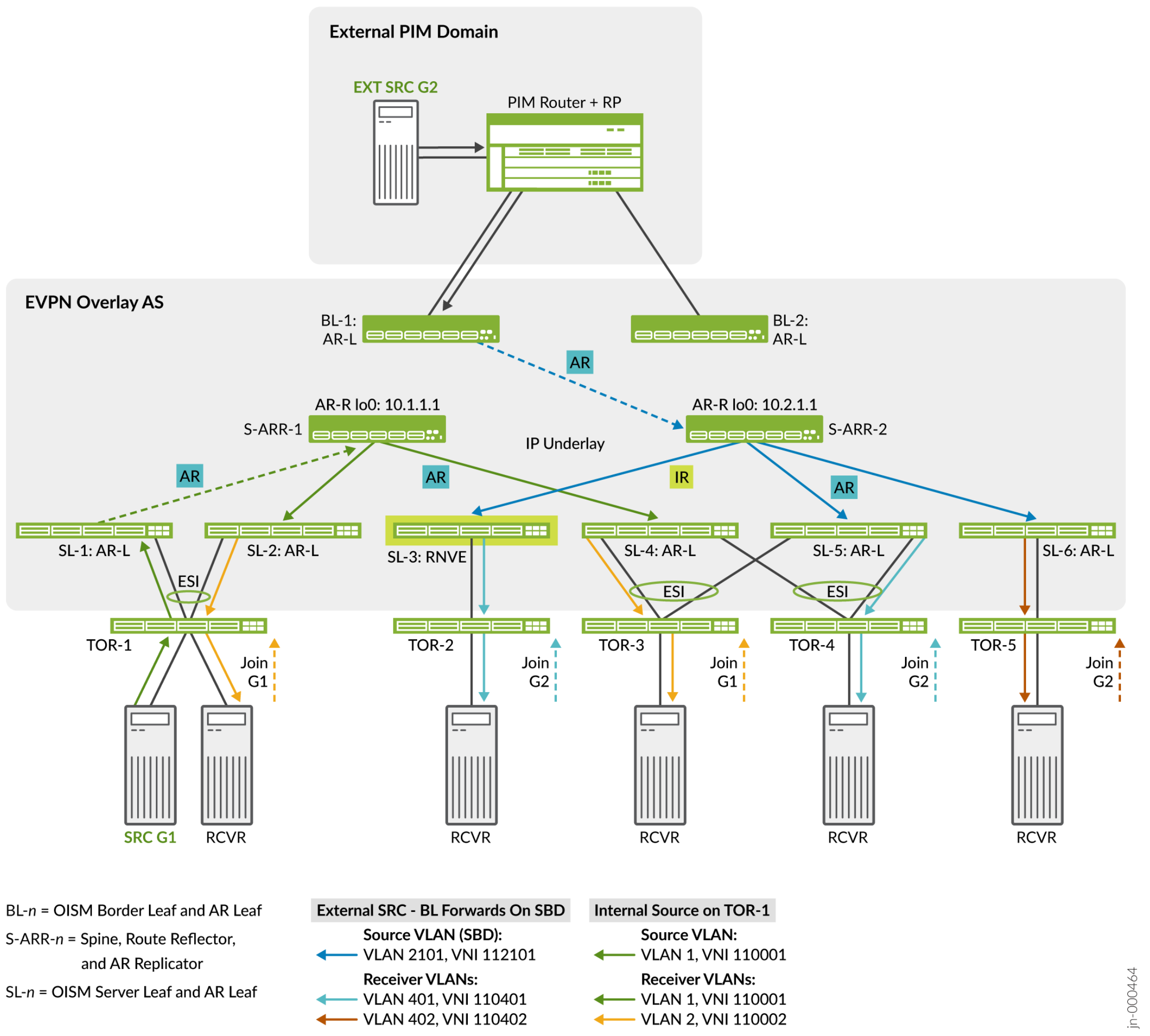
AR leaf devices balance the load of AR replicator requests among the available AR replicator devices using one of two methods, depending on the leaf device platform:

- QFX5000 line of switches (models that run either Junos OS or Junos OS Evolved)—These devices designate a particular AR replicator device for traffic associated with each VLAN or VNI. In this case, the `show evpn multicast-snooping assisted-replication next-hops` CLI command output shows the designated AR replicator for each VNI as the `(Designated Node)`.

- QFX10000 line of switches—These devices actively load-balance among the AR replicators based on traffic flow levels within a VNI. The device doesn’t designate a particular AR replicator for each VNI.

In this example, we have multicast flows from an internal source and an external source. The ERB overlay fabric spine devices (S-ARR-1 and S-ARR-2) act as AR replicator devices. The OISM SL and BL devices act as AR leaf devices, except SL-3, which simulates an RNVE device (we don’t enable the AR leaf role on that device). Figure 92 on page 541 shows how AR works if we consider the multicast streams and corresponding fabric parameters in this example from Table 18 on page 514.

Figure 92: AR with OISM Internal and External Multicast Sources



- In the internal source use case:

  1. SL-1 is the ingress device for an internal multicast stream from multihomed TOR-1 on source VLAN VLAN-1 for traffic to receivers in tenant VRF VRF-1.

  2. SL-1 (a QFX5120 switch) forwards the traffic to its designated AR replicator for VLAN-1 (VNI 110001). The designated AR replicator is S-ARR-1 in this case.

  3. S-ARR-1 replicates and forwards the stream on the source VLAN to the AR leaf devices that host TOR devices with subscribed receivers.

  4. The destination SL devices forward or locally route the traffic to their subscribed receivers.

- In the external source use case:

1. BL-1 is the ingress device for an external multicast stream from the external PIM domain for traffic to receivers in tenant VRF VRF-101.
2. BL-1 (a QFX5130 switch) forwards the traffic to its designated AR replicator for the SBD VLAN, VLAN-2101 (VNI 992101). The designated AR replicator is S-ARR-2 in this case.
3. S-ARR-2 replicates and forwards the stream on the SBD VLAN using AR tunnels to the AR leaf devices that host TORs with subscribed receivers.
4. S-ARR-2 also replicates the stream and uses an ingress replication (IR) tunnel to forward the stream to SL-3, an RNVE leaf device that hosts a TOR device with a subscribed receiver.
5. The destination SL devices forward or locally route the traffic toward their subscribed receivers.

For more details on AR device roles, how AR works, and other use cases besides the ones in this example, see Assisted Replication Multicast Optimization in EVPN Networks.

To configure AR in this example:

1. Configure the AR replicator role on the S-ARR devices.
   a. Configure the device loopback interface lo0 with a secondary IP address specifically for AR functions. The AR replicator advertises this IP address to the network in EVPN Type 3 AR tunnel routes.

      We include the primary loopback address configuration statements here as well so you can more easily identify each S-ARR device.

      *S-ARR-1*:

      ```
      set interfaces lo0 unit 0 family inet address 192.168.2.1/32 primary
      set interfaces lo0 unit 0 family inet address 192.168.2.1/32 preferred
      set interfaces lo0 unit 0 family inet address 10.1.1.1/32
      ```

      *S-ARR-2*:

      ```
      set interfaces lo0 unit 0 family inet address 192.168.2.2/32 primary
      set interfaces lo0 unit 0 family inet address 192.168.2.2/32 preferred
      set interfaces lo0 unit 0 family inet address 10.2.1.1/32
      ```

   b. Configure the AR replicator role on S-ARR-1 and S-ARR-2 in the MAC-VRF instance, using the secondary AR loopback interface you configured in the previous step.

*S-ARR-1*:

```
set routing-instances MACVRF-1 protocols evpn assisted-replication replicator inet 10.1.1.1
set routing-instances MACVRF-1 protocols evpn assisted-replication replicator vxlan-
encapsulation-source-ip ingress-replication-ip
```

*S-ARR-2*:

```
set routing-instances MACVRF-1 protocols evpn assisted-replication replicator inet 10.2.1.1
set routing-instances MACVRF-1 protocols evpn assisted-replication replicator vxlan-
encapsulation-source-ip ingress-replication-ip
```

c. On AR replicator devices in standalone mode, you must also configure the common OISM elements that you configure on the OISM SL and BL devices. You configure those elements in earlier steps in this example. See:

- "Configure an OISM-Enabled EVPN MAC-VRF Instance" on page 522
- "Configure the Tenant VRF Instances for IGMPv2 and IGMPv3 Multicast Receivers" on page 527

You don't need to configure any of the PIM or external multicast elements specific to OISM BL or SL devices.

2. Configure the AR leaf role in the MAC-VRF instance on the OISM BL devices and all SL devices except the RNVE device in this example, SL-3.

Include the `replicator-activation-delay` option as shown. By default, AR leaf devices delay 10 seconds after receiving an AR replicator advertisement before starting to send traffic to that AR replicator device. In a scaled environment, we recommend you make the delay longer to ensure the AR replicator devices have fully learned the current EVPN state from the network. The delay also helps in cases when an AR replicator goes down and comes up again.

*BL-1, BL-2, SL-1, SL-2, SL-4, SL-5, and SL-6*:

```
set routing-instances MACVRF-1 protocols evpn assisted-replication leaf replicator-activation-
delay 30
```

We skip this configuration on SL-3, which acts as a device that doesn't support AR.

## Verify OISM and AR Configuration and Operation

You can use the show commands in the following steps to verify OISM and AR configuration and operation.

1. Verify the underlay and overlay configurations and confirm the fabric has established the BGP state and traffic paths among the devices.

   *SL-1 (device lo0: 192.168.0.1)*:

```
user@SL-1> show bgp summary
Threading mode: BGP I/O
Default eBGP mode: advertise - accept, receive - accept
Groups: 2 Peers: 4 Down peers: 0
Table          Tot Paths  Act Paths Suppressed    History Damp State    Pending
bgp.evpn.0
                   75669      41585          0          0          0          0
inet.0
                      24         20          0          0          0          0
Peer                     AS      InPkt     OutPkt    OutQ   Flaps Last Up/Dwn State|#Active/
Received/Accepted/Damped...
172.16.1.0       4200000021       1201       1227       0       0     9:07:50 Establ
  inet.0: 10/12/12/0
172.16.3.0       4200000022       1208       1226       0       0     9:11:20 Establ
  inet.0: 10/12/12/0
192.168.2.1      4210000001      32260       3854       0       0     9:07:31 Establ
  bgp.evpn.0: 37834/37834/37834/0
  MACVRF-1.evpn.0: 37126/37126/37126/0
  default-switch.evpn.0: 0/0/0/0
  __default_evpn__.evpn.0: 708/708/708/0
192.168.2.2      4210000001      34838       3862       0       0     9:11:11 Establ
  bgp.evpn.0: 3751/37835/37835/0
  MACVRF-1.evpn.0: 3751/37127/37127/0
  default-switch.evpn.0: 0/0/0/0
  __default_evpn__.evpn.0: 0/708/708/0
```

   Run this command on each SL, BL, and S-ARR device in the fabric.

2. Verify the configured VTEPs in the MAC-VRF EVPN instance on the SL, BL, and S-ARR devices.

   You can use the `show ethernet-switching vxlan-tunnel-end-point remote` command, or its alias, the `show mac-vrf forwarding vxlan-tunnel-end-point remote command` (where supported).

   For example:

*SL-1*:

```
user@SL-1> show ethernet-switching vxlan-tunnel-end-point remote summary instance MACVRF-1

Logical System Name       Id  SVTEP-IP         IFL   L3-Idx    SVTEP-Mode    ELP-SVTEP-IP
<default>                  0   192.168.0.1    lo0.0    0
 RVTEP-IP         L2-RTT          IFL-Idx   Interface      NH-Id RVTEP-Mode  ELP-IP
Flags
 10.2.1.1         MACVRF-1        672178185 vtep-133.32777 30050 Replicator
Extended-MH
 10.1.1.1         MACVRF-1        672178188 vtep-133.32780 14959 Replicator
Extended-MH
 192.168.2.1      MACVRF-1        672178189 vtep-133.32781 14960 RNVE
 192.168.2.2      MACVRF-1        672178184 vtep-133.32776 14008 RNVE
 192.168.5.1      MACVRF-1        672178187 vtep-133.32779 14460 Leaf
 192.168.5.2      MACVRF-1        672178190 vtep-133.32782 17439 Leaf
 192.168.0.2      MACVRF-1        672178181 vtep-133.32773 26415 Leaf
 192.168.0.3      MACVRF-1        672178180 vtep-133.32772 23288 RNVE
 192.168.0.4      MACVRF-1        672178183 vtep-133.32775 14004 Leaf
 192.168.0.5      MACVRF-1        672178186 vtep-133.32778 14410 Leaf
 192.168.0.6      MACVRF-1        672178179 vtep-133.32771 22908 Leaf
```

**NOTE**: You also see the AR role of the device on each remote VTEP in the RVTEP-Mode column of the output from this command, as follows

- The primary loopback IP address on an AR replicator device is the ingress replication (IR) IP address, which the device uses to forward traffic to RNVE devices. That's why you see "RNVE" as the role corresponding to the S-ARR device primary loopback addresses in the RVTEP-IP column.

- The secondary loopback IP address you assign for the AR replicator role is the AR IP address. You see "Replicator" as the role for those RVTEP-IP addresses in this output.

- The AR leaf devices and the RNVE device only use IR tunnels, so this command displays "Leaf" or "RNVE" role corresponding to the primary loopback IP address for those devices in the RVTEP-IP column.

Run this command on each SL, BL, and S-ARR device in the fabric.

3. Verify the SL device to TOR device links. These links are ae3 on each SL device. Also, SL-4 and SL-5 are multihoming peers for both TOR-3 and TOR-4, and use ae5 for those additional TOR device links. (See Figure 91 on page 519.)

   For example:

   *SL-1 to TOR-1*:

```
user@SL-1> show lacp interfaces ae3
Aggregated interface: ae3
    LACP state:       Role   Exp   Def  Dist  Col  Syn  Aggr  Timeout  Activity
      xe-0/0/6:0     Actor    No    No   Yes  Yes  Yes   Yes     Fast    Active
      xe-0/0/6:0   Partner    No    No   Yes  Yes  Yes   Yes     Fast    Active
    LACP protocol:        Receive State  Transmit State          Mux State
      xe-0/0/6:0                Current   Fast periodic Collecting distributing

    LACP hold-timer:  Up, Enabled, Interval: 300 sec
                        Status         Re-Start Cnt   TTE(sec)     Hold Start
    xe-0/0/6:0          Not-Running    NA             NA           NA

user@SL-1> show lldp neighbors | grep ae3

xe-0/0/6:0         ae3                 10:0e:7e:af:14:c0   xe-0/0/22:0
tor-1.testdomain.net
```

   *SL-2 to TOR-1*:

```
user@SL-2> show lacp interfaces ae3

Aggregated interface: ae3
    LACP state:       Role   Exp   Def  Dist  Col  Syn  Aggr  Timeout  Activity
      xe-0/0/12      Actor    No    No   Yes  Yes  Yes   Yes     Fast    Active
      xe-0/0/12    Partner    No    No   Yes  Yes  Yes   Yes     Fast    Active
    LACP protocol:        Receive State  Transmit State          Mux State
      xe-0/0/12                 Current   Fast periodic Collecting distributing

    LACP hold-timer:  Up, Enabled, Interval: 300 sec
                        Status         Re-Start Cnt   TTE(sec)     Hold Start
    xe-0/0/12           Not-Running    NA             NA           NA

user@SL-2> show lldp neighbors | grep ae3
```

```
xe-0/0/12          ae3                 10:0e:7e:af:14:c0   xe-0/0/13:3
tor-1.testdomain.net
```

Repeat this command on each SL device for interface ae3, and on SL-4 and SL-5, repeat this command for ae5.

4. Verify external L3 interface OSPF neighbor reachability on the BL devices for each tenant VRF instance.

   *BL-1*:

```
user@BL-1> show interfaces terse | grep ae3.0
ae3.0                   up    up   inet     172.30.0.1/31

user@BL-1> show interfaces terse | grep ae3.100
ae3.100                 up    up   inet     172.30.100.1/31

user@BL-1> show ospf neighbor instance VRF-1
Address          Interface              State          ID               Pri  Dead
172.30.0.0      ae3.0                   Full           172.22.2.1       128    31
172.27.209.242  irb.2001                Full           10.9.9.1         128    38

user@BL-1> show ospf neighbor instance VRF-101
Address          Interface              State          ID               Pri  Dead
172.30.100.0     ae3.100                Full           172.22.102.1     128    36
172.28.53.242    irb.2101               Full           10.9.9.101       128    35
```

   *BL-2*:

```
user@BL-2> show interfaces terse | grep ae3.0
xe-0/0/14:0.0           up    up   aenet    --> ae3.0
ae3.0                   up    up   inet     172.31.0.1/31

user@BL-2> show interfaces terse | grep ae3.100
xe-0/0/14:0.100         up    up   aenet    --> ae3.100
ae3.100                 up    up   inet     172.31.100.1/31

user@BL-2> show ospf neighbor instance VRF-1
Address          Interface              State          ID               Pri  Dead
172.31.0.0       ae3.0                  Full           172.22.2.1       128    34
172.27.209.241   irb.2001               Full           10.8.8.1         128    35

user@BL-2> show ospf neighbor instance VRF-101
```

```
Address          Interface             State          ID               Pri  Dead
172.31.100.0     ae3.100               Full           172.22.102.1     128    33
172.28.53.241    irb.2101              Full           10.8.8.101       128    31
```

5. Verify the PIM router and RP device connections from the external PIM router and RP device to the BL devices.

```
user@pim-gw-rp> show interfaces terse | grep ae1.0
xe-1/3/0.0              up    up   aenet    --> ae1.0
ae1.0                   up    up   inet     172.30.0.0/31

user@pim-gw-rp> show interfaces terse | grep ae1.100
xe-1/3/0.100            up    up   aenet    --> ae1.100
ae1.100                 up    up   inet     172.30.100.0/31

user@pim-gw-rp> show interfaces terse | grep ae2.0
xe-2/1/2.0              up    up   aenet    --> ae2.0
ae2.0                   up    up   inet     172.31.0.0/31

user@pim-gw-rp> show interfaces terse | grep ae2.100
xe-2/1/2.100            up    up   aenet    --> ae2.100
ae2.100                 up    up   inet     172.30.100.0/31

user@pim-gw-rp> show lldp neighbors
Local Interface    Parent Interface    Chassis Id          Port info          System Name
xe-1/3/0           ae1                 68:22:8e:e3:a6:af   713
BL-1.testdomain.net
xe-2/1/2           ae2                 9c:8a:cb:a9:68:a1   1582
BL-2.testdomain.net


user@pim-gw-rp> show ospf neighbor instance PIM-GW-VR-1
Address          Interface             State     ID               Pri  Dead
172.30.0.1       ae1.0                 Full      10.8.8.1         128    34
172.31.0.1       ae2.0                 Full      10.9.9.1         128    36

user@pim-gw-rp> show ospf neighbor instance PIM-GW-VR-101
Address          Interface             State     ID               Pri  Dead
172.30.100.1     ae1.100               Full      10.8.8.101       128    36
172.30.100.1     ae2.100               Full      10.9.9.101       128    31
```

6. Verify AR leaf overlay tunnel load balancing to the available AR replicator devices.

   The AR leaf devices detect the advertised AR replicator devices and load-balance among them using different methods based on the leaf device platform. (See AR Leaf Device Load Balancing with Multiple Replicators for details.)

   In this example, SL-1 is a QFX5120 switch, so as an AR leaf device, SL-1 load balances by assigning an AR replicator device to each VLAN or VNI.

   Run the `show evpn multicast-snooping assisted-replication next-hops instance` *`mac-vrf-instance`* command on the AR leaf devices to see the overlay tunnels and load-balancing next hops to the available AR replicators. On SL devices that designate AR replicators by VNI, the output of this command tags the AR replicator as the `(Designated Node)`. The output doesn't include this tag on AR leaf devices that load balance based on active traffic flow.

   For example, the output here for SL-1 shows that the device assigned:

   - S-ARR-1 as the designated replicator for configured VNIs 110001 and 110003 (which corresponds to VLAN-1 and VLAN-3, respectively)

   - S-ARR-2 as the designated replicator for configured VNI 110002 and 110004 (which correspond to VLAN-2 and VLAN-4, respectively)

   *SL-1*:

```
user@SL-1> show evpn multicast-snooping assisted-replication next-hops instance MACVRF-1

Instance: MACVRF-1
AR Role: AR Leaf

  VN Identifier: 110001
    Load Balance Nexthop Index: 528073
      Load balance to:
      Nexthop Index     Interface         AR IP
      14959             vtep.32780        10.1.1.1 (Designated Node)
      30050             vtep.32777        10.2.1.1


  VN Identifier: 110002
    Load Balance Nexthop Index: 528072
      Load balance to:
      Nexthop Index     Interface         AR IP
      14959             vtep.32780        10.1.1.1
      30050             vtep.32777        10.2.1.1 (Designated Node)
  VN Identifier: 110003
    Load Balance Nexthop Index: 528071
      Load balance to:
      Nexthop Index     Interface         AR IP
      14959             vtep.32780        10.1.1.1 (Designated Node)
      30050             vtep.32777        10.2.1.1

  VN Identifier: 110004
    Load Balance Nexthop Index: 528070
      Load balance to:
      Nexthop Index     Interface         AR IP
      14959             vtep.32780        10.1.1.1
      30050             vtep.32777        10.2.1.1  (Designated Node)
```

Run this command on the SL and BL devices in the fabric.

7. Verify PIM join and multicast group status for a multicast stream from a source inside the fabric behind TOR-1. TOR-1 is multihomed to SL-1 and SL-2 (see Figure 89 on page 511). Receivers on the TOR devices connected to the other SL devices subscribe to multicast groups hosted by this source as listed in Table 18 on page 514. The stream we verify here is intra-VLAN and inter-VLAN traffic with source VLAN VLAN-1 and receiver VLANs VLAN-2, VLAN-3, and VLAN-4.

   With AR enabled, the ingress SL device forwards the multicast source traffic to the designated AR replicator. See Figure 92 on page 541. The AR replicator forwards copies to the SL devices with subscribed receivers on the source VLAN, VLAN-1. Then the SL devices forward the traffic toward the receivers on VLAN-1.

   In this step, you run the commands only for VRF-1, which is the tenant VRF that hosts the internal multicast source traffic in this example. Also, this stream is an IGMPv3 stream with SSM reports, so you see only (S,G) multicast routes. In this case, the output shows the source behind TOR-1 has source IP address 10.0.1.12.

   In this step, we show running verification commands for:

   - PIM join status on SL-1 and SL-2 for the source on multihomed device TOR-1.

     The `show pim join summary` output shows that the SL devices serving TOR-1 saw joins for a total of 6 multicast groups.

   - IGMP snooping multicast group membership status for the receiver behind device TOR-4, which is multihomed to SL-4 and SL-5.

     The `show igmp snooping membership` output shows the multicast group joins from the receiver behind TOR-4. TOR-4 hashes the join messages to either of the multihoming peer SL devices. The number of joins on both devices together (3 on each device) equals the total number of joins in the `show pim join summary` output (6).

- PIM join status summary and details on SL-4 and SL-5 for the receiver behind multihomed device TOR-4.

  When the `show pim join extensive` output on SL-4 and SL-5 show the same upstream and downstream IRB interface, the device is bridging the multicast stream within the same VLAN. When the downstream IRB interfaces are different from the upstream IRB interface, the device is routing the multicast stream between VLANs.

- The designated forwarder among multihoming peers SL-4 and SL-5 that the device elected to forward the traffic to the receiver behind multihomed TOR-4.

  We run the `show evpn instance MACVRF-1 designated-forwarder esi` command for the ESI you configured on the ae5 interfaces from SL-4 and SL-5 to TOR-4.

*SL-1: Internal source—PIM join status for VRF-1*:

```
user@SL-1> show pim join summary instance VRF-1
Instance: PIM.VRF-1 Family: INET

Route type  Route count
(s,g)                6

Instance: PIM.VRF-1 Family: INET6
```

*SL-2: Internal source—PIM join status for VRF-1*:

```
user@SL-1> show pim join summary instance VRF-1
Instance: PIM.VRF-1 Family: INET

Instance: PIM.VRF-1 Family: INET6
```

*SL-4: Receiver—Multicast group membership status on source VLAN VLAN-1 and receiver VLANs VLAN-1 through VLAN-4:*:

```
user@SL-4> show igmp snooping membership vlan VLAN-1 virtual-switch MACVRF-1
VLAN-1 virtual-switch MACVRF-1
Instance: MACVRF-1

Vlan: VLAN-1

Learning-Domain: default
Interface: ae5.0, Groups: 3
```

```
    Group: 233.252.0.21
        Group mode: Include
        Source: 10.0.1.12
        Source timeout: 210
        Last reported by: 10.0.1.162
        Group timeout:       0 Type: Dynamic
    Group: 233.252.0.22
        Group mode: Include
        Source: 10.0.1.12
        Source timeout: 210
        Last reported by: 10.0.1.162
        Group timeout:       0 Type: Dynamic
    Group: 233.252.0.23
        Group mode: Include
        Source: 10.0.1.12
        Source timeout: 210
        Last reported by: 10.0.1.162
        Group timeout:       0 Type: Dynamic

Learning-Domain: default
Interface: ae3.0, Groups: 0

user@SL-4> show igmp snooping membership vlan VLAN-2 virtual-switch MACVRF-1
Instance: MACVRF-1

Vlan: VLAN-2

Learning-Domain: default
Interface: ae3.0, Groups: 0

Learning-Domain: default
Interface: ae5.0, Groups: 0

user@SL-4> show igmp snooping membership vlan VLAN-3 virtual-switch MACVRF-1
Instance: MACVRF-1

Vlan: VLAN-3

Learning-Domain: default
Interface: ae3.0, Groups: 3
    Group: 233.252.0.121
        Group mode: Include
        Source: 10.0.1.12
        Source timeout: 245
        Last reported by: 10.0.3.183
        Group timeout:       0 Type: Dynamic
    Group: 233.252.0.122
        Group mode: Include
        Source: 10.0.1.12
        Source timeout: 245
        Last reported by: 10.0.3.183
        Group timeout:       0 Type: Dynamic
    Group: 233.252.0.123
        Group mode: Include
        Source: 10.0.1.12
        Source timeout: 245
        Last reported by: 10.0.3.183
        Group timeout:       0 Type: Dynamic

Learning-Domain: default
Interface: ae5.0, Groups: 0

user@SL-4> show igmp snooping membership vlan VLAN-4 virtual-switch MACVRF-1
Instance: MACVRF-1

Vlan: VLAN-4

Learning-Domain: default
Interface: ae3.0, Groups: 3
    Group: 233.252.0.121
        Group mode: Include
        Source: 10.0.1.12
        Source timeout: 255
        Last reported by: 10.0.4.183
        Group timeout:       0 Type: Dynamic
    Group: 233.252.0.122
        Group mode: Include
        Source: 10.0.1.12
        Source timeout: 255
        Last reported by: 10.0.4.183
        Group timeout:       0 Type: Dynamic
    Group: 233.252.0.123
        Group mode: Include
        Source: 10.0.1.12
        Source timeout: 255
        Last reported by: 10.0.4.183
        Group timeout:       0 Type: Dynamic

Learning-Domain: default
Interface: ae5.0, Groups: 0
```

*SL-5: Receiver—Multicast group membership status on source VLAN VLAN-1 and receiver VLANs VLAN-1 through VLAN-4*:

```
user@SL-5> show igmp snooping membership vlan VLAN-1 virtual-switch MACVRF-1
Instance: MACVRF-1

Vlan: VLAN-1

Learning-Domain: default
Interface: ae5.0, Groups: 0

Learning-Domain: default
Interface: ae3.0, Groups: 3
    Group: 233.252.0.21
        Group mode: Include
        Source: 10.0.1.12
        Source timeout: 159
        Last reported by: 10.0.1.161
        Group timeout:       0 Type: Dynamic
    Group: 233.252.0.22
        Group mode: Include
        Source: 10.0.1.12
        Source timeout: 159
        Last reported by: 10.0.1.161
        Group timeout:       0 Type: Dynamic
    Group: 233.252.0.23
        Group mode: Include
        Source: 10.0.1.12
        Source timeout: 159
        Last reported by: 10.0.1.161
        Group timeout:       0 Type: Dynamic

user@SL-5> show igmp snooping membership vlan VLAN-2 virtual-switch MACVRF-1
Instance: MACVRF-1

Vlan: VLAN-2
```

```
Learning-Domain: default
Interface: ae3.0, Groups: 3
    Group: 233.252.0.121
        Group mode: Include
        Source: 10.0.1.12
        Source timeout: 235
        Last reported by: 10.0.2.183
        Group timeout:       0 Type: Dynamic
    Group: 233.252.0.122
        Group mode: Include
        Source: 10.0.1.12
        Source timeout: 235
        Last reported by: 10.0.2.183
        Group timeout:       0 Type: Dynamic
    Group: 233.252.0.123
        Group mode: Include
        Source: 10.0.1.12
        Source timeout: 235
        Last reported by: 10.0.2.183
        Group timeout:       0 Type: Dynamic

Learning-Domain: default
Interface: ae5.0, Groups: 3
    Group: 233.252.0.121
        Group mode: Include
        Source: 10.0.1.12
        Source timeout: 179
        Last reported by: 10.0.2.173
        Group timeout:       0 Type: Dynamic
    Group: 233.252.0.122
        Group mode: Include
        Source: 10.0.1.12
        Source timeout: 179
        Last reported by: 10.0.2.173
        Group timeout:       0 Type: Dynamic
    Group: 233.252.0.123
        Group mode: Include
        Source: 10.0.1.12
        Source timeout: 179
        Last reported by: 10.0.2.173
        Group timeout:       0 Type: Dynamic

user@SL-5> show igmp snooping membership vlan VLAN-3 virtual-switch MACVRF-1
```

```
Instance: MACVRF-1

Vlan: VLAN-3

Learning-Domain: default
Interface: ae3.0, Groups: 0

Learning-Domain: default
Interface: ae5.0, Groups: 3
    Group: 233.252.0.121
        Group mode: Include
        Source: 10.0.1.12
        Source timeout: 182
        Last reported by: 10.0.3.173
        Group timeout:       0 Type: Dynamic
    Group: 233.252.0.122
        Group mode: Include
        Source: 10.0.1.12
        Source timeout: 182
        Last reported by: 10.0.3.173
        Group timeout:       0 Type: Dynamic
    Group: 233.252.0.123
        Group mode: Include
        Source: 10.0.1.12
        Source timeout: 182
        Last reported by: 10.0.3.173
        Group timeout:       0 Type: Dynamic

user@SL-5> show igmp snooping membership vlan VLAN-4 virtual-switch MACVRF-1
Instance: MACVRF-1

Vlan: VLAN-4

Learning-Domain: default
Interface: ae3.0, Groups: 0

Learning-Domain: default
Interface: ae5.0, Groups: 3
    Group: 233.252.0.121
        Group mode: Include
        Source: 10.0.1.12
        Source timeout: 189
        Last reported by: 10.0.4.173
        Group timeout:       0 Type: Dynamic
    Group: 233.252.0.122
        Group mode: Include
        Source: 10.0.1.12
        Source timeout: 189
        Last reported by: 10.0.4.173
        Group timeout:       0 Type: Dynamic
    Group: 233.252.0.123
        Group mode: Include
        Source: 10.0.1.12
        Source timeout: 189
        Last reported by: 10.0.4.173
        Group timeout:       0 Type: Dynamic
```

*SL-4: Receiver—PIM join status for VRF-1*:

```
user@SL-4> show pim join summary instance VRF-1
Instance: PIM.VRF-1 Family: INET

Route type  Route count
(s,g)                 6

Instance: PIM.VRF-1 Family: INET6

user@SL-4> show pim join extensive instance VRF-1
Instance: PIM.VRF-1 Family: INET
R = Rendezvous Point Tree, S = Sparse, W = Wildcard

Group: 233.252.0.21
    Source: 10.0.1.12
    Flags: sparse,spt
    Upstream interface: irb.1
    Upstream neighbor: Direct
    Upstream state: Local Source, Local RP
    Keepalive timeout: 340
    Uptime: 03:20:36
    Downstream neighbors:
        Interface: irb.1
            10.0.1.247 State: Join Flags: S   Timeout: Infinity
            Uptime: 03:20:36 Time since last Join: 03:20:26
    Number of downstream interfaces: 1
    Number of downstream neighbors: 1
```

```
.
.
.
Group: 233.252.0.121
    Source: 10.0.1.12
    Flags: sparse,spt
    Upstream interface: irb.1
    Upstream neighbor: Direct
    Upstream state: Local Source, Local RP
    Keepalive timeout:
    Uptime: 03:20:36
    Downstream neighbors:
        Interface: irb.4
            10.0.4.247 State: Join Flags: S   Timeout: Infinity
            Uptime: 03:20:35 Time since last Join: 03:19:16
        Interface: irb.3
            10.0.3.247 State: Join Flags: S   Timeout: Infinity
            Uptime: 03:20:36 Time since last Join: 03:20:26
        Interface: irb.2
            10.0.2.247 State: Join Flags: S   Timeout: Infinity
            Uptime: 03:20:35 Time since last Join: 03:20:26
    Number of downstream interfaces: 3
    Number of downstream neighbors: 3
.
.
.
Instance: PIM.VRF-1 Family: INET6
R = Rendezvous Point Tree, S = Sparse, W = Wildcard
```

*SL-5: Receiver—PIM join status for VRF-1*:

```
user@SL-5> show pim join summary instance VRF-1
Instance: PIM.VRF-1 Family: INET

Route type  Route count
(s,g)                 6

Instance: PIM.VRF-1 Family: INET6

user@SL-5> show pim join extensive instance VRF-1
Instance: PIM.VRF-1 Family: INET
R = Rendezvous Point Tree, S = Sparse, W = Wildcard
```

```
Group: 233.252.0.21
    Source: 10.0.1.12
    Flags: sparse,spt
    Upstream interface: irb.1
    Upstream neighbor: Direct
    Upstream state: Local Source, Local RP
    Keepalive timeout: 353
    Uptime: 03:16:52
    Downstream neighbors:
        Interface: irb.1
            10.0.1.248 State: Join Flags: S   Timeout: Infinity
            Uptime: 03:16:52 Time since last Join: 03:16:43
    Number of downstream interfaces: 1
    Number of downstream neighbors: 1
.
.
.
Group: 233.252.0.121
    Source: 10.0.1.12
    Flags: sparse,spt
    Upstream interface: irb.1
    Upstream neighbor: Direct
    Upstream state: Local Source, Local RP
    Keepalive timeout:
    Uptime: 03:16:52
    Downstream neighbors:
        Interface: irb.2
            10.0.2.248 State: Join Flags: S   Timeout: Infinity
            Uptime: 03:16:52 Time since last Join: 03:16:43
        Interface: irb.3
            10.0.3.248 State: Join Flags: S   Timeout: Infinity
            Uptime: 03:16:04 Time since last Join: 03:16:04
        Interface: irb.4
            10.0.4.248 State: Join Flags: S   Timeout: Infinity
            Uptime: 03:16:02 Time since last Join: 03:16:02
    Number of downstream interfaces: 3
    Number of downstream neighbors: 3
.
.
.
```

```
Instance: PIM.VRF-1 Family: INET6
R = Rendezvous Point Tree, S = Sparse, W = Wildcard
```

*SL-4: Check the designated forwarder to the receiver behind TOR-4*:

Recall that in the SL device configuration, we assign ae5 as the link from both SL-4 and SL5 to TOR-4. We set ESI 00:00:00:ff:00:04:00:01:00:05 on those links. The following output shows that SL-4 is not the designated forwarder for this ESI.

```
user@SL-4> show interfaces terse | grep lo0
lo0                     up    up
lo0.0                   up    up   inet     192.168.0.4         --> 0/0

user@SL-4> show evpn instance MACVRF-1 designated-forwarder esi
00:00:00:ff:00:04:00:01:00:05
Instance: MACVRF-1
  Number of ethernet segments: 630
    ESI: 00:00:00:ff:00:04:00:01:00:05
      Designated forwarder: 192.168.0.5
```

*SL-5: Check the designated forwarder to the receiver behind TOR-4*:

The following output confirms that SL-5 (lo0.0 192.168.0.5) is the designated forwarder for ESI 00:00:00:ff:00:04:00:01:00:05.

```
user@SL-5> show interfaces terse | grep lo0
lo0                     up    up
lo0.0                   up    up   inet     192.168.0.5         --> 0/0

user@SL-5> show evpn instance MACVRF-1 designated-forwarder esi
00:00:00:ff:00:04:00:01:00:05
Instance: MACVRF-1
  Number of ethernet segments: 630
    ESI: 00:00:00:ff:00:04:00:01:00:05
      Designated forwarder: 192.168.0.5
```

8. Verify PIM join and multicast group status for a multicast stream from a source outside the fabric in the external PIM domain. See Figure 89 on page 511. Receivers behind the TOR devices connected to the SL devices subscribe to multicast groups hosted by this source as listed in Table 18 on page 514. The ingress BL device routes the external source traffic from the L3 interface connection to the SBD VLAN (VLAN-2101 in this case).

With AR enabled, the BL device forwards the traffic on the SBD VLAN to an AR replicator (either the designated AR replicator or an AR replicator based on traffic load-balancing, depending on the BL device platform). See Figure 92 on page 541. The AR replicator forwards copies on the SBD to the SL devices with subscribed receivers. Then the SL devices forward or locally route the traffic toward the receivers on the tenant VLANs.

In this step, you run the commands only for VRF-101, which is the tenant VRF that hosts external multicast source traffic in this example. Also, this stream is an IGMPv2 stream with ASM reports, so you see only (*,G) multicast routes.

In this step, you run verification commands for:

- PIM join status on BL-1 and BL-2 as the ingress devices for the external multicast source.
- IGMP snooping multicast group membership status for a receiver behind device TOR-1, which is multihomed to SL-1 and SL-2.
- PIM join status on SL-1 and SL-2 for the receiver on multihomed device TOR-1.

*BL-1: Ingress BL device for external source—PIM join status for VRF-101*:

```
user@BL-1> show pim join summary instance VRF-101

Instance: PIM.VRF-101 Family: INET

Route type  Route count
(*,g)                 6

Instance: PIM.VRF-101 Family: INET6

user@BL-1> show pim join extensive instance VRF-101

Instance: PIM.VRF-101 Family: INET
R = Rendezvous Point Tree, S = Sparse, W = Wildcard

Group: 233.252.0.1
    Source: *
    RP: 172.22.102.1
    Flags: sparse,rptree,wildcard
    Upstream interface: ae3.100
    Upstream neighbor: 172.30.100.0
    Upstream state: None
    Uptime: 00:17:02
    Downstream neighbors:
        Interface: irb.2101
            172.28.53.241 State: Join Flags: SRW  Timeout: Infinity
            Uptime: 00:17:02 Time since last Join: 00:17:02
    Number of downstream interfaces: 1
    Number of downstream neighbors: 1
.
.
.
Group: 233.252.0.101
    Source: *
    RP: 172.22.102.1
    Flags: sparse,rptree,wildcard
    Upstream interface: ae3.100
    Upstream neighbor: 172.30.100.0
    Upstream state: None
    Uptime: 00:17:03
    Downstream neighbors:
        Interface: irb.2101
            172.28.53.241 State: Join Flags: SRW  Timeout: Infinity
            Uptime: 00:17:03 Time since last Join: 00:17:03
    Number of downstream interfaces: 1
    Number of downstream neighbors: 1
.
.
.

Instance: PIM.VRF-101 Family: INET6
R = Rendezvous Point Tree, S = Sparse, W = Wildcard
```

*BL-2: Ingress BL device for external source—PIM join status for VRF-101*:

```
user@BL-2> show pim join summary instance VRF-101

Instance: PIM.VRF-101 Family: INET

Route type  Route count
(*,g)                6

Instance: PIM.VRF-101 Family: INET6

user@BL-2> show pim join extensive instance VRF-101
```

```
Instance: PIM.VRF-101 Family: INET
R = Rendezvous Point Tree, S = Sparse, W = Wildcard

Group: 233.252.0.1
    Source: *
    RP: 172.22.102.1
    Flags: sparse,rptree,wildcard
    Upstream interface: ae3.100
    Upstream neighbor: 172.31.100.0
    Upstream state: Join to RP
    Uptime: 01:02:35
    Downstream neighbors:
        Interface: irb.2101
            172.28.53.242 State: Join Flags: SRW  Timeout: Infinity
            Uptime: 01:02:35 Time since last Join: 01:02:35
    Number of downstream interfaces: 1
    Number of downstream neighbors: 1
.
.
.
Group: 233.252.0.101
    Source: *
    RP: 172.22.102.1
    Flags: sparse,rptree,wildcard
    Upstream interface: ae3.100
    Upstream neighbor: 172.31.100.0
    Upstream state: Join to RP
    Uptime: 01:02:39
    Downstream neighbors:
        Interface: irb.2101
            172.28.53.242 State: Join Flags: SRW  Timeout: Infinity
            Uptime: 01:02:39 Time since last Join: 01:02:39
    Number of downstream interfaces: 1
    Number of downstream neighbors: 1
.
.
.

Instance: PIM.VRF-101 Family: INET6
R = Rendezvous Point Tree, S = Sparse, W = Wildcard
```

*SL-1: Receiver—Multicast group membership status for VLANs associated with VRF-101 (VLAN-401 through VLAN-404)*:

```
user@SL-1> show igmp snooping membership vlan VLAN-401 virtual-switch MACVRF-1
Instance: MACVRF-1

Vlan: VLAN-401

Learning-Domain: default
Interface: ae3.0, Groups: 2
    Group: 233.252.0.1
        Group mode: Exclude
        Source: 0.0.0.0
        Last reported by: 10.1.145.18
        Group timeout:     213 Type: Dynamic
    Group: 233.252.0.3
        Group mode: Exclude
        Source: 0.0.0.0
        Last reported by: 10.1.145.18
        Group timeout:     206 Type: Dynamic

user@SL-1> show igmp snooping membership vlan VLAN-402 virtual-switch MACVRF-1
Instance: MACVRF-1

Vlan: VLAN-402

Learning-Domain: default
Interface: ae3.0, Groups: 1
    Group: 233.252.0.103
        Group mode: Exclude
        Source: 0.0.0.0
        Last reported by: 10.1.146.18
        Group timeout:     205 Type: Dynamic

user@SL-1> show igmp snooping membership vlan VLAN-403 virtual-switch MACVRF-1
Instance: MACVRF-1

Vlan: VLAN-403

Learning-Domain: default
Interface: ae3.0, Groups: 2
    Group: 233.252.0.102
```

```
        Group mode: Exclude
        Source: 0.0.0.0
        Last reported by: 10.1.147.18
        Group timeout:     197 Type: Dynamic
    Group: 233.252.0.103
        Group mode: Exclude
        Source: 0.0.0.0
        Last reported by: 10.1.147.18
        Group timeout:     201 Type: Dynamic

user@SL-1> show igmp snooping membership vlan VLAN-404 virtual-switch MACVRF-1
Instance: MACVRF-1

Vlan: VLAN-404

Learning-Domain: default
Interface: ae3.0, Groups: 2
    Group: 233.252.0.101
        Group mode: Exclude
        Source: 0.0.0.0
        Last reported by: 10.1.148.18
        Group timeout:     187 Type: Dynamic
    Group: 233.252.0.102
        Group mode: Exclude
        Source: 0.0.0.0
        Last reported by: 10.1.148.18
        Group timeout:     192 Type: Dynamic
```

*SL-2: Receiver—Multicast group membership status for VLANs associated with VRF-101 (VLAN-401 through VLAN-404)*:

```
user@SL-2> show igmp snooping membership vlan VLAN-401 virtual-switch MACVRF-1
Instance: MACVRF-1

Vlan: VLAN-401

Learning-Domain: default
Interface: ae3.0, Groups: 1
    Group: 233.252.0.2
        Group mode: Exclude
        Source: 0.0.0.0
        Last reported by: 10.1.145.18
        Group timeout:     230 Type: Dynamic

user@SL-2> **show igmp snooping membership vlan VLAN-402 virtual-switch MACVRF-1**
Instance: MACVRF-1

Vlan:VLAN-402

Learning-Domain: default
Interface: ae3.0, Groups: 2
    Group: 233.252.0.101
        Group mode: Exclude
        Source: 0.0.0.0
        Last reported by: 10.1.146.18
        Group timeout:     222 Type: Dynamic
    Group: 233.252.0.102
        Group mode: Exclude
        Source: 0.0.0.0
        Last reported by: 10.1.146.18
        Group timeout:     229 Type: Dynamic

user@SL-2> **show igmp snooping membership vlan VLAN-403 virtual-switch MACVRF-1**
Instance: MACVRF-1

Vlan: VLAN-403

Learning-Domain: default
Interface: ae3.0, Groups: 1
    Group: 233.252.0.101
        Group mode: Exclude
        Source: 0.0.0.0
        Last reported by: 10.1.147.18
        Group timeout:     223 Type: Dynamic

user@SL-2> **show igmp snooping membership vlan VLAN-404 virtual-switch MACVRF-1**
Instance: MACVRF-1

Vlan: VLAN-404

Learning-Domain: default
Interface: ae3.0, Groups: 1
    Group: 233.252.0.103
        Group mode: Exclude
        Source: 0.0.0.0
        Last reported by: 10.1.148.18
        Group timeout:     219 Type: Dynamic
```

*SL-1: Receiver—PIM join status for VRF-101*:

```
user@SL-1> show pim join summary instance VRF-101

Instance: PIM.VRF-101 Family: INET

Route type  Route count
(*,g)                 6

Instance: PIM.VRF-101 Family: INET6


user@SL-1> show pim join extensive instance VRF-101

Instance: PIM.VRF-101 Family: INET
R = Rendezvous Point Tree, S = Sparse, W = Wildcard

Group: 233.252.0.1
    Source: *
    RP: 172.22.102.1
    Flags: sparse,rptree,wildcard
    Upstream interface: Local
    Upstream neighbor: Local
    Upstream state: Local RP
    Uptime: 04:33:22
    Downstream neighbors:
        Interface: irb.401
            10.1.145.243 State: Join Flags: SRW  Timeout: Infinity
            Uptime: 04:33:22 Time since last Join: 04:33:12
        Interface: irb.2101
            172.28.53.243 State: Join Flags: SRW  Timeout: Infinity
            Uptime: 04:29:18 Time since last Join: 04:29:09
    Number of downstream interfaces: 2
    Number of downstream neighbors: 2
.
.
.
```

```
Group: 233.252.0.101
    Source: *
    RP: 172.22.102.1
    Flags: sparse,rptree,wildcard
    Upstream interface: Local
    Upstream neighbor: Local
    Upstream state: Local RP
    Uptime: 04:33:22
    Downstream neighbors:
        Interface: irb.402
            10.1.146.243 State: Join Flags: SRW  Timeout: Infinity
            Uptime: 04:26:44 Time since last Join: 04:26:44
        Interface: irb.403
            10.1.147.243 State: Join Flags: SRW  Timeout: Infinity
            Uptime: 04:26:44 Time since last Join: 04:26:44
        Interface: irb.404
            10.1.148.243 State: Join Flags: SRW  Timeout: Infinity
            Uptime: 04:33:22 Time since last Join: 04:33:12
        Interface: irb.2101
            172.28.53.243 State: Join Flags: SRW  Timeout: Infinity
            Uptime: 04:29:18 Time since last Join: 04:29:09
    Number of downstream interfaces: 4
    Number of downstream neighbors: 4
.
.
.

Instance: PIM.VRF-101 Family: INET6
R = Rendezvous Point Tree, S = Sparse, W = Wildcard
```

*SL-2: Receiver—PIM join status for VRF-101*:

```
user@SL-2> show pim join summary instance VRF-101

Instance: PIM.VRF-101 Family: INET

Route type  Route count
(*,g)                 6

Instance: PIM.VRF-101 Family: INET6
```

```
user@SL-2> show pim join extensive instance VRF-101

Instance: PIM.VRF-101 Family: INET
R = Rendezvous Point Tree, S = Sparse, W = Wildcard

Group: 233.252.0.1
    Source: *
    RP: 172.22.102.1
    Flags: sparse,rptree,wildcard
    Upstream interface: Local
    Upstream neighbor: Local
    Upstream state: Local RP
    Uptime: 04:29:35
    Downstream neighbors:
        Interface: irb.2101
            172.28.53.244 State: Join Flags: SRW  Timeout: Infinity
            Uptime: 04:29:35 Time since last Join: 04:29:35
        Interface: irb.401
            10.1.145.244 State: Join Flags: SRW  Timeout: Infinity
            Uptime: 04:29:35 Time since last Join: 04:29:35
    Number of downstream interfaces: 2
    Number of downstream neighbors: 2
.
.
.
Group: 233.252.0.101
    Source: *
    RP: 172.22.102.1
    Flags: sparse,rptree,wildcard
    Upstream interface: Local
    Upstream neighbor: Local
    Upstream state: Local RP
    Uptime: 04:35:36
    Downstream neighbors:
        Interface: irb.2101
            172.28.53.244 State: Join Flags: SRW  Timeout: Infinity
            Uptime: 04:31:48 Time since last Join: 04:31:40
        Interface: irb.404
            10.1.148.244 State: Join Flags: SRW  Timeout: Infinity
            Uptime: 04:29:35 Time since last Join: 04:29:35
        Interface: irb.403
            10.1.147.244 State: Join Flags: SRW  Timeout: Infinity
            Uptime: 04:35:36 Time since last Join: 04:35:27
        Interface: irb.402
            10.1.146.244 State: Join Flags: SRW  Timeout: Infinity
            Uptime: 04:35:31 Time since last Join: 04:35:27
    Number of downstream interfaces: 4
    Number of downstream neighbors: 4
.
.
.

Instance: PIM.VRF-101 Family: INET6
R = Rendezvous Point Tree, S = Sparse, W = Wildcard
```

9. Verify the OISM devices use EVPN Type 6 (SMET) routes to optimize multicast traffic flow in the EVPN core. View the Type 6 routes in the EVPN routing tables (bgp.evpn.0 and MACVRF-1.evpn.0) on OISM leaf devices and the spine devices that act as AR replicators. Type 6 routes are advertised only on the SBD VLAN.

   For example, here we view Type 6 routes for interested receivers behind TOR-4, which is multihomed to peer SL devices SL-4 and SL-5. We show results for parameters of the featured multicast stream in Table 18 on page 514 for tenant VRF-1, with IGMPv3 traffic between an internal source and internal receivers:

   - VLANs: VLAN-1 through VLAN-4, which map to VNIs 110001 through 110004
   - SBD VLAN: VLAN-2001, which maps to VNI 992001
   - Internal source: Behind SL-1 and SL-2 on TOR-1, with internal IP address 10.0.1.12
   - Multicast groups: 233.252.0.121 through 233.252.0.123

   These commands show:

   - S-ARR-1 and SL-ARR-2 received Type 6 routes from SL-4 and SL-5 on the SBD (VLAN-2001 VNI 992001).
   - SL-4 (lo0: 192.168.0.4) and SL-5 (lo0: 192.168.0.5) received joins from a receiver behind multihomed TOR-4 for multicast groups 233.252.0.121 through 233.252.0.123.
   - Source (10.0.1.12) and Group information, because the receivers send IGMPv3 joins.

   *For S-ARR-1 — Type 6 routes from SL-4 and SL-5*:

   S-ARR-1 links to SL-4 on ae4 (172.16.7.0/31).

```
user@S-ARR-1> show route table bgp.evpn.0 match-prefix 6:192.168.0.4* | find 233.252.0.121
6:192.168.0.4:33301::992001::10.0.1.12::233.252.0.121::192.168.0.4/520
                   *[BGP/170] 03:59:03, localpref 100, from 192.168.0.4
```

```
                      AS path: I, validation-state: unverified
                    >  to 172.16.7.1 via ae4.0
6:192.168.0.4:33301::992001::10.0.1.12::233.252.0.122::192.168.0.4/520
                   *[BGP/170] 03:59:03, localpref 100, from 192.168.0.4
                      AS path: I, validation-state: unverified
                    >  to 172.16.7.1 via ae4.0
6:192.168.0.4:33301::992001::10.0.1.12::233.252.0.123::192.168.0.4/520
                   *[BGP/170] 03:59:03, localpref 100, from 192.168.0.4
                      AS path: I, validation-state: unverified
                    >  to 172.16.7.1 via ae4.0

user@S-ARR-1> show route table MACVR-1.evpn.0 match-prefix 6:192.168.0.4* | find
233.252.0.121
6:192.168.0.4:33301::994002::10.0.1.12::233.252.0.121::192.168.0.4/520
                   *[BGP/170] 04:04:56, localpref 100, from 192.168.0.4
                      AS path: I, validation-state: unverified
                    >  to 172.16.7.1 via ae4.0
6:192.168.0.4:33301::994002::10.0.1.12::233.252.0.122::192.168.0.4/520
                   *[BGP/170] 04:04:56, localpref 100, from 192.168.0.4
                      AS path: I, validation-state: unverified
                    >  to 172.16.7.1 via ae4.0
6:192.168.0.4:33301::994002::10.0.1.12::233.252.0.123::192.168.0.4/520
                   *[BGP/170] 04:04:56, localpref 100, from 192.168.0.4
                      AS path: I, validation-state: unverified
                    >  to 172.16.7.1 via ae4.0
```

S-ARR-1 links to SL-5 on ae5 (172.16.8.0/31):

```
user@S-ARR-1> show route table bgp.evpn.0 match-prefix 6:192.168.0.5* | find 233.252.0.121
6:192.168.0.5:33301::992001::10.0.1.12::233.252.0.121::192.168.0.5/520
                   *[BGP/170] 04:03:51, localpref 100, from 192.168.0.5
                      AS path: I, validation-state: unverified
                    >  to 172.16.8.1 via ae5.0
6:192.168.0.5:33301::992001::10.0.1.12::233.252.0.122::192.168.0.5/520
                   *[BGP/170] 04:03:51, localpref 100, from 192.168.0.5
                      AS path: I, validation-state: unverified
                    >  to 172.16.8.1 via ae5.0
6:192.168.0.5:33301::992001::10.0.1.12::233.252.0.123::192.168.0.5/520
                   *[BGP/170] 04:03:51, localpref 100, from 192.168.0.5
                      AS path: I, validation-state: unverified
                    >  to 172.16.8.1 via ae5.0
```

```
user@S-ARR-1> show route table MACVR-1.evpn.0  match-prefix 6:192.168.0.5* | find
233.252.0.121
6:192.168.0.5:33301::994002::10.0.1.12::233.252.0.121::192.168.0.5/520
                   *[BGP/170] 04:06:52, localpref 100, from 192.168.0.5
                      AS path: I, validation-state: unverified
                    >  to 172.16.8.1 via ae5.0
6:192.168.0.5:33301::994002::10.0.1.12::233.252.0.122::192.168.0.5/520
                   *[BGP/170] 04:06:52, localpref 100, from 192.168.0.5
                      AS path: I, validation-state: unverified
                    >  to 172.16.8.1 via ae5.0
6:192.168.0.5:33301::994002::10.0.1.12::233.252.0.123::192.168.0.5/520
                   *[BGP/170] 04:06:52, localpref 100, from 192.168.0.5
                      AS path: I, validation-state: unverified
                    >  to 172.16.8.1 via ae5.0
```

Run the same commands on S-ARR-2 to see similar output on that device. S-ARR-2 links to SL-4 on ae4 (172.16.9.0/31) and SL-5 on ae5 (172.16.10.0/31).

*SL-4 — Locally generated Type 6 routes, and Type 6 routes from other SL devices by way of S-ARR-1 and S-ARR-2*:

```
user@SL-4> show route table MACVRF-1.evpn.0 match-prefix 6*192.168.0.4 | find 233.252.0.121
6:192.168.0.4:33301::994002::10.0.1.12::233.252.0.121::192.168.0.4/520
                   *[EVPN/170] 03:38:48
                       Indirect
6:192.168.0.4:33301::994002::10.0.1.12::233.252.0.122::192.168.0.4/520
                   *[EVPN/170] 03:38:48
                       Indirect
6:192.168.0.4:33301::994002::10.0.1.12::233.252.0.123::192.168.0.4/520
                   *[EVPN/170] 03:38:48
                       Indirect

user@SL-4> show route table MACVRF-1.evpn.0 match-prefix 6*192.168.0.5 | find 233.252.0.121
6:192.168.0.5:33301::994002::10.0.1.12::233.252.0.121::192.168.0.5/520
                   *[BGP/170] 03:42:51, localpref 100, from 192.168.2.1
                      AS path: I, validation-state: unverified
                    >  to 172.16.7.0 via ae1.0
                       to 172.16.9.0 via ae2.0
                    [BGP/170] 03:42:51, localpref 100, from 192.168.2.2
                      AS path: I, validation-state: unverified
                    >  to 172.16.7.0 via ae1.0
```

```
                         to 172.16.9.0 via ae2.0
  6:192.168.0.5:33301::994002::10.0.1.12::233.252.0.122::192.168.0.5/520
                      *[BGP/170] 03:42:51, localpref 100, from 192.168.2.1
                          AS path: I, validation-state: unverified
                       >  to 172.16.7.0 via ae1.0
                          to 172.16.9.0 via ae2.0
                       [BGP/170] 03:42:51, localpref 100, from 192.168.2.2
                          AS path: I, validation-state: unverified
                       >  to 172.16.7.0 via ae1.0
                          to 172.16.9.0 via ae2.0
  6:192.168.0.5:33301::994002::10.0.1.12::233.252.0.123::192.168.0.5/520
                      *[BGP/170] 03:42:51, localpref 100, from 192.168.2.1
                          AS path: I, validation-state: unverified
                       >  to 172.16.7.0 via ae1.0
                          to 172.16.9.0 via ae2.0
                       [BGP/170] 03:42:51, localpref 100, from 192.168.2.2
                          AS path: I, validation-state: unverified
                       >  to 172.16.7.0 via ae1.0
                          to 172.16.9.0 via ae2.0
```

You see similar Type 6 routes as remote routes from the other SL devices, which also serve receivers for the same groups in the same tenant VRF for the internal source on TOR-1 (multihomed to SL-1 and SL-2).

Run these commands on SL-5 to see the Type 6 routes on that device. Match on prefix 6:192.168.0.5 to see the locally generated Type 6 routes for SL-5. Match on other device prefixes (such as 6*192.168.0.4 for SL-4) to see the remotely generated Type 6 routes.

10. Verify the peer devices for multihoming ESs use EVPN Type 7 routes to synchronize multicast join states.

    Use the `show route table __default_evpn__.evpn.0` command to see Type 7 route prefixes.

    For example, here we show Type 7 routes generated by peer SL devices SL-4 and SL-5 for the receiver behind TOR-4, with an internal source and IGMPv3 joins (see the parameters in Table 18 on page 514 for VRF-1). TOR-4 hashes join messages from its receivers to either SL-4 or SL-5. The devices each advertise Type 7 routes to their multihoming peers for the joins they receive, to synchronize the join status among them.

    These commands show:

    - SL-5 locally generated Type 7 routes to advertise to SL-4
    - SL-5 received Type 7 route advertisements from SL-4 for joins that TOR-4 hashed to SL-4
    - OISM devices advertise Type 7 and Type 8 routes on the OISM revenue VLANs, not the SBD.

In this case, receivers joined groups on VLAN-2 (VNI 110002, hashed to SL-5) and VLAN-3 (VNI-110003, hashed to SL-4).

Note that the ESI we configured for the SL-4 and SL-5 links to TOR-4 is 00:00:00:ff:00:04:00:01:00:05, which you see in the Type 7 route entry in the routing table.

*SL-5 — Locally generated Type 7 routes to advertise to SL-4*:

This output shows that SL-5 locally generated three Type 7 routes for joins on VLAN-2 (VNI 110002) for multicast groups 233.252.0.121 through 233.252.0.123. .

```
user@SL-5> show route table __default_evpn__.evpn.0 match-prefix 7*192.168.0.5 | find
233.252.0.121
7:192.168.0.5:33301::ff000400010005::110002::10.0.1.12::233.252.0.121::192.168.0.5/600

                   *[EVPN/170] 03:56:17
                       Indirect
7:192.168.0.5:33301::ff000400010005::110002::10.0.1.12::233.252.0.122::192.168.0.5/600

                   *[EVPN/170] 03:56:17
                       Indirect
7:192.168.0.5:33301::ff000400010005::110002::10.0.1.12::233.252.0.123::192.168.0.5/600

                   *[EVPN/170] 03:56:17
                       Indirect
```

*SL-5 — Type 7 routes from SL-4*:

This output shows that SL-5 received three Type 7 route advertisements from SL-4 for joins on VLAN-3 (VNI 110003) for multicast groups 233.252.0.121 through 233.252.0.123.

```
user@SL-5> show route table __default_evpn__.evpn.0 match-prefix 7*192.168.0.4 | find
233.252.0.121
7:192.168.0.4:33301::ff000400010005::110003::10.0.1.12::233.252.0.121::192.168.0.4/600

                   *[BGP/170] 03:54:39, localpref 100, from 192.168.2.1
                      AS path: I, validation-state: unverified
                       to 172.16.8.0 via ae1.0
                    >  to 172.16.10.0 via ae2.0
                    [BGP/170] 03:54:39, localpref 100, from 192.168.2.2
                      AS path: I, validation-state: unverified
                       to 172.16.8.0 via ae1.0
                    >  to 172.16.10.0 via ae2.0
```

```
7:192.168.0.4:33301::ff000400010005::110003::10.0.1.12::233.252.0.122::192.168.0.4/600

                   *[BGP/170] 03:54:39, localpref 100, from 192.168.2.1
                      AS path: I, validation-state: unverified
                       to 172.16.8.0 via ae1.0
                    >  to 172.16.10.0 via ae2.0
                    [BGP/170] 03:54:39, localpref 100, from 192.168.2.2
                      AS path: I, validation-state: unverified
                       to 172.16.8.0 via ae1.0
                    >  to 172.16.10.0 via ae2.0
7:192.168.0.4:33301::ff000400010005::110003::10.0.1.12::233.252.0.123::192.168.0.4/600

                   *[BGP/170] 03:54:39, localpref 100, from 192.168.2.1
                      AS path: I, validation-state: unverified
                       to 172.16.8.0 via ae1.0
                    >  to 172.16.10.0 via ae2.0
                    [BGP/170] 03:54:39, localpref 100, from 192.168.2.2
                      AS path: I, validation-state: unverified
                       to 172.16.8.0 via ae1.0
                    >  to 172.16.10.0 via ae2.0
```

Run these commands on SL-4 to see Type 7 routes on that device. Match on prefix 7:192.168.0.5 to see the Type 7 routes from its multihoming peer, SL-5. Match on prefix 7:192.168.0.4 to see the locally generated Type 7 routes that SL-4 advertises to SL-5.

You can use these commands to see Type 7 routes on the other multihoming peer SL device pairs.

## RELATED DOCUMENTATION

Optimized Intersubnet Multicast for ERB Overlay Networks | **39**

Optimized Intersubnet Multicast in EVPN Networks

Assisted Replication Multicast Optimization in EVPN Networks

# Enhanced Optimized Intersubnet Multicast (OISM) Implementation

SUMMARY

Configure enhanced optimized intersubnet multicast (OISM) in a scaled EVPN-VXLAN edge-routed bridging (ERB) overlay fabric when the OISM leaf devices host a large number of diverse revenue VLANs.

IN THIS SECTION

In EVPN-VXLAN edge-routed bridging (ERB) overlay fabric designs, the leaf devices forward traffic within tenant VLANs and route traffic between tenant VLANs. To support efficient multicast traffic flows in scaled ERB overlay fabrics, we support optimized intersubnet multicast (OISM) with both internal and external multicast sources and receivers.

## Regular OISM and Enhanced OISM Differences

We refer to our original OISM implementation as *regular OISM*. Regular OISM uses a symmetric bridge domains OISM model that requires you to configure all revenue VLANs (the tenant VLANs) in the network on all OISM leaf devices. See "Optimized Intersubnet Multicast (OISM) with Assisted Replication (AR) for Edge-Routed Bridging Overlays" on page 509 in this guide for an example of a regular OISM configuration. Due to the symmetric VLANs configuration requirement, we also call regular OISM the "bridge domains everywhere" (BDE) version of OISM.

Enhanced OISM uses an asymmetric bridge domains OISM model in which you don't need to configure all revenue VLANs in the network on all OISM devices. On each leaf device that is not a multihoming peer with another leaf device, you can configure only the revenue VLANs that device hosts. However, the design still requires you to configure matching revenue VLANs on leaf devices that are multihoming peers. Enhanced OISM has some operational differences from regular OISM that support the asymmetric bridge domains model, but most of the configuration elements you need to set up are the same. Because you can configure the VLANs asymmetrically, we also call enhanced OISM the "bridge domains not everywhere" (BDNE) version of OISM.

**NOTE**: Multihoming peer leaf devices are leaf devices that share an Ethernet segment (ES) for an attached multihomed client host, customer edge (CE) device, or top-of-rack (TOR) device.

The enhanced OISM asymmetric bridge domain model enables OISM to scale well when your network has leaf devices that host a large number of diverse revenue VLANs.

This example shows enhanced OISM configuration elements and verification steps tested in our scaled reference environment. We describe a few use cases here that highlight the main differences between regular OISM mode and enhanced OISM mode.

## Enhanced OISM Use Cases Overview

**IN THIS SECTION**

The enhanced OISM use cases included in this example don't include full device configurations. Instead, we focus on the sections of the configuration for the test environment differences and the enhanced OISM operational differences compared to the regular OISM example in "Optimized Intersubnet Multicast (OISM) with Assisted Replication (AR) for Edge-Routed Bridging Overlays" on page 509, such as:

- Using EBGP for the overlay peering in the EVPN core network (instead of IBGP, which the regular OISM example configuration environment uses for the overlay peering).
- Configuring the same set of revenue VLANs only on multihoming peer leaf devices, and configuring different sets of revenue VLANs on the other OISM leaf devices.
- Configuring the VRF instances and IRB interfaces on leaf devices that host different sets of revenue VLANs.
- Enabling IPv6 multicast with MLDv1 any-source multicast (ASM) or MLDv2 source-specific multicast (SSM), which is supported with regular OISM but wasn't included in the regular OISM configuration example.
- Verifying behaviors specific to enhanced OISM operation (see "Operational Differences in Enhanced OISM Mode" on page 579).

Refer to the regular OISM example in "Optimized Intersubnet Multicast (OISM) with Assisted Replication (AR) for Edge-Routed Bridging Overlays" on page 509 for complete details on configuring all of the required elements that are common for both regular OISM and enhanced OISM, such as:

- EVPN fabric interfaces
- Underlay and overlay peering (except here we cover EBGP for the overlay peering)
- EVPN MAC-VRF instances
- Multicast protocols—IGMP, IGMP snooping, MLD, MLD snooping, Protocol Independent Multicast (PIM)

- The SBD, the revenue VLANs, and their corresponding IRB interfaces
- The tenant VRF instances
- OISM device roles—Server leaf, or border leaf as a PIM EVPN gateway (PEG) for exchanging multicast traffic from and to external sources and external receivers
- Interfaces to and from an external PIM domain

## Configuration Differences in Enhanced OISM Mode

You configure enhanced OISM in almost the same way that you configure regular OISM, using the same statements to set up the OISM environment. The only differences are:

- To enable enhanced mode instead of regular mode, configure the `enhanced-oism` option at the `[edit forwarding-options multicast-replication evpn irb]` hierarchy level instead of configuring the `oism` option that enables regular OISM.
- Configure the virtual routing and forwarding (VRF) instances on the OISM leaf devices with the revenue VLANs each device hosts. On each set of multihoming peer leaf devices, however, be sure to configure the same set of revenue VLANs, which should be a combination of all revenue VLANs used by the receivers behind those peer leaf devices.
- Configure an OSPF area for server leaf device connectivity on the SBD. With enhanced OISM, the server leaf devices need Layer 3 connectivity to route source traffic onto the SBD for east-west traffic. As a result, in each tenant VRF, you configure an OSPF area on the server leaf devices with the SBD IRB interface in active mode to form adjacencies on the SBD. You configure the other interfaces in OSPF passive mode.

  Note that on the OISM border leaf devices, for both regular and enhanced OISM, we require you to configure an OSPF area in each tenant VRF with the following interfaces:

  - The SBD IRB interface, in OSPF active mode
  - The PEG interface that provides access to external multicast sources and receivers, in OSPF active mode
  - The remaining interfaces in the VRF, including the revenue VLAN IRB interfaces, in OSPF passive mode

## Operational Differences in Enhanced OISM Mode

The main operational differences in enhanced OISM mode operation compared to regular OISM mode operation are:

- **East-west traffic from internal sources**—The ingress leaf devices forward east-west multicast source traffic on the source VLAN only to their multihoming peer leaf devices with which they share at least one Ethernet segment. For all other OISM leaf devices, the ingress leaf devices route the source traffic only on the supplemental bridge domain (SBD), even if those other devices host the source VLAN. Then each leaf device locally routes the traffic from the SBD to the destination VLAN.

  Conversely, regular OISM sends multicast traffic from internal sources only on the source VLAN. Then each leaf device locally forwards the traffic on the source VLAN or routes the traffic to the destination VLAN. Only the border leaf devices route traffic on the SBD; the border leaf devices use the SBD to support multicast flows from external sources to receivers inside the EVPN fabric.

  **NOTE**: Enhanced OISM, like regular OISM, requires you to enable IGMP snooping or MLD snooping, so the ingress leaf device for a multicast flow sends the traffic only toward other OISM leaf devices with receivers that subscribed to (sent an IGMP or MLD join for) that flow.

- **North-south traffic from internal sources toward external receivers**—The ingress leaf devices generate EVPN Type 10 Selective P-router Multicast Service Interface (S-PMSI) Auto-Discovery (A-D) routes for internal multicast (S,G) sources and groups.

  The OISM border leaf devices act as PIM EVPN gateway (PEG) devices to connect to external multicast sources and receivers. The PEG devices need to perform PIM source registration only for multicast sources inside the EVPN network, so they look for and only do PIM registration for the sources in advertised S-PMSI A-D routes.

- **Enhanced OISM limitation for data packets with a time to live (TTL) of 1**—Enhanced OISM routes most multicast traffic on the SBD rather than on the source VLAN (even if the destination device hosts the source VLAN). The TTL value on multicast data packets routed to the SBD and then to the destination VLAN will have the packet TTL decremented more than once. As a result, packets with TTL=1 won't reach the receivers. This limitation applies to traffic for any multicast groups other than 224.0.0.0/24 (for IPv4 multicast) and ff02::/16 (for IPv6 multicast).

For more details on the operational differences between regular OISM and enhanced OISM mode, see the following pages in the EVPN User Guide:

- *Overview of Enhanced OISM*
- *How Enhanced OISM Works*

For full details on all OISM concepts, components, configuration and operation, see *Optimized Inter-Subnet Multicast in EVPN Networks*.

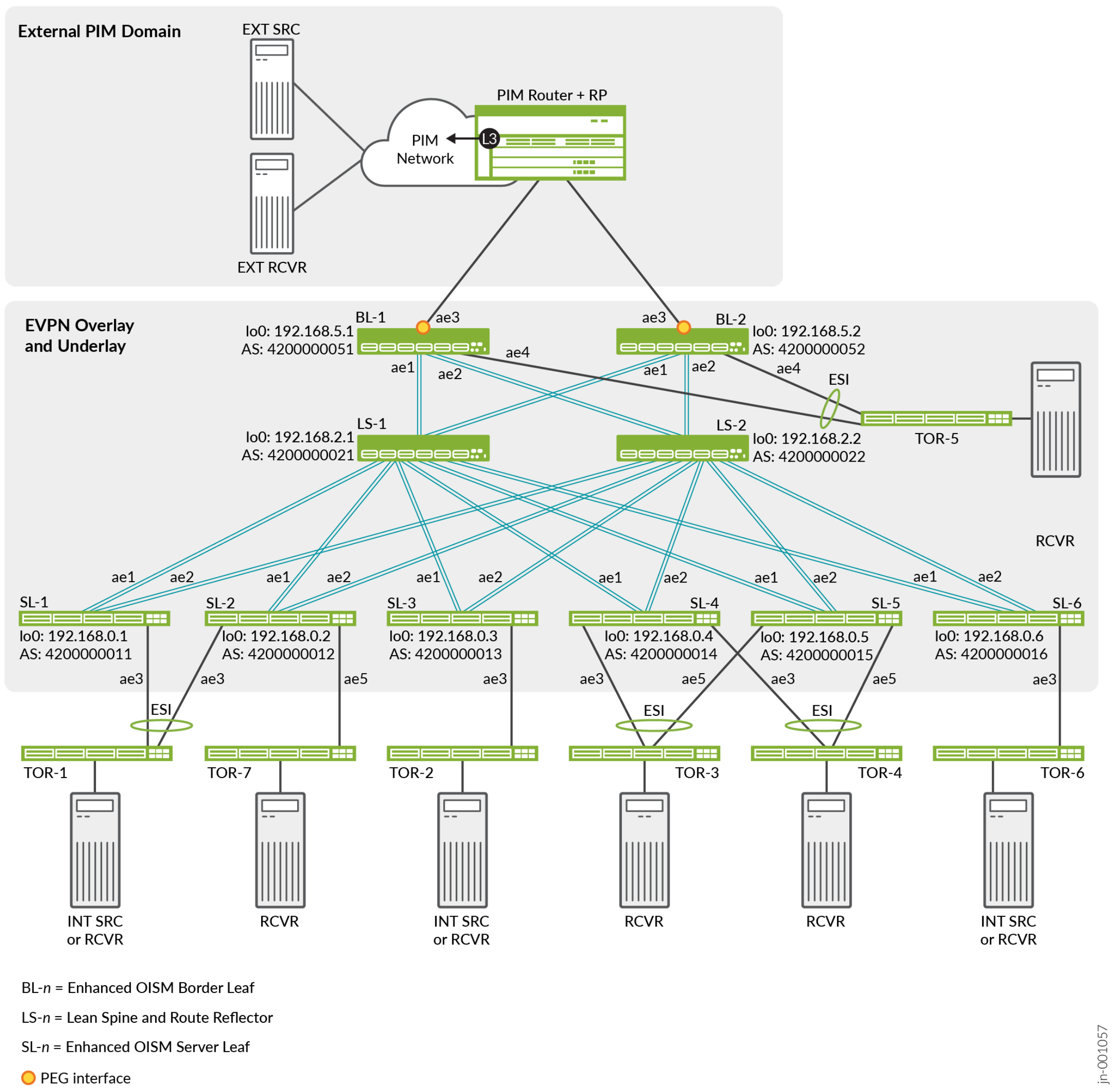
## Configure and Verify the EBGP Underlay and EBGP Overlay Peering

**IN THIS SECTION**

In the EVPN-VXLAN fabric configuration for regular OISM in "Optimized Intersubnet Multicast (OISM) with Assisted Replication (AR) for Edge-Routed Bridging Overlays" on page 509, we configure the underlay peering with EBGP and the overlay peering with IBGP. However, in this example for enhanced OISM, the EVPN-VXLAN reference architecture test environment uses EBGP for both the underlay peering and the overlay peering. See Figure 93 on page 582.

**NOTE**: We support both IPv4 and IPv6 multicast data traffic over the IPv4 underlay and overlay in this configuration.

**Figure 93: Enhanced OISM EVPN-VXLAN Fabric—EBGP Underlay and EBGP Overlay Peering**



All of the OISM server leaf devices (SL-*n*) and border leaf devices (BL-*n*) peer with the lean spine devices LS-1 and LS-2 in a full mesh spine and leaf configuration. See:

- Figure 93 on page 582 for the EVPN core interface names, device loopback addresses, and AS numbers.
- Table 20 on page 583 for the corresponding subnet addresses for the underlay peering interfaces—all interface addresses are .0 on the spine device side and .1 on the leaf device side.

**Table 20: Enhanced OISM Interfaces for Spine and Leaf Peering**

| Leaf Device | Spine Device | Interface | Underlay Subnet |
|---|---|---|---|
| SL-1<br>192.168.0.1 | LS-1 | ae1 | 172.16.1.0/31 |
| | LS-2 | ae2 | 172.16.3.0/31 |
| SL-2<br>192.168.0.2 | LS-1 | ae1 | 172.16.2.0/31 |
| | LS-2 | ae2 | 172.16.4.0/31 |
| SL-3<br>192.168.0.3 | LS-1 | ae1 | 172.16.5.0/31 |
| | LS-2 | ae2 | 172.16.6.0/31 |
| SL-4<br>192.168.0.4 | LS-1 | ae1 | 172.16.7.0/31 |
| | LS-2 | ae2 | 172.16.9.0/31 |
| SL-5<br>192.168.0.5 | LS-1 | ae1 | 172.16.8.0/31 |
| | LS-2 | ae2 | 172.16.10.0/31 |
| SL-6<br>192.168.0.6 | LS-1 | ae1 | 172.16.15.0/31 |
| | LS-2 | ae2 | 172.16.16.0/31 |
| BL-1<br>192.168.5.1 | LS-1 | ae1 | 172.16.11.0/31 |
| | LS-2 | ae2 | 172.16.13.0/31 |
| BL-2<br>192.168.5.2 | LS-1 | ae1 | 172.16.12.0/31 |

Table 20: Enhanced OISM Interfaces for Spine and Leaf Peering *(Continued)*

| Leaf Device | Spine Device | Interface | Underlay Subnet |
|---|---|---|---|
| | LS-2 | ae2 | 172.16.14.0/31 |

## Configure EBGP Underlay Peering

The underlay configuration is similar to the regular OISM EBGP underlay configuration example, but the enhanced OISM test environment has a few interface subnet differences. On each leaf device, configure EBGP for the underlay peering with neighbor devices LS-1 and LS-2. For example:

**SL-1:**

```
set policy-options policy-statement underlay-clos-export term loopback from interface lo0.0
set policy-options policy-statement underlay-clos-export term loopback then accept
set protocols bgp group underlay-bgp type external
set protocols bgp group underlay-bgp export underlay-clos-export
set protocols bgp group underlay-bgp local-as 4200000011
set protocols bgp group underlay-bgp multipath multiple-as
set protocols bgp group underlay-bgp bfd-liveness-detection minimum-interval 1000
set protocols bgp group underlay-bgp bfd-liveness-detection multiplier 3
set protocols bgp group underlay-bgp bfd-liveness-detection session-mode automatic
set protocols bgp group underlay-bgp neighbor 172.16.1.0 peer-as 4200000021
set protocols bgp group underlay-bgp neighbor 172.16.3.0 peer-as 4200000022
set protocols bgp log-updown
set protocols bgp graceful-restart
```

Substitute the corresponding interface IP addresses and AS numbers when you configure the underlay on the other OISM leaf devices.

## Configure EBGP Overlay Peering

On each leaf device, configure the EBGP overlay peering with neighbor devices LS-1 and LS-2. For example:

**SL-1:**

```
set protocols bgp group overlay-ebgp type external
set protocols bgp group overlay-ebgp multihop no-nexthop-change
set protocols bgp group overlay-ebgp local-address 192.168.0.1
set protocols bgp group overlay-ebgp family evpn signaling
set protocols bgp group overlay-ebgp local-as 4200000011
set protocols bgp group overlay-ebgp multipath multiple-as
set protocols bgp group overlay-ebgp bfd-liveness-detection minimum-interval 4000
set protocols bgp group overlay-ebgp bfd-liveness-detection multiplier 3
set protocols bgp group overlay-ebgp bfd-liveness-detection session-mode automatic
set protocols bgp group overlay-ebgp neighbor 192.168.2.1 peer-as 4200000021
set protocols bgp group overlay-ebgp neighbor 192.168.2.2 peer-as 4200000022
set protocols bgp group overlay-ebgp vpn-apply-export
```

Substitute the corresponding device loopback addresses and AS numbers when you configure the overlay on the other OISM leaf devices.

## Verify Underlay and Overlay Peering

To verify the EBGP underlay and overlay peering on SL-1 (lo0: 192.168.0.1, AS 4200000011), for example, look for the following in the output from the `show bgp neighbor` command:

- Underlay peering on:
  - LS-1: Subnet 172.16.1.0/31
  - LS-2: Subnet 172.16.3.0/31
- Overlay peering with:
  - LS-1: lo0 192.168.2.1, AS 4200000021
  - LS-2: lo0 192.168.2.2, AS 4200000022

```
user@SL-1> show bgp neighbor | grep Peer
Peer: 172.16.1.0+179 AS 4200000021 Local: 172.16.1.1+58263 AS 4200000011
  Options: <AuthKey GracefulRestart LogUpDown PeerAS Multipath LocalAS Refresh>
  Peer ID: 192.168.2.1     Local ID: 192.168.0.1       Active Holdtime: 90
  Keepalive Interval: 30         Group index: 0    Peer index: 0    SNMP index: 0
  NLRI for restart configured on peer: inet-unicast
  NLRI advertised by peer: inet-unicast
  Peer supports Refresh capability (2)
  Restart time configured on the peer: 120
  Stale routes from peer are kept for: 300
  Peer does not support Restarter functionality
  Restart flag received from the peer: Notification
  Peer does not support LLGR Restarter functionality
  Peer supports 4 byte AS extension (peer-as 4200000021)
  Peer does not support Addpath
Peer: 172.16.3.0+179 AS 4200000022 Local: 172.16.3.1+57766 AS 4200000011
  Options: <AuthKey GracefulRestart LogUpDown PeerAS Multipath LocalAS Refresh>
  Peer ID: 192.168.2.2     Local ID: 192.168.0.1       Active Holdtime: 90
  Keepalive Interval: 30         Group index: 0    Peer index: 1    SNMP index: 1
  NLRI for restart configured on peer: inet-unicast
  NLRI advertised by peer: inet-unicast
  Peer supports Refresh capability (2)
  Restart time configured on the peer: 120
  Stale routes from peer are kept for: 300
  Peer does not support Restarter functionality
  Restart flag received from the peer: Notification
  Peer does not support LLGR Restarter functionality
  Peer supports 4 byte AS extension (peer-as 4200000022)
  Peer does not support Addpath
Peer: 192.168.2.1+51349 AS 4200000021 Local: 192.168.0.1+179 AS 4200000011
  Options: <Multihop NoNextHopChange LocalAddress AuthKey GracefulRestart LogUpDown
AddressFamily PeerAS Multipath LocalAS Rib-group Refresh>
  Peer ID: 192.168.2.1     Local ID: 192.168.0.1       Active Holdtime: 90
  Keepalive Interval: 30         Group index: 1    Peer index: 0    SNMP index: 2
  NLRI for restart configured on peer: evpn
  NLRI advertised by peer: evpn
  Peer supports Refresh capability (2)
  Restart time configured on the peer: 120
  Stale routes from peer are kept for: 300
  Peer does not support Restarter functionality
  Restart flag received from the peer: Notification
  Peer does not support LLGR Restarter functionality
  Peer supports 4 byte AS extension (peer-as 4200000021)
  Peer does not support Addpath
Peer: 192.168.2.2+60628 AS 4200000022 Local: 192.168.0.1+179 AS 4200000011
  Options: <Multihop NoNextHopChange LocalAddress AuthKey GracefulRestart LogUpDown
AddressFamily PeerAS Multipath LocalAS Rib-group Refresh>
  Peer ID: 192.168.2.2     Local ID: 192.168.0.1       Active Holdtime: 90
  Keepalive Interval: 30         Group index: 1    Peer index: 1    SNMP index: 3
  NLRI for restart configured on peer: evpn
  NLRI advertised by peer: evpn
  Peer supports Refresh capability (2)
  Restart time configured on the peer: 120
  Stale routes from peer are kept for: 300
  Peer does not support Restarter functionality
  Restart flag received from the peer: Notification
```

```
Peer does not support LLGR Restarter functionality
Peer supports 4 byte AS extension (peer-as 4200000022)
Peer does not support Addpath
```

Run the command on each OISM leaf devices and look for the corresponding interconnecting subnets and overlay device loopback addresses.

## Configure the EVPN MAC-VRF Instance and Enable Enhanced OISM

You configure the same EVPN MAC-VRF instance named MACVRF-1 on all OISM server leaf and OISM border leaf devices, with only a few differences for parameters that depend on which device you're configuring. In later sections for different use case configurations, you add the VLANs, corresponding IRB interfaces, and VXLAN VNI mappings instance that are specific to that use case.

1. Configure the EVPN-VXLAN MAC-VRF instance named MACVRF-1 in the same way you configure the MAC-VRF instance for regular OISM, including:

   - VLAN-aware service type.
   - Device loopback interface as the VXLAN tunnel endpoint (VTEP) source interface.
   - Enable all VNIs in the instance to extend into the EVPN BGP domain.
   - The same route target for the instance on all of the OISM leaf devices.
   - A route distinguisher on each OISM leaf device in which the first part of the value matches the device loopback IP address—see Figure 93 on page 582.

   ```
   set routing-instances MACVRF-1 instance-type mac-vrf
   set routing-instances MACVRF-1 protocols evpn encapsulation vxlan
   set routing-instances MACVRF-1 protocols evpn default-gateway no-gateway-community
   set routing-instances MACVRF-1 protocols evpn extended-vni-list all
   set routing-instances MACVRF-1 vtep-source-interface lo0.0
   set routing-instances MACVRF-1 service-type vlan-aware
   set routing-instances MACVRF-1 route-distinguisher device-loopback-IP:33301
   set routing-instances MACVRF-1 vrf-target target:33300:1
   ```

2. On each OISM leaf device, include the device's host-side access interfaces in the MACVRF-1 instance based on the topology in Figure 93 on page 582:

a. On SL-1, SL-3, and SL-6, include ae3.0:

```
set routing-instances MACVRF-1 interface ae3.0
```

b. On SL-2, SL-4, and S5, include ae3.0 and ae5.0:

```
set routing-instances MACVRF-1 interface ae3.0
set routing-instances MACVRF-1 interface ae5.0
```

c. On BL-1 and BL-2, include ae4.0:

```
set routing-instances MACVRF-1 interface ae4.0
```

**NOTE**: The border leaf devices in this example don't use ae3 on the access side, but as L3 PEG interfaces to route traffic to and from the external PIM domain (see Figure 93 on page 582).

3. Enable OISM in enhanced mode on all OISM server leaf devices and OISM border leaf devices.

```
set forwarding-options multicast-replication evpn irb enhanced-oism
```

OISM is not enabled in either regular mode or enhanced mode by default. You must explicitly enable OISM in regular mode or enhanced mode.

## Configure Platform-specific Parameters

Configure the following platform-specific settings required in scaled EVPN-VXLAN environments on all OISM server leaf or OISM border leaf devices of the indicated device types. For details, see the similar configuration step 2 for regular OISM in "Configure an OISM-Enabled EVPN MAC-VRF Instance" on page 522.

- On devices in the QFX5000 line of switches:

```
set forwarding-options evpn-vxlan shared-tunnels
```

- On QFX5120 switches:

```
set forwarding-options vxlan-routing next-hop 49152
set forwarding-options vxlan-routing interface-num 8192
```

- On QFX5130 and QFX5700 switches:

```
set system packet-forwarding-options forwarding-profile host-profile
```

- On QFX5130 and QFX5700 switches in the EVPN MAC-VRF instance MACVRF-1:

```
set routing-instances MACVRF-1 multicast-snooping-options oism conserve-mcast-routes-in-pfe
```

## Use Case #1: Internal Source to Internal Receivers (Including Multihoming Peer) with IGMPv3—SSM

In use case #1, we configure the topology in Figure 94 on page 590 with a tenant VRF called VRF-1. This use case includes:

- An internal multicast source and internal multicast receivers using IGMPv3 with intra-VLAN and inter-VLAN multicast flows.
- A single-homed receiver behind a server leaf device that is a multihoming peer of the ingress server leaf device, so:
  - The multihoming peer devices must have same set of revenue VLANs configured (even if both devices don't host all of the VLANs).
  - The ingress server leaf device forwards multicast traffic on the source VLAN (not the SBD) to the multihoming peer device toward the receiver behind that peer device.
  - The ingress server leaf device routes the multicast traffic on the SBD toward the receivers on all of the other OISM leaf devices.

**Figure 94: Enhanced OISM Use Case #1 Topology—IGMPv3 with Source Behind Multihoming Peers SL-1 and SL-2**

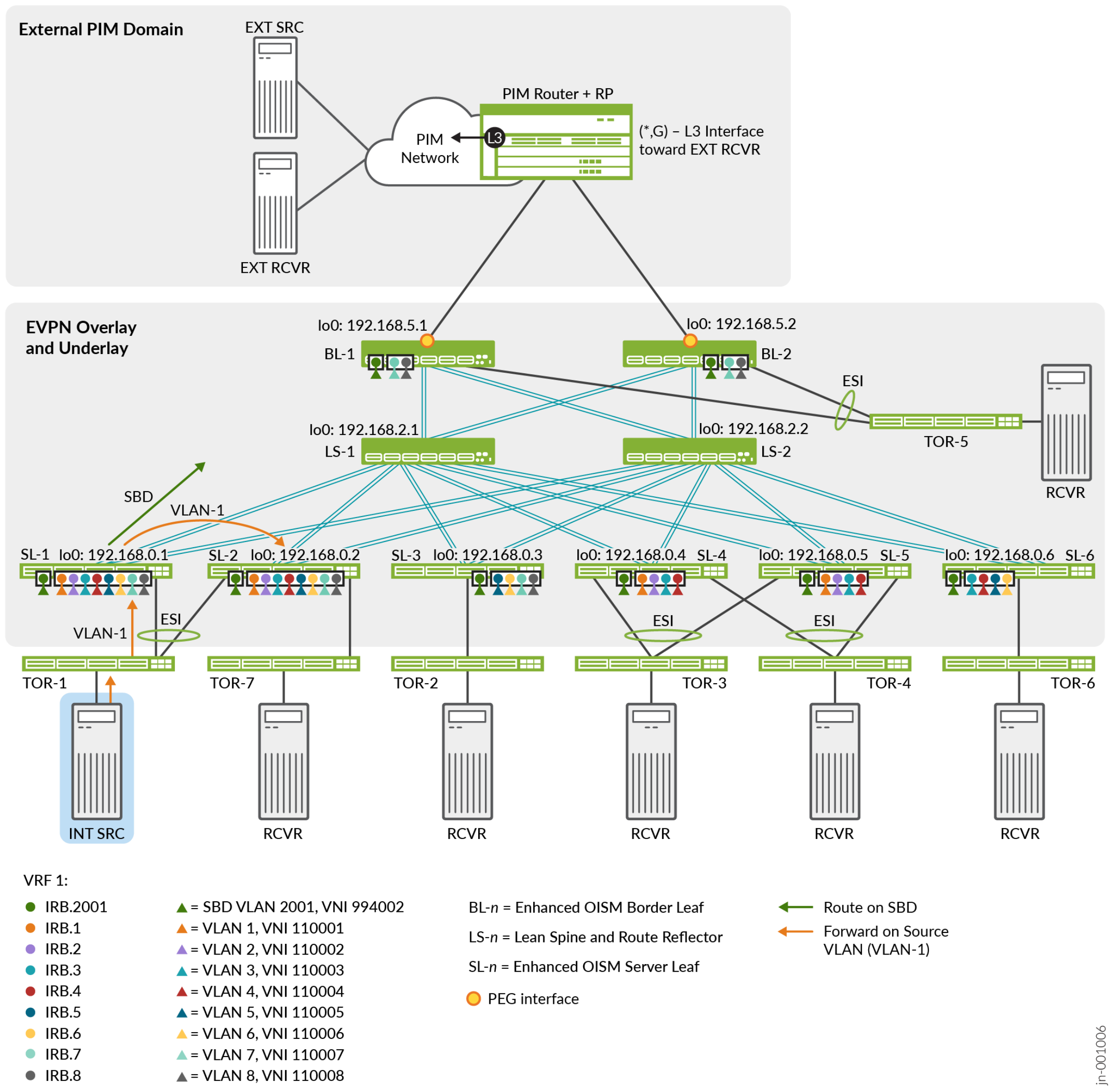


Table 21 on page 591 describes the multicast groups, device roles, configured VLANs, VXLAN VNI mappings for the VLANs, and the corresponding IRB interfaces for each VLAN.

**Table 21: Use Case #1 Elements for Internal Multicast Flow with Multihoming Peer and IGMPv3**

| Role | Device | Configured Revenue VLANs | Configured IRB Interfaces | VXLAN VNI Mappings |
|---|---|---|---|---|
| SBD for VRF-1 on all enhanced OISM leaf devices: VLAN-2001, irb.2001, VNI 994002<br>Multicast source VLAN: VLAN-1, Source Host IP address: 10.0.1.12<br>IGMPv3—SSM multicast groups: 233.252.0.1 – 233.252.0.3 for intra-VLAN (L2), and 233.252.0.101 – 233.252.0.103 for inter-VLAN (L3) | | | | |
| Source | TOR-1—Multihomed to SL-1[1] and SL-2[1] | VLAN-1 - VLAN-8 | irb.1 - irb.8 | VNI 110001 - VNI 110008 |
| Receivers | TOR-7—Single-homed to SL-2[1] | VLAN-1 - VLAN-8 | irb.1 - irb.8 | VNI 110001 - VNI 110008 |
| | TOR-2—Single-homed to SL-3 | VLAN-5- VLAN-8 | irb.5- irb.8 | VNI 110005- VNI 110008 |
| | TOR-3—Multihomed to SL-4[2] and SL-5[2] | VLAN-1 - VLAN-4 | irb.1 - irb.4 | VNI 110001 - VNI 110004 |
| | TOR-4—Multihomed to SL-4[2] and SL-5[2] | VLAN-1 - VLAN-4 | irb.1 - irb.4 | VNI 110001 - VNI 110004 |
| | TOR-5—Multihomed to BL-1[3] and BL-2[3] | VLAN-7 - VLAN-8 | irb.7 - irb.8 | VNI 110007 - VNI 110008 |
| | TOR-6—Single-homed to SL-6 | VLAN-3 - VLAN-6 | irb.3 - irb.6 | VNI 110003 - VNI 110006 |

Table 21: Use Case #1 Elements for Internal Multicast Flow with Multihoming Peer and IGMPv3 *(Continued)*

| Role | Device | Configured Revenue VLANs | Configured IRB Interfaces | VXLAN VNI Mappings |
|---|---|---|---|---|

[1] SL-1 and SL-2 are multihoming peers, so we configure the same revenue VLANs on SL-1 and SL-2.

[2] SL-4 and SL-5 are multihoming peers, so we configure the same revenue VLANs on SL-4 and SL-5.

[3] BL-1 and BL-2 are multihoming peers, so we configure the same revenue VLANs on BL-1 and BL-2.

## Configure Use Case #1: Internal Source and Receivers with Multihoming Peer Receiver and IGMPv3—SSM

Configure the revenue VLANs, SBD, tenant VRF, and multicast protocols specific to the use case in "Use Case #1: Internal Source to Internal Receivers (Including Multihoming Peer) with IGMPv3—SSM" on page 589.

1. On each OISM leaf device, in the MACVRF-1 instance, configure the hosted revenue VLANs and their corresponding IRB interfaces and VNI mappings.

   **SL-1 and SL-2 (multihoming peer devices)—VLANs 1 through 8:**

```
set routing-instances MACVRF-1 vlans VLAN-1 vlan-id 1
set routing-instances MACVRF-1 vlans VLAN-1 l3-interface irb.1
set routing-instances MACVRF-1 vlans VLAN-1 vxlan vni 110001
set routing-instances MACVRF-1 vlans VLAN-2 vlan-id 2
set routing-instances MACVRF-1 vlans VLAN-2 l3-interface irb.2
set routing-instances MACVRF-1 vlans VLAN-2 vxlan vni 110002
set routing-instances MACVRF-1 vlans VLAN-3 vlan-id 3
set routing-instances MACVRF-1 vlans VLAN-3 l3-interface irb.3
set routing-instances MACVRF-1 vlans VLAN-3 vxlan vni 110003
set routing-instances MACVRF-1 vlans VLAN-4 vlan-id 4
set routing-instances MACVRF-1 vlans VLAN-4 l3-interface irb.4
set routing-instances MACVRF-1 vlans VLAN-4 vxlan vni 110004
set routing-instances MACVRF-1 vlans VLAN-5 vlan-id 5
set routing-instances MACVRF-1 vlans VLAN-5 l3-interface irb.5
set routing-instances MACVRF-1 vlans VLAN-5 vxlan vni 110005
set routing-instances MACVRF-1 vlans VLAN-6 vlan-id 6
set routing-instances MACVRF-1 vlans VLAN-6 l3-interface irb.6
set routing-instances MACVRF-1 vlans VLAN-6 vxlan vni 110006
set routing-instances MACVRF-1 vlans VLAN-7 vlan-id 7
set routing-instances MACVRF-1 vlans VLAN-7 l3-interface irb.7
set routing-instances MACVRF-1 vlans VLAN-7 vxlan vni 110007
set routing-instances MACVRF-1 vlans VLAN-8 vlan-id 8
set routing-instances MACVRF-1 vlans VLAN-8 l3-interface irb.8
set routing-instances MACVRF-1 vlans VLAN-8 vxlan vni 110008
```

SL-3—VLANs 5 through 8:

```
set routing-instances MACVRF-1 vlans VLAN-5 vlan-id 5
set routing-instances MACVRF-1 vlans VLAN-5 l3-interface irb.5
set routing-instances MACVRF-1 vlans VLAN-5 vxlan vni 110005
set routing-instances MACVRF-1 vlans VLAN-6 vlan-id 6
set routing-instances MACVRF-1 vlans VLAN-6 l3-interface irb.6
set routing-instances MACVRF-1 vlans VLAN-6 vxlan vni 110006
set routing-instances MACVRF-1 vlans VLAN-7 vlan-id 7
set routing-instances MACVRF-1 vlans VLAN-7 l3-interface irb.7
set routing-instances MACVRF-1 vlans VLAN-7 vxlan vni 110007
set routing-instances MACVRF-1 vlans VLAN-8 vlan-id 8
set routing-instances MACVRF-1 vlans VLAN-8 l3-interface irb.8
set routing-instances MACVRF-1 vlans VLAN-8 vxlan vni 110008
```

SL-4 and SL-5 (multihoming peer devices)—VLANs 1 through 4:

```
set routing-instances MACVRF-1 vlans VLAN-1 vlan-id 1
se routing-instances MACVRF-1 vlans VLAN-1 l3-interface irb.1
set routing-instances MACVRF-1 vlans VLAN-1 vxlan vni 110001
set routing-instances MACVRF-1 vlans VLAN-2 vlan-id 2
set routing-instances MACVRF-1 vlans VLAN-2 l3-interface irb.2
set routing-instances MACVRF-1 vlans VLAN-2 vxlan vni 110002
set routing-instances MACVRF-1 vlans VLAN-3 vlan-id 3
set routing-instances MACVRF-1 vlans VLAN-3 l3-interface irb.3
set routing-instances MACVRF-1 vlans VLAN-3 vxlan vni 110003
set routing-instances MACVRF-1 vlans VLAN-4 vlan-id 4
set routing-instances MACVRF-1 vlans VLAN-4 l3-interface irb.4
set routing-instances MACVRF-1 vlans VLAN-4 vxlan vni 110004
```

**SL-6—VLANs 3 through 6:**

```
set routing-instances MACVRF-1 vlans VLAN-3 vlan-id 3
set routing-instances MACVRF-1 vlans VLAN-3 l3-interface irb.3
set routing-instances MACVRF-1 vlans VLAN-3 vxlan vni 110003
set routing-instances MACVRF-1 vlans VLAN-4 vlan-id 4
set routing-instances MACVRF-1 vlans VLAN-4 l3-interface irb.4
set routing-instances MACVRF-1 vlans VLAN-4 vxlan vni 110004
set routing-instances MACVRF-1 vlans VLAN-5 vlan-id 5
set routing-instances MACVRF-1 vlans VLAN-5 l3-interface irb.5
set routing-instances MACVRF-1 vlans VLAN-5 vxlan vni 110005
set routing-instances MACVRF-1 vlans VLAN-6 vlan-id 6
set routing-instances MACVRF-1 vlans VLAN-6 l3-interface irb.6
set routing-instances MACVRF-1 vlans VLAN-6 vxlan vni 110006
```

**BL-1 and BL-2 (multihoming peer devices)—VLANs 7 and 8:**

```
set routing-instances MACVRF-1 vlans VLAN-7 vlan-id 7
set routing-instances MACVRF-1 vlans VLAN-7 l3-interface irb.7
set routing-instances MACVRF-1 vlans VLAN-7 vxlan vni 110007
set routing-instances MACVRF-1 vlans VLAN-8 vlan-id 8
set routing-instances MACVRF-1 vlans VLAN-8 l3-interface irb.8
set routing-instances MACVRF-1 vlans VLAN-8 vxlan vni 110008
```

2. On all OISM leaf devices, in the MACVRF-1 instance in this use case, configure the SBD VLAN, corresponding IRB interface, and VNI mapping.

```
set routing-instances MACVRF-1 vlans VLAN-2001 vlan-id 2001
set routing-instances MACVRF-1 vlans VLAN-2001 l3-interface irb.2001
set routing-instances MACVRF-1 vlans VLAN-2001 vxlan vni 994002
```

3. On each device, configure the hosted revenue VLAN IRB interfaces and the SBD IRB interface in this use case as L3 gateways with IPv4 and IPv6 dual stack addresses.

   In this example, we configure the interfaces as gateways using a unique IRB IP address with a virtual gateway address (VGA).

   **On SL-1 for irb.1 (corresponds to revenue VLAN 1), for example:**

```
set interfaces irb unit 1 virtual-gateway-accept-data
set interfaces irb unit 1 no-auto-virtual-gateway-esi
set interfaces irb unit 1 family inet address 10.0.1.243/24 preferred
set interfaces irb unit 1 family inet address 10.0.1.243/24 virtual-gateway-address 10.0.1.254
set interfaces irb unit 1 family inet6 address 2001:db8::10:0:1:243/112 preferred
set interfaces irb unit 1 family inet6 address 2001:db8::10:0:1:243/112 virtual-gateway-
address 2001:db8::10:0:1:254
set interfaces irb unit 1 virtual-gateway-v4-mac 00:00:5e:00:00:04
set interfaces irb unit 1 virtual-gateway-v6-mac 00:00:5e:00:00:04
```

Use the same configuration on each leaf device to configure the gateway settings for the IRB interfaces for the hosted revenue VLANs and the SBD IRB interface, but substitute the following values to use unique addresses per IRB interface and per leaf device:

**NOTE**: The SBD IRB interface VGA is common to all of the leaf devices for the VRF in this use case.

**Table 22: Use Case #1 IRB Addresses and Virtual Gateway Addresses**

| Leaf Device | IRB Interface Unit# | IRB IPv4 Address<br><br>IPv4 VGA | IRB IPv6 Address<br><br>IPv6 VGA |
|---|---|---|---|
| SL-1 | Revenue VLANs—1 through 8 | 10.0.*unit#*.243/24<br><br>10.0.*unit#*.254 | 2001:db8::10:0:*unit#*:243/112<br><br>2001:db8::10:0:*unit#*:254 |
| | SBD—2001 | 10.20.1.243/24<br><br>10.20.1.254 | 2001:db8::10:0:7d1:243/112<br><br>2001:db8::10:0:7d1:254 |
| SL-2 | Revenue VLANs—1 through 8 | 10.0.*unit#*.244/24<br><br>10.0.*unit#*.254 | 2001:db8::10:0:*unit#*:244/112<br><br>2001:db8::10:0:*unit#*:254 |
| | SBD—2001 | 10.20.1.244/24<br><br>10.20.1.254 | 2001:db8::10:0:7d1:244/112<br><br>2001:db8::10:0:7d1:254 |
| SL-3 | Revenue VLANs—5 through 8 | 10.0.*unit#*.245/24<br><br>10.0.*unit#*.254 | 2001:db8::10:0:*unit#*:245/112<br><br>2001:db8::10:0:*unit#*:254 |

**Table 22: Use Case #1 IRB Addresses and Virtual Gateway Addresses *(Continued)***

| Leaf Device | IRB Interface Unit# | IRB IPv4 Address<br><br>IPv4 VGA | IRB IPv6 Address<br><br>IPv6 VGA |
|---|---|---|---|
| | SBD—2001 | 10.20.1.245/24<br><br>10.20.1.254 | 2001:db8::10:0:7d1:245/112<br><br>2001:db8::10:0:7d1:254 |
| SL-4 | Revenue VLANs—1 through 4 | 10.0.*unit#*.246/24<br><br>10.0.*unit#*.254 | 2001:db8::10:0:*unit#*:246/112<br><br>2001:db8::10:0:*unit#*:254 |
| | SBD—2001 | 10.20.1.246/24<br><br>10.20.1.254 | 2001:db8::10:0:7d1:246/112<br><br>2001:db8::10:0:7d1:254 |
| SL-5 | Revenue VLANs—1 through 4 | 10.0.*unit#*.247/24<br><br>10.0.*unit#*.254 | 2001:db8::10:0:*unit#*:247/112<br><br>2001:db8::10:0:*unit#*:254 |
| | SBD—2001 | 10.20.1.247/24<br><br>10.20.1.254 | 2001:db8::10:0:7d1:247/112<br><br>2001:db8::10:0:7d1:254 |
| SL-6 | Revenue VLANs—3 through 6 | 10.0.*unit#*.248/24<br><br>10.0.*unit#*.254 | 2001:db8::10:0:*unit#*:248/112<br><br>2001:db8::10:0:*unit#*:254 |
| | SBD—2001 | 10.20.1.248/24<br><br>10.20.1.254 | 2001:db8::10:0:7d1:248/112<br><br>2001:db8::10:0:7d1:254 |
| BL-1 | Revenue VLANs—7 and 8 | 10.0.*unit#*.241/24<br><br>10.0.*unit#*.254 | 2001:db8::10:0:*unit#*:241/112<br><br>2001:db8::10:0:*unit#*:254 |
| | SBD—2001 | 10.20.1.241/24<br><br>10.20.1.254 | 2001:db8::10:0:7d1:241/112<br><br>2001:db8::10:0:7d1:254 |

**Table 22: Use Case #1 IRB Addresses and Virtual Gateway Addresses *(Continued)***

| Leaf Device | IRB Interface Unit# | IRB IPv4 Address<br><br>IPv4 VGA | IRB IPv6 Address<br><br>IPv6 VGA |
|---|---|---|---|
| BL-2 | Revenue VLANs—7 and 8 | 10.0.*unit#*.242/24<br><br>10.0.*unit#*.254 | 2001:db8::10:0:*unit#*:242/112<br><br>2001:db8::10:0:*unit#*:254 |
| | SBD—2001 | 10.20.1.242/24<br><br>10.20.1.254 | 2001:db8::10:0:7d1:242/112<br><br>2001:db8::10:0:7d1:254 |

4. This use case tests IGMPv3 SSM flows, so on each OISM leaf device, enable IGMP with the `version 3` option on the IRB interfaces for the revenue VLANs the device hosts, and for the SBD.

   **All OISM leaf devices—SBD:**

   ```
   set protocols igmp interface irb.2001 version 3
   ```

   **SL-1 and SL-2 (multihoming peer devices)—irb.1 through irb.8:**

   ```
   set protocols igmp interface irb.1 version 3
   set protocols igmp interface irb.2 version 3
   set protocols igmp interface irb.3 version 3
   set protocols igmp interface irb.4 version 3
   set protocols igmp interface irb.5 version 3
   set protocols igmp interface irb.6 version 3
   set protocols igmp interface irb.7 version 3
   set protocols igmp interface irb.8 version 3
   ```

   **SL-3—irb.5 through irb.8:**

   ```
   set protocols igmp interface irb.5 version 3
   set protocols igmp interface irb.6 version 3
   set protocols igmp interface irb.7 version 3
   set protocols igmp interface irb.8 version 3
   ```

**SL-4 and SL-5 (multihoming peer devices)—irb.1 through irb.4:**

```
set protocols igmp interface irb.1 version 3
set protocols igmp interface irb.2 version 3
set protocols igmp interface irb.3 version 3
set protocols igmp interface irb.4 version 3
```

**SL-6—irb.3 through irb.6:**

```
set protocols igmp interface irb.3 version 3
set protocols igmp interface irb.4 version 3
set protocols igmp interface irb.5 version 3
set protocols igmp interface irb.6 version 3
```

**BL-1 and BL-2 (multihoming peer devices)—irb.7 and irb.8:**

```
set protocols igmp interface irb.7 version 3
set protocols igmp interface irb.8 version 3
```

5. On each OISM leaf device, enable IGMP snooping for IGMP version 3 in the MACVRF-1 instance for the revenue VLANs the device hosts and the SBD.

   In EVPN-VXLAN fabrics, we support IGMPv2 traffic with ASM reports only. We support IGMPv3 traffic with SSM reports only. As a result, when you enable IGMP snooping for IGMPv3 traffic, you must include the SSM-specific *evpn-ssm-reports-only* configuration option. See *Supported IGMP or MLD Versions and Group Membership Report Modes* for more on ASM and SSM support with EVPN-VXLAN.

   **All OISM leaf devices—SBD:**

```
set routing-instances MACVRF-1 protocols igmp-snooping vlan VLAN-2001 evpn-ssm-reports-only
```

   **SL-1 and SL-2—VLANs 1 through 8:**

```
set routing-instances MACVRF-1 protocols igmp-snooping vlan VLAN-1 evpn-ssm-reports-only
set routing-instances MACVRF-1 protocols igmp-snooping vlan VLAN-2 evpn-ssm-reports-only
set routing-instances MACVRF-1 protocols igmp-snooping vlan VLAN-3 evpn-ssm-reports-only
set routing-instances MACVRF-1 protocols igmp-snooping vlan VLAN-4 evpn-ssm-reports-only
set routing-instances MACVRF-1 protocols igmp-snooping vlan VLAN-5 evpn-ssm-reports-only
set routing-instances MACVRF-1 protocols igmp-snooping vlan VLAN-6 evpn-ssm-reports-only
set routing-instances MACVRF-1 protocols igmp-snooping vlan VLAN-7 evpn-ssm-reports-only
set routing-instances MACVRF-1 protocols igmp-snooping vlan VLAN-8 evpn-ssm-reports-only
```

SL-3—VLANs 5 through 8:

```
set routing-instances MACVRF-1 protocols igmp-snooping vlan VLAN-5 evpn-ssm-reports-only
set routing-instances MACVRF-1 protocols igmp-snooping vlan VLAN-6 evpn-ssm-reports-only
set routing-instances MACVRF-1 protocols igmp-snooping vlan VLAN-7 evpn-ssm-reports-only
set routing-instances MACVRF-1 protocols igmp-snooping vlan VLAN-8 evpn-ssm-reports-only
```

SL-4 and SL-5—VLANs 1 through 4:

```
set routing-instances MACVRF-1 protocols igmp-snooping vlan VLAN-1 evpn-ssm-reports-only
set routing-instances MACVRF-1 protocols igmp-snooping vlan VLAN-2 evpn-ssm-reports-only
set routing-instances MACVRF-1 protocols igmp-snooping vlan VLAN-3 evpn-ssm-reports-only
set routing-instances MACVRF-1 protocols igmp-snooping vlan VLAN-4 evpn-ssm-reports-only
```

SL-6—VLANs 3 through 6:

```
set routing-instances MACVRF-1 protocols igmp-snooping vlan VLAN-3 evpn-ssm-reports-only
set routing-instances MACVRF-1 protocols igmp-snooping vlan VLAN-4 evpn-ssm-reports-only
set routing-instances MACVRF-1 protocols igmp-snooping vlan VLAN-5 evpn-ssm-reports-only
set routing-instances MACVRF-1 protocols igmp-snooping vlan VLAN-6 evpn-ssm-reports-only
```

BL-1 and BL-2—VLANs 7 and 8:

```
set routing-instances MACVRF-1 protocols igmp-snooping vlan VLAN-7 evpn-ssm-reports-only
set routing-instances MACVRF-1 protocols igmp-snooping vlan VLAN-8 evpn-ssm-reports-only
```

6. On each OISM leaf device, configure a loopback logical interface for the VRF instance. We include this interface in the VRF instance, VRF-1, in the next step.

   For the loopback logical interface configuration, we use the following conventions:

   - The logical unit number matches the VRF instance number.
   - The last octet of the interface's IP address also matches the VRF instance number.

SL-1:

```
set interfaces lo0 unit 1 family inet address 10.3.1.1/32 primary
```

SL-2:

```
set interfaces lo0 unit 1 family inet address 10.3.2.1/32 primary
```

SL-3:

```
set interfaces lo0 unit 1 family inet address 10.4.3.1/32 primary
```

SL-4:

```
set interfaces lo0 unit 1 family inet address 10.7.4.1/32 primary
```

SL-5:

```
set interfaces lo0 unit 1 family inet address 10.7.5.1/32 primary
```

SL-6:

```
set interfaces lo0 unit 1 family inet address 10.8.6.1/32 primary
```

BL-1:

```
set interfaces lo0 unit 1 family inet address 10.1.1.1/32 primary
```

BL-2:

```
set interfaces lo0 unit 1 family inet address 10.1.2.1/32 primary
```

7. Configure the tenant VRF instance named VRF-1 on each of the OISM leaf devices.

   VRF instance configuration with enhanced OISM is very similar to VRF instance configuration with regular OISM, except you include the revenue VLAN IRB interfaces corresponding only to the revenue VLANs the device hosts. You also include the following:

   - Identify the OISM SBD IRB interface associated with the VRF instance.
   - On the server leaf devices:
     - Configure an OSPF area with the SBD IRB interface in OSPF active mode. Include the OSPF `priority 0` option with the SBD IRB interface so server leaf device SBD IRB interfaces never assume the designated router (DR) or backup designated router (BDR) role in the OSPF area. Configure OSPF passive mode for the remaining interfaces in the VRF instance so they can share routes internally without forming OSPF adjacencies.
     - Configure PIM in the VRF instance in passive mode on the interfaces for all hosted revenue VLANs and the SBD. Include the PIM `accept-remote-source` option so the SBD IRB interface accepts traffic arriving on that interface as the source interface.
     - Add the following interfaces to the VRF instance:
       - The SBD IRB interface for this VRF
       - The hosted revenue VLAN IRB interfaces
       - The loopback logical interface for this VRF (you configure the loopback logical interface in Step 6 above)
   - On the border leaf devices, which act as PEG devices to route multicast traffic from internal sources to external receivers, and from external sources to internal receivers:
     - Identify the border leaf device as a PEG device in the VRF instance using the `pim-evpn-gateway` option at the `[edit routing-instances name protocols evpn oism]` hierarchy level.
     - Configure an OSPF area for the VRF instance with an export policy to share OSPF learned routes, and includes the following interfaces:
       - The interface that connects to the external PIM domain, in OSPF active mode

**NOTE**: The border leaf devices in this example use ae3 as L3 PEG interfaces to route traffic to and from the external PIM domain (see Figure 93 on page 582). Each VRF instance uses a different logical unit on the ae3 interface for that purpose to configure multiple use cases in the same topology. In this use case, VRF-1 on the border leaf devices uses ae3.0 (logical unit = *VRF instance number - 1*).

  - The SBD IRB interface for the VRF instance (irb.2001 in this case), in OSPF active mode
  - The remaining interfaces in passive mode to share internal routes without forming OSPF adjacencies

- Configure PIM in the VRF instance as follows:
  - Statically configure the address for the PIM rendezvous point (RP) in the external PIM domain.

    **NOTE**: In our test environment, we use an MX Series router as the PIM router and PIM RP in an external PIM domain. See "Configure External Multicast PIM Router and PIM RP Router" on page 538 for more on how to configure an MX Series router as a PIM router and RP with regular OISM; the steps are similar with enhanced OISM.

  - Set `distributed-dr` mode on the IRB interfaces for the revenue VLANs.
  - Enable PIM in regular mode on the interface that connects to the external PIM domain and the loopback logical interface for the VRF instance.
  - Enable PIM in regular mode on the SBD IRB interface, with the following options:
    - Bidirectional Forwarding Detection (BFD) to improve convergence time with interface issues to help avoid traffic loss.
    - The `stickydr` option to avoid designated router switchover convergence delays during reboot events.
    - The `accept-remote-source` option to enable the SBD IRB interface to accept traffic arriving on that interface as the source interface.
  - (QFX Series switches) Set the PIM `disable-packet-register` option for the VRF instance. The border leaf devices are the first-hop routers (FHRs) toward the external multicast PIM domain router and RP. However, for the PIM source registration process, QFX Series switches don't encapsulate (or decapsulate) multicast packets in PIM register messages and

don't generate the PIM register stop message. Without a PIM register stop message, the source registration process doesn't terminate properly. As a result, to avoid keeping the PIM registration process in an unpredictable state, we use this option on the border leaf devices that are QFX Series switches.

- Add the following interfaces to the VRF instance:
    - The interface that connects to the external PIM domain—ae3.0
    - The SBD IRB interface for this VRF
    - The hosted revenue VLAN IRB interfaces
    - The loopback logical interface for this VRF

Our enhanced OISM configuration also uses the following conventions in VRF instance configurations on all OISM leaf devices:

- Enable the graceful restart feature to help with OISM traffic convergence after failure events.
- Use a route target that's the same for this VRF instance on all of the leaf devices, in which the last segment of the value matches the number in the VRF instance name.
- Use a route distinguisher (RD) for this VRF instance on each leaf device, in which the first part of the RD value mirrors the device loopback (unit 0) IP address, and the second part matches the VRF instance number.

**All server leaf devices SL-1 through SL-6:**

```
set routing-instances VRF-1 instance-type vrf
set routing-instances VRF-1 routing-options graceful-restart
set routing-instances VRF-1 protocols evpn oism supplemental-bridge-domain-irb irb.2001
set routing-instances VRF-1 protocols ospf area 1.1.1.1 interface irb.2001 priority 0
set routing-instances VRF-1 protocols ospf area 1.1.1.1 interface all passive
set routing-instances VRF-1 protocols pim passive
set routing-instances VRF-1 protocols pim interface all
set routing-instances VRF-1 protocols pim interface irb.2001 accept-remote-source
set routing-instances VRF-1 protocols pim disable-packet-register
set routing-instances VRF-1 interface lo0.1
set routing-instances VRF-1 interface irb.2001
set routing-instances VRF-1 route-distinguisher device-loopback-unit-0-address:1
set routing-instances VRF-1 vrf-target target:100:1
```

**Border leaf devices BL-1 and BL-2:**

```
set policy-options policy-statement export-direct term t1 from protocol direct
set policy-options policy-statement export-direct term t1 then accept
set routing-instances VRF-1 instance-type vrf
set routing-instances VRF-1 routing-options graceful-restart
set routing-instances VRF-1 protocols evpn oism supplemental-bridge-domain-irb irb.2001
set routing-instances VRF-1 protocols evpn oism pim-evpn-gateway
set routing-instances VRF-1 protocols ospf area 1.1.1.1 interface ae3.0 mtu 9178
set routing-instances VRF-1 protocols ospf area 1.1.1.1 interface irb.2001
set routing-instances VRF-1 protocols ospf area 1.1.1.1 interface all passive
set routing-instances VRF-1 protocols ospf export export-direct
set routing-instances VRF-1 protocols pim rp static address pim-rp-IP-address
set routing-instances VRF-1 protocols pim interface irb.7 distributed-dr
set routing-instances VRF-1 protocols pim interface irb.8 distributed-dr
set routing-instances VRF-1 protocols pim interface lo0.1
set routing-instances VRF-1 protocols pim interface ae3.0
set routing-instances VRF-1 protocols pim interface irb.2001 family inet bfd-liveness-
detection minimum-interval 1000
set routing-instances VRF-1 protocols pim interface irb.2001 family inet bfd-liveness-
detection multiplier 3
set routing-instances VRF-1 protocols pim interface irb.2001 stickydr
set routing-instances VRF-1 protocols pim interface irb.2001 hello-interval 30
set routing-instances VRF-1 protocols pim interface irb.2001 accept-remote-source
set routing-instances VRF-1 protocols pim disable-packet-register
set routing-instances VRF-1 route-distinguisher device-loopback-unit-0-address:1
set routing-instances VRF-1 vrf-target target:100:1
```

Then in the VRF instance configuration on each of the leaf devices, add the IRB interfaces only for the revenue VLANs the device hosts:

**SL-1 and SL-2—irb.1 through irb.8**

```
set routing-instances VRF-1 interface irb.1
set routing-instances VRF-1 interface irb.2
set routing-instances VRF-1 interface irb.3
set routing-instances VRF-1 interface irb.4
set routing-instances VRF-1 interface irb.5
set routing-instances VRF-1 interface irb.6
set routing-instances VRF-1 interface irb.7
set routing-instances VRF-1 interface irb.8
```

SL-3—irb.5 through irb.8:

```
set routing-instances VRF-1 interface irb.5
set routing-instances VRF-1 interface irb.6
set routing-instances VRF-1 interface irb.7
set routing-instances VRF-1 interface irb.8
```

SL-4 and SL-5—irb.1 through irb.4:

```
set routing-instances VRF-1 interface irb.1
set routing-instances VRF-1 interface irb.2
set routing-instances VRF-1 interface irb.3
set routing-instances VRF-1 interface irb.4
```

SL-6—irb.3 through irb.6:

```
set routing-instances VRF-1 interface irb.3
set routing-instances VRF-1 interface irb.4
set routing-instances VRF-1 interface irb.5
set routing-instances VRF-1 interface irb.6
```

BL-1 and BL-2—irb.7 and irb.8:

```
set routing-instances VRF-1 interface irb.7
set routing-instances VRF-1 interface irb.8
```

## Verify Use Case#1: Internal Source and Receivers with Multihoming Peer Receiver and IGMPv3—SSM

Verify enhanced OISM operation for use case #1, in which the internal source is behind SL-1 and SL-2. SL-2 is a multihoming peer of SL-1, and also has an internal receiver. SL-3 through SL-6, BL-1, and BL-2 also have internal receivers. See the topology in Figure 94 on page 590, and the configuration in "Configure Use Case #1: Internal Source and Receivers with Multihoming Peer Receiver and IGMPv3—SSM" on page 592.

1. Run the `show vlans` command for the revenue VLANs on each leaf device to see the VXLAN tunnel end points (VTEPs) between the leaf devices.

With the enhanced OISM asymmetric bridge domains model, each leaf device only needs to maintain VTEPs to the other leaf devices for the VLANs they host in common.

For example, on SL-1, you see the following:

- For VLAN-1—3 VTEPs for the leaf devices that host VLAN-1 (SL-2, SL-4, and SL-5)
- For VLAN-2—3 VTEPs for the leaf devices that host VLAN-2 (SL-2, SL-4, and SL-5)
- For VLAN-3—4 VTEPs for the leaf devices that host VLAN-3 (SL-2, SL-4, SL-5, and SL-6)
- For VLAN-4—4 VTEPs for the leaf devices that host VLAN-4 (SL-2, SL-4, SL-5, and SL-6)
- For VLAN-5—3 VTEPs for the leaf devices that host VLAN-5 (SL-2, SL-3, and SL-6)
- For VLAN-6—3 VTEPs for the leaf devices that host VLAN-6 (SL-2, SL-3, and SL-6)
- For VLAN-7—4 VTEPs for the leaf devices that host VLAN-7 (SL-2, SL-3, BL-1, and BL-2)
- For VLAN-8—4 VTEPs for the leaf devices that host VLAN-8 (SL-2, SL-3, BL-1, and BL-2)

On the other devices, you see output only for the VLANs those devices host, with the corresponding VTEPs to the other leaf devices that host the same VLANs.

**SL-1 (output is similar on SL-2, which is a multihoming peer of SL-1):**

```
user@SL-1> show vlans 1

Routing instance        VLAN name           Tag          Interfaces
MACVRF-1                VLAN-1              1
                                                         ae3.0*
                                                         esi.15861*
                                                         vtep-5.32770*
                                                         vtep-5.32771*
                                                         vtep-5.32773*

user@SL-1> show vlans 2

Routing instance        VLAN name           Tag          Interfaces
MACVRF-1                VLAN-2              2
                                                         ae3.0*
                                                         esi.15861*
                                                         vtep-5.32770*
                                                         vtep-5.32771*
                                                         vtep-5.32773*
```

```
user@SL-1> show vlans 3

Routing instance        VLAN name             Tag          Interfaces
MACVRF-1                VLAN-3                3
                                                           ae3.0*
                                                           esi.15861*
                                                           vtep-5.32769*
                                                           vtep-5.32770*
                                                           vtep-5.32771*
                                                           vtep-5.32773*

user@SL-1> show vlans 4

Routing instance        VLAN name             Tag          Interfaces
MACVRF-1                VLAN-4                4
                                                           ae3.0*
                                                           esi.15861*
                                                           vtep-5.32769*
                                                           vtep-5.32770*
                                                           vtep-5.32771*
                                                           vtep-5.32773*

user@SL-1> show vlans 5

Routing instance        VLAN name             Tag          Interfaces
MACVRF-1                VLAN-5                5
                                                           ae3.0*
                                                           esi.15861*
                                                           vtep-5.32769*
                                                           vtep-5.32770*
                                                           vtep-5.32774*

user@SL-1> show vlans 6

Routing instance        VLAN name             Tag          Interfaces
MACVRF-1                VLAN-6                6
                                                           ae3.0*
                                                           esi.15861*
                                                           vtep-5.32769*
                                                           vtep-5.32770*
                                                           vtep-5.32774*

user@SL-1> show vlans 7
```

```
Routing instance        VLAN name            Tag          Interfaces
MACVRF-1                VLAN-7               7
                                                          ae3.0*
                                                          esi.15861*
                                                          esi.16553*
                                                          vtep-5.32770*
                                                          vtep-5.32772*
                                                          vtep-5.32774*
                                                          vtep-5.32775*

user@SL-1> show vlans 8

Routing instance        VLAN name            Tag          Interfaces
MACVRF-1                VLAN-8               8
                                                          ae3.0*
                                                          esi.15861*
                                                          esi.16553*
                                                          vtep-5.32770**
                                                          vtep-5.32772*
                                                          vtep-5.32774**
                                                          vtep-5.32775*
```

SL-3 (output is similar on SL-6 for the revenue VLANs SL-6 hosts):

```
user@SL-3> show vlans 1

user@SL-3> show vlans 2

user@SL-3> show vlans 3

user@SL-3> show vlans 4

user@SL-3> show vlans 5
Routing instance        VLAN name            Tag          Interfaces
MACVRF-1                VLAN-5               5
                                                          ae3.0*
                                                          vtep-4.32769*
                                                          vtep-4.32771*
                                                          vtep-4.32775*
```

```
user@SL-3> show vlans 6

Routing instance        VLAN name              Tag          Interfaces
MACVRF-1                VLAN-6                 6
                                                            ae3.0*
                                                            vtep-4.32769*
                                                            vtep-4.32771*
                                                            vtep-4.32775*

user@SL-3> show vlans 7

Routing instance        VLAN name              Tag          Interfaces
MACVRF-1                VLAN-7                 7
                                                            ae3.0*
                                                            esi.16195*
                                                            vtep-4.32769*
                                                            vtep-4.32771*
                                                            vtep-4.32774*
                                                            vtep-4.32775*

user@SL-3> show vlans 8

Routing instance        VLAN name              Tag          Interfaces
MACVRF-1                VLAN-8                 8
                                                            ae3.0*
                                                            esi.16195*
                                                            vtep-4.32769*
                                                            vtep-4.32771*
                                                            vtep-4.32774*
                                                            vtep-4.32775*
```

SL-5 (output is similar on SL-4, which is a multihoming peer of SL-5):

```
user@SL-5> show vlans 1

Routing instance        VLAN name              Tag          Interfaces
MACVRF-1                VLAN-1                 1
                                                            ae3.0*
                                                            ae5.0*
                                                            esi.43435*
                                                            vtep-133.32802*
```

```
                                                                   vtep-133.32804*
                                                                   vtep-133.32805*

user@SL-5> show vlans 2

Routing instance        VLAN name            Tag          Interfaces
MACVRF-1                VLAN-2               2
                                                                   ae3.0*
                                                                   ae5.0*
                                                                   vtep-133.32802*
                                                                   vtep-133.32804*
                                                                   vtep-133.32805*

user@SL-5> show vlans 3

Routing instance        VLAN name            Tag          Interfaces
MACVRF-1                VLAN-3               3
                                                                   ae3.0*
                                                                   ae5.0*
                                                                   vtep-133.32802*
                                                                   vtep-133.32804*
                                                                   vtep-133.32805*
                                                                   vtep-133.32808*

user@SL-5> show vlans 4

Routing instance        VLAN name            Tag          Interfaces
MACVRF-1                VLAN-4               4
                                                                   ae3.0*
                                                                   ae5.0*
                                                                   vtep-133.32802*
                                                                   vtep-133.32804*
                                                                   vtep-133.32805*
                                                                   vtep-133.32808*

user@SL-5> show vlans 5

user@SL-5> show vlans 6

user@SL-5> show vlans 7
```

```
user@SL-5> show vlans 8
```

2. Run the `show vlans` command for the SBD (VLAN-2001) on each leaf device to see the SBD VTEPs between the leaf devices.

   You should see VTEPs to each of the other leaf devices (seven VTEPs in this case). We show output only for SL-2; the results should be similar on all of the leaf devices.

   SL-2:

```
user@SL-2> show vlans 2001

Routing instance        VLAN name             Tag          Interfaces
MACVRF-1                VLAN-2001             2001
                                                           vtep-5.32769*
                                                           vtep-5.32770*
                                                           vtep-5.32771*
                                                           vtep-5.32772*
                                                           vtep-5.32773*
                                                           vtep-5.32774*
                                                           vtep-5.32775*
```

3. Run the `show interfaces irb.`*`unit#`*` terse` command on each of the leaf devices to verify the revenue VLAN IRB interfaces are up, and the SBD IRB interface is up.

   **SL-2 (output is similar on SL-1):**

```
user@SL-2> show interfaces irb.1 terse
Interface               Admin Link Proto    Local                 Remote
irb.1                   up    up   inet     10.0.1.244/24
                                            10.0.1.254/24
                                   inet6    2001:db8::10:0:1:244/112
                                            2001:db8::10:0:1:254/112
                                            fe80::8243:3f00:1e2:5d80/64

user@SL-2> show interfaces irb.2 terse
Interface               Admin Link Proto    Local                 Remote
irb.2                   up    up   inet     10.0.2.244/24
                                            10.0.2.254/24
                                   inet6    2001:db8::10:0:2:244/112
                                            2001:db8::10:0:2:254/112
                                            fe80::8243:3f00:2e2:5d80/64
```

```
user@SL-2> show interfaces irb.3 terse
Interface               Admin Link Proto    Local                 Remote
irb.3                   up    up   inet     10.0.3.244/24
                                            10.0.3.254/24
                                   inet6    2001:db8::10:0:3:244/112
                                            2001:db8::10:0:3:254/112
                                            fe80::8243:3f00:3e2:5d80/64

user@SL-2> show interfaces irb.4 terse
Interface               Admin Link Proto    Local                 Remote
irb.4                   up    up   inet     10.0.4.244/24
                                            10.0.4.254/24
                                   inet6    2001:db8::10:0:4:244/112
                                            2001:db8::10:0:4:254/112
                                            fe80::8243:3f00:4e2:5d80/64

user@SL-2> show interfaces irb.5 terse
Interface               Admin Link Proto    Local                 Remote
irb.5                   up    up   inet     10.0.5.244/24
                                            10.0.5.254/24
                                   inet6    2001:db8::10:0:5:244/112
                                            2001:db8::10:0:5:254/112
                                            fe80::8243:3f00:5e2:5d80/64

user@SL-2> show interfaces irb.6 terse
Interface               Admin Link Proto    Local                 Remote
irb.6                   up    up   inet     10.0.6.244/24
                                            10.0.6.254/24
                                   inet6    2001:db8::10:0:6:244/112
                                            2001:db8::10:0:6:254/112
                                            fe80::8243:3f00:6e2:5d80/64

user@SL-2> show interfaces irb.7 terse
Interface               Admin Link Proto    Local                 Remote
irb.7                   up    up   inet     10.0.7.244/24
                                            10.0.7.254/24
                                   inet6    2001:db8::10:0:7:244/112
                                            2001:db8::10:0:7:254/112
                                            fe80::8243:3f00:7e2:5d80/64

user@SL-2> show interfaces irb.8 terse
Interface               Admin Link Proto    Local                 Remote
```

```
irb.8                   up    up   inet     10.0.8.244/24
                                            10.0.8.254/24
                                   inet6    2001:db8::10:0:8:244/112
                                            2001:db8::10:0:8:254/112
                                            fe80::8243:3f00:8e2:5d80/64

user@SL-2> show interfaces irb.2001 terse
Interface               Admin Link Proto    Local                 Remote
irb.2001                up    up   inet     10.20.1.244/24
                                            10.20.1.254/24
                                   inet6    2001:db8::10:0:7d1:244/112
                                            2001:db8::10:0:7d1:254/112
                                            fe80::8243:3f07:d1e2:5d80/64
```

SL-3 (output is similar on SL-6 for the revenue VLANs and IRB interfaces you configure on SL-6):

```
user@SL-3> show interfaces irb.1 terse
error: interface irb.1 not found

user@SL-3> show interfaces irb.2 terse
error: interface irb.2 not found

user@SL-3> show interfaces irb.3 terse
error: interface irb.3 not found

user@SL-3> show interfaces irb.4 terse
error: interface irb.4 not found

user@SL-3> show interfaces irb.5 terse
Interface               Admin Link Proto    Local                 Remote
irb.5                   up    up   inet     10.0.5.245/24
                                            10.0.5.254/24
                                   inet6    2001:db8::10:0:5:245/112
                                            2001:db8::10:0:5:254/112
                                            fe80::8a30:3700:576:f630/64

user@SL-3> show interfaces irb.6 terse
Interface               Admin Link Proto    Local                 Remote
irb.6                   up    up   inet     10.0.6.245/24
                                            10.0.6.254/24
                                   inet6    2001:db8::10:0:6:245/112
```

```
                                             2001:db8::10:0:6:254/112
                                             fe80::8a30:3700:676:f630/64

user@SL-3> show interfaces irb.7 terse
Interface               Admin Link Proto    Local                 Remote
irb.7                   up    up   inet     10.0.7.245/24
                                            10.0.7.254/24
                                   inet6    2001:db8::10:0:7:245/112
                                            2001:db8::10:0:7:254/112
                                            fe80::8a30:3700:776:f630/64

user@SL-3> show interfaces irb.8 terse
Interface               Admin Link Proto    Local                 Remote
irb.8                   up    up   inet     10.0.8.245/24
                                            10.0.8.254/24
                                   inet6    2001:db8::10:0:8:245/112
                                            2001:db8::10:0:8:254/112
                                            fe80::8a30:3700:876:f630/64

user@SL-3> show interfaces irb.2001 terse
Interface               Admin Link Proto    Local                 Remote
irb.2001                up    up   inet     10.20.1.245/24
                                            10.20.1.254/24
                                   inet6    2001:db8::10:0:7d1:245/112
                                            2001:db8::10:0:7d1:254/112
                                            fe80::8a30:3707:d176:f630/64
```

SL-5 (output is similar on SL-4):

```
user@SL-5> show interfaces irb.1 terse
Interface               Admin Link Proto    Local                 Remote
irb.1                   up    up   inet     10.0.1.247/24
                                            10.0.1.254          --> 0/0
                                   inet6    2001:db8::10:0:1:247/112
                                            2001:db8::10:0:1:254-->
                                            fe80::d699:6c00:17a:7ef7/64
                                   multiservice

user@SL-5> show interfaces irb.2 terse
Interface               Admin Link Proto    Local                 Remote
irb.2                   up    up   inet     10.0.2.247/24
```

```
                                     10.0.2.254          --> 0/0
                            inet6    2001:db8::10:0:2:247/112
                                     2001:db8::10:0:2:254-->
                                     fe80::d699:6c00:27a:7ef7/64
                            multiservice

user@SL-5> show interfaces irb.3 terse
Interface               Admin Link Proto    Local                 Remote
irb.3                   up    up   inet     10.0.3.247/24
                                     10.0.3.254          --> 0/0
                            inet6    2001:db8::10:0:3:247/112
                                     2001:db8::10:0:3:254-->
                                     fe80::d699:6c00:37a:7ef7/64
                            multiservice

user@SL-5> show interfaces irb.4 terse
Interface               Admin Link Proto    Local                 Remote
irb.4                   up    up   inet     10.0.4.247/24
                                     10.0.4.254          --> 0/0
                            inet6    2001:db8::10:0:4:247/112
                                     2001:db8::10:0:4:254-->
                                     fe80::d699:6c00:47a:7ef7/64
                            multiservice

user@SL-5> show interfaces irb.5 terse
error: device irb.5 not found

user@SL-5> show interfaces irb.6 terse
error: device irb.6 not found

user@SL-5> show interfaces irb.7 terse
error: device irb.7 not found

user@SL-5> show interfaces irb.8 terse
error: device irb.8 not found
```

4. Run the `show pim join extensive instance VRF-1` command on the leaf devices to see the joined multicast groups, source address, upstream neighbors, and downstream neighbors. These values show enhanced OISM in action. For more details on how enhanced OISM east-west traffic flow works, see *How Enhanced OISM Works*.

For example, we show output from this command for this use case on SL-2 as a source leaf device, and SL-3 and SL-5 as receiving leaf devices. Based on the test environment for this use case (see Table 21 on page 591), note the following in the output:

- **Group**—We use multicast groups 233.252.0.1 through 233.252.0.3 for intra-VLAN traffic (L2 forwarding), and 233.252.0.101 through 233.252.0.103 for inter-VLAN (L3 routing) traffic.
- **Source**—The IGMPv3 SSM multicast source is behind multihoming peer leaf devices SL-1 and SL-2, with source address 10.0.1.12.
- **Upstream Interface**, **Downstream neighbors**—The source VLAN is VLAN-1 with corresponding IRB interface irb.1 (the upstream interface). The output shows that with enhanced OISM, either SL-1 or SL-2 send the source traffic toward the receivers from the source VLAN out onto the SBD (irb.2001 is the downstream neighbor) instead of on the source VLAN the way regular OISM operates.

  Also, with enhanced OISM, the leaf devices that host receivers, such as SL-3, SL-4, and SL-5 shown below, receive the traffic on the SBD (irb.2001 is the upstream neighbor), whether the traffic is intra-VLAN or inter-VLAN. Then those leaf devices route the traffic from the SBD to the destination VLAN using the corresponding IRB interfaces. The output is similar for the receivers behind the other leaf devices based on the VLANs they host.

**SL-2 in the source role (output is similar on SL-1):**

```
user@SL-2> show pim join extensive instance VRF-1
Instance: PIM.VRF-1 Family: INET
R = Rendezvous Point Tree, S = Sparse, W = Wildcard

Group: 233.252.0.1
    Source: 10.0.1.12
    Flags: sparse,spt
    Upstream interface: irb.1
    Upstream neighbor: Direct
    Upstream state: Local Source, Local RP
    Keepalive timeout: 311
    Uptime: 00:44:56
    Downstream neighbors:
        Interface: irb.2001
            10.20.1.244 State: Join Flags: S   Timeout: Infinity
            Uptime: 00:44:56 Time since last Join: 00:44:56
    Number of downstream interfaces: 1
    Number of downstream neighbors: 1

Group: 233.252.0.2
```

```
    Source: 10.0.1.12
    Flags: sparse,spt
    Upstream interface: irb.1
    Upstream neighbor: Direct
    Upstream state: Local Source, Local RP
    Keepalive timeout: 311
    Uptime: 00:44:56
    Downstream neighbors:
        Interface: irb.2001
            10.20.1.244 State: Join Flags: S   Timeout: Infinity
            Uptime: 00:44:56 Time since last Join: 00:44:56
    Number of downstream interfaces: 1
    Number of downstream neighbors: 1

...

Group: 233.252.0.101
    Source: 10.0.1.12
    Flags: sparse,spt
    Upstream interface: irb.1
    Upstream neighbor: Direct
    Upstream state: Local Source, Local RP
    Keepalive timeout: 311
    Uptime: 1d 00:06:01
    Downstream neighbors:
        Interface: irb.2001
            10.20.1.244 State: Join Flags: S   Timeout: Infinity
            Uptime: 1d 00:06:01 Time since last Join: 1d 00:06:01
    Number of downstream interfaces: 1
    Number of downstream neighbors: 1

Group: 233.252.0.102
    Source: 10.0.1.12
    Flags: sparse,spt
    Upstream interface: irb.1
    Upstream neighbor: Direct
    Upstream state: Local Source, Local RP
    Keepalive timeout: 311
    Uptime: 1d 00:06:01
    Downstream neighbors:
        Interface: irb.2001
            10.20.1.244 State: Join Flags: S   Timeout: Infinity
            Uptime: 1d 00:06:01 Time since last Join: 1d 00:06:01
    Number of downstream interfaces: 1
    Number of downstream neighbors: 1

...
```

SL-3 in the receiver role (SL-3 hosts VLANs 5 through 8):

```
user@SL-3> show pim join extensive instance VRF-1 source 10.0.1.12
Instance: PIM.VRF-1 Family: INET
R = Rendezvous Point Tree, S = Sparse, W = Wildcard

Group: 233.252.0.1
    Source: 10.0.1.12
    Flags: sparse,spt
    Upstream interface: irb.2001
    Upstream neighbor: 10.20.1.245
    Upstream state: Local Source, Local RP
    Keepalive timeout: 342
    Uptime: 01:57:14
    Downstream neighbors:
        Interface: irb.2001
            10.20.1.245 State: Join Flags: S   Timeout: Infinity
            Uptime: 01:57:14 Time since last Join: 01:57:14
    Number of downstream interfaces: 1
    Number of downstream neighbors: 1

Group: 233.252.0.2
    Source: 10.0.1.12
    Flags: sparse,spt
    Upstream interface: irb.2001
    Upstream neighbor: 10.20.1.245
    Upstream state: Local Source, Local RP
    Keepalive timeout: 342
    Uptime: 01:57:14
    Downstream neighbors:
        Interface: irb.2001
            10.20.1.245 State: Join Flags: S   Timeout: Infinity
            Uptime: 01:57:14 Time since last Join: 01:57:14
    Number of downstream interfaces: 1
    Number of downstream neighbors: 1
```

```
...

Group: 233.252.0.101
    Source: 10.0.1.12
    Flags: sparse,spt
    Upstream interface: irb.2001
    Upstream neighbor: 10.20.1.245
    Upstream state: Local Source, Local RP
    Keepalive timeout: 342
    Uptime: 5d 23:24:42
    Downstream neighbors:
        Interface: irb.5
            10.0.5.245 State: Join Flags: S   Timeout: Infinity
            Uptime: 5d 23:24:21 Time since last Join: 01:57:15
        Interface: irb.6
            10.0.6.245 State: Join Flags: S   Timeout: Infinity
            Uptime: 5d 23:24:30 Time since last Join: 01:57:15
        Interface: irb.7
            10.0.7.245 State: Join Flags: S   Timeout: Infinity
            Uptime: 5d 23:24:29 Time since last Join: 01:57:15
        Interface: irb.8
            10.0.8.245 State: Join Flags: S   Timeout: Infinity
            Uptime: 5d 23:24:27 Time since last Join: 01:57:15
        Interface: irb.2001
            10.20.1.245 State: Join Flags: S   Timeout: Infinity
            Uptime: 5d 23:24:42 Time since last Join: 01:57:14
    Number of downstream interfaces: 5
    Number of downstream neighbors: 5

Group: 233.252.0.102
    Source: 10.0.1.12
    Flags: sparse,spt
    Upstream interface: irb.2001
    Upstream neighbor: 10.20.1.245
    Upstream state: Local Source, Local RP
    Keepalive timeout: 342
    Uptime: 5d 23:24:42
    Downstream neighbors:
        Interface: irb.5
            10.0.5.245 State: Join Flags: S   Timeout: Infinity
            Uptime: 5d 23:24:21 Time since last Join: 01:57:15
        Interface: irb.6
            10.0.6.245 State: Join Flags: S   Timeout: Infinity
            Uptime: 5d 23:24:30 Time since last Join: 01:57:15
        Interface: irb.7
            10.0.7.245 State: Join Flags: S   Timeout: Infinity
            Uptime: 5d 23:24:29 Time since last Join: 01:57:15
        Interface: irb.8
            10.0.8.245 State: Join Flags: S   Timeout: Infinity
            Uptime: 5d 23:24:27 Time since last Join: 01:57:15
        Interface: irb.2001
            10.20.1.245 State: Join Flags: S   Timeout: Infinity
            Uptime: 5d 23:24:42 Time since last Join: 01:57:14
    Number of downstream interfaces: 5
    Number of downstream neighbors: 5

...
```

**SL-5 in the receiver role (SL-5 hosts VLANs 1 through 4; output is similar for multihoming peer device SL-4):**

```
user@SL-5> show pim join extensive instance VRF-1

Instance: PIM.VRF-1 Family: INET
R = Rendezvous Point Tree, S = Sparse, W = Wildcard

Group: 233.252.0.1
    Source: 10.0.1.12
    Flags: sparse,spt
    Upstream interface: irb.1
    Upstream neighbor: Direct
    Upstream state: Local Source, Local RP
    Keepalive timeout:
    Uptime: 5d 23:31:08
    Downstream neighbors:
        Interface: irb.1
            10.0.1.247 State: Join Flags: S   Timeout: Infinity
            Uptime: 5d 23:31:08 Time since last Join: 01:24:27
        Interface: irb.2001
            10.20.1.247 State: Join Flags: S   Timeout: Infinity
            Uptime: 5d 23:31:08 Time since last Join: 5d 23:28:34
    Number of downstream interfaces: 2
    Number of downstream neighbors: 2
```

```
Group: 233.252.0.2
    Source: 10.0.1.12
    Flags: sparse,spt
    Upstream interface: irb.1
    Upstream neighbor: Direct
    Upstream state: Local Source, Local RP
    Keepalive timeout:
    Uptime: 5d 23:31:08
    Downstream neighbors:
        Interface: irb.1
            10.0.1.247 State: Join Flags: S   Timeout: Infinity
            Uptime: 5d 23:31:08 Time since last Join: 01:24:27
        Interface: irb.2001
            10.20.1.247 State: Join Flags: S   Timeout: Infinity
            Uptime: 5d 23:31:08 Time since last Join: 5d 23:28:34
    Number of downstream interfaces: 2
    Number of downstream neighbors: 2

...

Group: 233.252.0.101
    Source: 10.0.1.12
    Flags: sparse,spt
    Upstream interface: irb.1
    Upstream neighbor: Direct
    Upstream state: Local Source, Local RP
    Keepalive timeout:
    Uptime: 5d 23:30:51
    Downstream neighbors:
        Interface: irb.2
            10.0.2.247 State: Join Flags: S   Timeout: Infinity
            Uptime: 5d 23:30:50 Time since last Join: 01:24:08
        Interface: irb.3
            10.0.3.247 State: Join Flags: S   Timeout: Infinity
            Uptime: 5d 23:30:49 Time since last Join: 01:24:08
        Interface: irb.4
            10.0.4.247 State: Join Flags: S   Timeout: Infinity
            Uptime: 5d 23:30:49 Time since last Join: 01:24:08
        Interface: irb.2001
            10.20.1.247 State: Join Flags: S   Timeout: Infinity
            Uptime: 5d 23:30:51 Time since last Join: 5d 23:28:15
    Number of downstream interfaces: 4
    Number of downstream neighbors: 4
```

```
Group: 233.252.0.102
    Source: 10.0.1.12
    Flags: sparse,spt
    Upstream interface: irb.1
    Upstream neighbor: Direct
    Upstream state: Local Source, Local RP
    Keepalive timeout:
    Uptime: 5d 23:30:51
    Downstream neighbors:
        Interface: irb.2
            10.0.2.247 State: Join Flags: S   Timeout: Infinity
            Uptime: 5d 23:30:50 Time since last Join: 01:24:08
        Interface: irb.3
            10.0.3.247 State: Join Flags: S   Timeout: Infinity
            Uptime: 5d 23:30:49 Time since last Join: 01:24:08
        Interface: irb.4
            10.0.4.247 State: Join Flags: S   Timeout: Infinity
            Uptime: 5d 23:30:49 Time since last Join: 01:24:08
        Interface: irb.2001
            10.20.1.247 State: Join Flags: S   Timeout: Infinity
            Uptime: 5d 23:30:51 Time since last Join: 5d 23:28:15
    Number of downstream interfaces: 4
    Number of downstream neighbors: 4

...
```

5. Run the `show multicast route source-prefix 10.0.1.12 instance VRF-1 extensive` command on the leaf devices to check for the expected routes in the multicast routing table.

   For example, we include the output for this use case on SL-2 as a source leaf device, and SL-3 and SL-5 as receiving leaf devices.

   Like in Step 4 of this verification, the output on SL-1 and SL-2 (source devices) should show that SL-1 and SL-2:

   - Receive the traffic on source VLAN 1 (irb.1) as the upstream interface.
   - Route the traffic out onto the SBD (irb.2001) toward the receivers for either intra-VLAN (L2) or inter-vlan (L3) multicast flows.

   The output on SL-3 and SL-5 should show that SL-3 and SL-5:

   - Receive the traffic on the SBD as the upstream interface.

- Route the traffic to the destination VLAN for the VLANs they host.

**SL-2 in the source role (output is similar on SL-1):**

```
user@SL-2> show multicast route source-prefix 10.0.1.12 instance VRF-1 extensive
Instance: VRF-1 Family: INET

Group: 233.252.0.1
    Source: 10.0.1.12/32
    Upstream interface: irb.1
    Downstream interface list:
        irb.2001
    Number of outgoing interfaces: 1
    Session description: Source specific multicast
    Statistics: 855 kBps, 1670 pps, 292063350 packets
    Next-hop ID: 524287
    Upstream protocol: PIM
    Route state: Active
    Forwarding state: Forwarding
    Cache lifetime/timeout: forever
    Wrong incoming interface notifications: 2
    Uptime: 2d 14:30:00

Group: 233.252.0.2
    Source: 10.0.1.12/32
    Upstream interface: irb.1
    Downstream interface list:
        irb.2001
    Number of outgoing interfaces: 1
    Session description: Source specific multicast
    Statistics: 855 kBps, 1670 pps, 292063350 packets
    Next-hop ID: 524287
    Upstream protocol: PIM
    Route state: Active
    Forwarding state: Forwarding
    Cache lifetime/timeout: forever
    Wrong incoming interface notifications: 2
    Uptime: 2d 14:30:00

...

Group: 233.252.0.101
```

```
    Source: 10.0.1.12/32
    Upstream interface: irb.1
    Downstream interface list:
        irb.2001
    Number of outgoing interfaces: 1
    Session description: Source specific multicast
    Statistics: 855 kBps, 1670 pps, 292851768 packets
    Next-hop ID: 524287
    Upstream protocol: PIM
    Route state: Active
    Forwarding state: Forwarding
    Cache lifetime/timeout: forever
    Wrong incoming interface notifications: 2
    Uptime: 2d 14:30:00

Group: 233.252.0.102
    Source: 10.0.1.12/32
    Upstream interface: irb.1
    Downstream interface list:
        irb.2001
    Number of outgoing interfaces: 1
    Session description: Source specific multicast
    Statistics: 570 kBps, 1114 pps, 263540784 packets
    Next-hop ID: 524287
    Upstream protocol: PIM
    Route state: Active
    Forwarding state: Forwarding
    Cache lifetime/timeout: forever
    Wrong incoming interface notifications: 2
    Uptime: 2d 14:30:00

...
```

SL-3 in the receiver role (SL-3 hosts VLANs 5 through 8):

```
user@SL-3> show multicast route source-prefix 10.0.1.12 instance VRF-1 extensive
Instance: VRF-1 Family: INET

Group: 233.252.0.1
    Source: 10.0.1.12/32
    Upstream interface: irb.2001 (rpf check disabled)
    Number of outgoing interfaces: 0
    Session description: Source specific multicast
    Statistics: 345 kBps, 614 pps, 208539837 packets
    Next-hop ID: 0
    Upstream protocol: PIM
    Route state: Active
    Forwarding state: Pruned
    Cache lifetime/timeout: forever
    Wrong incoming interface notifications: 0
    Uptime: 02:03:08

Group: 233.252.0.2
    Source: 10.0.1.12/32
    Upstream interface: irb.2001 (rpf check disabled)
    Number of outgoing interfaces: 0
    Session description: Source specific multicast
    Statistics: 345 kBps, 614 pps, 208539616 packets
    Next-hop ID: 0
    Upstream protocol: PIM
    Route state: Active
    Forwarding state: Pruned
    Cache lifetime/timeout: forever
    Wrong incoming interface notifications: 0
    Uptime: 02:03:08

...

Group: 233.252.0.101
    Source: 10.0.1.12/32
    Upstream interface: irb.2001 (rpf check disabled)
    Downstream interface list:
        irb.5 irb.6 irb.7 irb.8
    Number of outgoing interfaces: 4
    Session description: Source specific multicast
    Statistics: 345 kBps, 614 pps, 550061134 packets
    Next-hop ID: 524404
    Upstream protocol: PIM
    Route state: Active
    Forwarding state: Forwarding
    Cache lifetime/timeout: forever
    Wrong incoming interface notifications: 1
    Uptime: 5d 23:30:36
```

```
Group: 233.252.0.102
    Source: 10.0.1.12/32
    Upstream interface: irb.2001 (rpf check disabled)
    Downstream interface list:
        irb.5 irb.6 irb.7 irb.8
    Number of outgoing interfaces: 4
    Session description: Source specific multicast
    Statistics: 345 kBps, 614 pps, 550060356 packets
    Next-hop ID: 524404
    Upstream protocol: PIM
    Route state: Active
    Forwarding state: Forwarding
    Cache lifetime/timeout: forever
    Wrong incoming interface notifications: 1
    Uptime: 5d 23:30:36

...
```

SL-5 in the receiver role (SL-5 hosts VLANs 1 through 4; output is similar for multihoming peer device SL-4):

```
user@SL-5> show multicast route source-prefix 10.0.1.12 instance VRF-1 extensive
Instance: VRF-1 Family: INET

Group: 233.252.0.1
    Source: 10.0.1.12/32
    Upstream interface: irb.2001
    Downstream interface list:
        irb.1
    Number of outgoing interfaces: 1
    Session description: Source specific multicast
    Statistics: 374 kBps, 670 pps, 338600484 packets
    Next-hop ID: 33506
    Upstream protocol: Multicast
    Route state: Active
    Forwarding state: Forwarding
    Cache lifetime/timeout: forever
    Wrong incoming interface notifications: 1
    Uptime: 5d 23:32:10
    Sensor ID: 0xf00001ce
```

```
Group: 233.252.0.2
    Source: 10.0.1.12/32
    Upstream interface: irb.2001
    Downstream interface list:
        irb.1
    Number of outgoing interfaces: 1
    Session description: Source specific multicast
    Statistics: 374 kBps, 670 pps, 338600490 packets
    Next-hop ID: 33506
    Upstream protocol: Multicast
    Route state: Active
    Forwarding state: Forwarding
    Cache lifetime/timeout: forever
    Wrong incoming interface notifications: 1
    Uptime: 5d 23:34:15
    Sensor ID: 0xf000009f

...

Group: 233.252.0.101
    Source: 10.0.1.12/32
    Upstream interface: irb.2001
    Downstream interface list:
        irb.2 irb.3 irb.4
    Number of outgoing interfaces: 3
    Session description: Source specific multicast
    Statistics: 374 kBps, 670 pps, 3428920 packets
    Next-hop ID: 33697
    Upstream protocol: Multicast
    Route state: Active
    Forwarding state: Forwarding
    Cache lifetime/timeout: forever
    Wrong incoming interface notifications: 1
    Uptime: 5d 23:34:15
    Sensor ID: 0xf0000020

Group: 233.252.0.102
    Source: 10.0.1.12/32
    Upstream interface: irb.2001
    Downstream interface list:
        irb.2 irb.3 irb.4
    Number of outgoing interfaces: 3
    Session description: Source specific multicast
    Statistics: 374 kBps, 670 pps, 3428920 packets
    Next-hop ID: 33697
    Upstream protocol: Multicast
    Route state: Active
    Forwarding state: Forwarding
    Cache lifetime/timeout: forever
    Wrong incoming interface notifications: 1
    Uptime: 5d 23:34:15
    Sensor ID: 0xf0000021

...
```

6. Verify that multihoming peer devices receive source traffic on the source VLAN rather than on the SBD.

   In use case #1, SL-1 and SL-2 are multihoming peer devices with the multicast source behind them. On SL-2, we add a single-homed receiver connected on interface ae5 to TOR-7. See Figure 93 on page 582. In this case, if SL-1 is the ingress leaf device and receives the source traffic, then with enhanced OISM, SL-1 sends the traffic to SL-2 on the source VLAN instead of on the SBD. SL-1 sends the source traffic on the SBD to the other leaf devices with interested receivers.

   For more details on how enhanced OISM east-west traffic flow works, see *How Enhanced OISM Works*.

   **Run these commands on SL-2 with the single-homed receiver behind TOR-7:**

   a. Run the `show igmp snooping membership` command for the EVPN instance MACVRF-1 to verify that IGMP snooping is enabled on the VLANs that SL-2 hosts. You can also verify the multicast groups the receiver behind TOR-7 has joined. We show output for the first few VLANs. The output is similar for the remaining VLANs.

```
user@SL-2> show igmp snooping membership vlan VLAN-1 virtual-switch MACVRF-1
Instance: MACVRF-1

Vlan: VLAN-1

Learning-Domain: default
Interface: ae3.0, Groups: 0

Learning-Domain: default
Interface: ae5.0, Groups: 3
    Group: 233.252.0.1
        Group mode: Include
        Source: 10.0.1.12
        Source timeout: 143
        Last reported by: 10.0.1.98
        Group timeout:       0 Type: Dynamic
    Group: 233.252.0.2
        Group mode: Include
        Source: 10.0.1.12
        Source timeout: 143
        Last reported by: 10.0.1.98
        Group timeout:       0 Type: Dynamic
    Group: 233.252.0.3
        Group mode: Include
        Source: 10.0.1.12
        Source timeout: 143
        Last reported by: 10.0.1.98
        Group timeout:       0 Type: Dynamic

user@SL-2> show igmp snooping membership vlan VLAN-2 virtual-switch MACVRF-1
Instance: MACVRF-1

Vlan: VLAN-2

Learning-Domain: default
Interface: ae3.0, Groups: 0

Learning-Domain: default
Interface: ae5.0, Groups: 3
    Group: 233.252.0.101
        Group mode: Include
        Source: 10.0.1.12
        Source timeout: 144
        Last reported by: 10.0.2.98
        Group timeout:       0 Type: Dynamic
    Group: 233.252.0.102
        Group mode: Include
        Source: 10.0.1.12
        Source timeout: 144
        Last reported by: 10.0.2.98
        Group timeout:       0 Type: Dynamic
    Group: 233.252.0.103
        Group mode: Include
        Source: 10.0.1.12
        Source timeout: 144
        Last reported by: 10.0.2.98
        Group timeout:       0 Type: Dynamic

user@SL-2> show igmp snooping membership vlan VLAN-3 virtual-switch MACVRF-1
Instance: MACVRF-1

Vlan: VLAN-3

Learning-Domain: default
Interface: ae3.0, Groups: 0

Learning-Domain: default
Interface: ae5.0, Groups: 3
    Group: 233.252.0.101
        Group mode: Include
        Source: 10.0.1.12
        Source timeout: 150
        Last reported by: 10.0.3.98
        Group timeout:       0 Type: Dynamic
    Group: 233.252.0.102
        Group mode: Include
        Source: 10.0.1.12
        Source timeout: 150
        Last reported by: 10.0.3.98
        Group timeout:       0 Type: Dynamic
    Group: 233.252.0.103
        Group mode: Include
        Source: 10.0.1.12
        Source timeout: 150
        Last reported by: 10.0.3.98
        Group timeout:       0 Type: Dynamic

...
```

b. Run the `show pim join extensive instance VRF-1` command to verify the upstream interface is the IRB interface for the source VLAN, which is irb.1 in this case. The downstream neighbors include the SBD IRB interface and the other revenue VLAN IRB interfaces that have local receivers on the corresponding VLAN.

```
user@SL-2> show pim join extensive instance VRF-1
Instance: PIM.VRF-1 Family: INET
```

```
R = Rendezvous Point Tree, S = Sparse, W = Wildcard

Group: 233.252.0.1
    Source: 10.0.1.12
    Flags: sparse,spt
    Upstream interface: irb.1
    Upstream neighbor: Direct
    Upstream state: Local Source, Local RP
    Keepalive timeout:
    Uptime: 00:08:37
    Downstream neighbors:
        Interface: irb.1
            10.0.1.244 State: Join Flags: S   Timeout: Infinity
            Uptime: 00:08:37 Time since last Join: 00:08:37
        Interface: irb.2001
            10.20.1.244 State: Join Flags: S   Timeout: Infinity
            Uptime: 00:08:37 Time since last Join: 00:08:37
    Number of downstream interfaces: 2
    Number of downstream neighbors: 2

Group: 233.252.0.2
    Source: 10.0.1.12
    Flags: sparse,spt
    Upstream interface: irb.1
    Upstream neighbor: Direct
    Upstream state: Local Source, Local RP
    Keepalive timeout:
    Uptime: 00:08:37
    Downstream neighbors:
        Interface: irb.1
            10.0.1.244 State: Join Flags: S   Timeout: Infinity
            Uptime: 00:08:37 Time since last Join: 00:08:37
        Interface: irb.2001
            10.20.1.244 State: Join Flags: S   Timeout: Infinity
            Uptime: 00:08:37 Time since last Join: 00:08:37
    Number of downstream interfaces: 2
    Number of downstream neighbors: 2

...

Group: 233.252.0.101
    Source: 10.0.1.12
    Flags: sparse,spt
    Upstream interface: irb.1
    Upstream neighbor: Direct
    Upstream state: Local Source, Local RP
    Keepalive timeout: 0
    Uptime: 00:14:32
    Downstream neighbors:
        Interface: irb.2
            10.0.2.244 State: Join Flags: S   Timeout: Infinity
            Uptime: 00:08:37 Time since last Join: 00:08:37
        Interface: irb.3
            10.0.3.244 State: Join Flags: S   Timeout: Infinity
            Uptime: 00:08:37 Time since last Join: 00:08:37
        Interface: irb.4
            10.0.4.244 State: Join Flags: S   Timeout: Infinity
            Uptime: 00:08:37 Time since last Join: 00:08:37
        Interface: irb.5
            10.0.5.244 State: Join Flags: S   Timeout: Infinity
            Uptime: 00:08:37 Time since last Join: 00:08:37
        Interface: irb.6
            10.0.6.244 State: Join Flags: S   Timeout: Infinity
            Uptime: 00:08:37 Time since last Join: 00:08:37
        Interface: irb.7
            10.0.7.244 State: Join Flags: S   Timeout: Infinity
            Uptime: 00:08:37 Time since last Join: 00:08:37
        Interface: irb.8
            10.0.8.244 State: Join Flags: S   Timeout: Infinity
            Uptime: 00:08:37 Time since last Join: 00:08:37
        Interface: irb.2001
            10.20.1.244 State: Join Flags: S   Timeout: Infinity
            Uptime: 00:14:32 Time since last Join: 00:14:31
    Number of downstream interfaces: 8
    Number of downstream neighbors: 8

Group: 233.252.0.102
    Source: 10.0.1.12
    Flags: sparse,spt
    Upstream interface: irb.1
    Upstream neighbor: Direct
    Upstream state: Local Source, Local RP
    Keepalive timeout: 0
    Uptime: 00:14:32
    Downstream neighbors:
        Interface: irb.2
```

```
            10.0.2.244 State: Join Flags: S   Timeout: Infinity
            Uptime: 00:08:37 Time since last Join: 00:08:37
        Interface: irb.3
            10.0.3.244 State: Join Flags: S   Timeout: Infinity
            Uptime: 00:08:37 Time since last Join: 00:08:37
        Interface: irb.4
            10.0.4.244 State: Join Flags: S   Timeout: Infinity
            Uptime: 00:08:37 Time since last Join: 00:08:37
        Interface: irb.5
            10.0.5.244 State: Join Flags: S   Timeout: Infinity
            Uptime: 00:08:37 Time since last Join: 00:08:37
        Interface: irb.6
            10.0.6.244 State: Join Flags: S   Timeout: Infinity
            Uptime: 00:08:37 Time since last Join: 00:08:37
        Interface: irb.7
            10.0.7.244 State: Join Flags: S   Timeout: Infinity
            Uptime: 00:08:37 Time since last Join: 00:08:37
        Interface: irb.8
            10.0.8.244 State: Join Flags: S   Timeout: Infinity
            Uptime: 00:08:37 Time since last Join: 00:08:37
        Interface: irb.2001
            10.20.1.244 State: Join Flags: S   Timeout: Infinity
            Uptime: 00:14:32 Time since last Join: 00:14:31
    Number of downstream interfaces: 8
    Number of downstream neighbors: 8

...
```

c. Run the `show multicast route source-prefix 10.0.1.12 instance VRF-1` command to check for the expected routes in the multicast forwarding table. The the source VLAN IRB interface, irb.1, is the upstream interface. The outgoing interface list (the downstream interface list) consists of the other IRB interfaces to which the device forwards and routes the multicast traffic.

```
user@SL-2> show multicast route source-prefix 10.0.1.12 instance VRF-1
Instance: VRF-1 Family: INET

Group: 233.252.0.1
    Source: 10.0.1.12/32
    Upstream interface: irb.1
    Downstream interface list:
        irb.2001
```

```
Group: 233.252.0.2
    Source: 10.0.1.12/32
    Upstream interface: irb.1
    Downstream interface list:
        irb.2001

...

Group: 233.252.0.101
    Source: 10.0.1.12/32
    Upstream interface: irb.1
    Downstream interface list:
        irb.2 irb.3 irb.4 irb.5 irb.6 irb.7 irb.8 irb.2001

Group: 233.252.0.102
    Source: 10.0.1.12/32
    Upstream interface: irb.1
    Downstream interface list:
        irb.2 irb.3 irb.4 irb.5 irb.6 irb.7 irb.8 irb.2001

...
```

d. Run the `show multicast snooping route source-prefix 10.0.1.12 instance MACVRF-1 extensive` command to check for the expected entries in the multicast snooping forwarding table with the outgoing interface for the multicast traffic. (For brevity, in the output below we have trimmed similar results for some of the multicast groups.)

```
user@SL-2> show multicast snooping route source-prefix 10.0.1.12 instance MACVRF-1
extensive
Nexthop Bulking: OFF

             Family: INET

Group: 233.252.0.1/32
    Source: 10.0.1.12/32
    Vlan: VLAN-1
    Mesh-group: __all_ces__
        Downstream interface list:
        ae5.0 -(2411)
    Statistics: 0 kBps, 0 pps, 0 packets
    Next-hop ID: 524481
    Route state: Active
    Forwarding state: Forwarding

Group: 233.252.0.2/32
    Source: 10.0.1.12/32
    Vlan: VLAN-1
    Mesh-group: __all_ces__
        Downstream interface list:
        ae5.0 -(2411)
    Statistics: 0 kBps, 0 pps, 0 packets
    Next-hop ID: 524481
    Route state: Active
    Forwarding state: Forwarding

...

Group: 233.252.0.101/32
    Source: 10.0.1.12/32
    Vlan: VLAN-1
    Mesh-group: __all_ces__
    Statistics: 0 kBps, 0 pps, 0 packets
    Next-hop ID: 0
    Route state: Active
    Forwarding state: Pruned

...

Group: 233.252.0.101/32
    Source: 10.0.1.12/32
    Vlan: VLAN-2
    Mesh-group: __all_ces__
        Downstream interface list:
        ae5.0 -(2411)
    Statistics: 0 kBps, 0 pps, 0 packets
    Next-hop ID: 524481
    Route state: Active
    Forwarding state: Forwarding

...

Group: 233.252.0.1/32
    Source: 10.0.1.12/32
    Vlan: VLAN-2001
    Mesh-group: __all_ces__
        Downstream interface list:
        evpn-core-nh -(524454)
    Statistics: 0 kBps, 0 pps, 0 packets
    Next-hop ID: 524475
    Route state: Active
    Forwarding state: Forwarding

...

Group: 233.252.0.101/32
    Source: 10.0.1.12/32
    Vlan: VLAN-2001
    Mesh-group: __all_ces__
        Downstream interface list:
        evpn-core-nh -(524451)
    Statistics: 0 kBps, 0 pps, 0 packets
    Next-hop ID: 524371
    Route state: Active
    Forwarding state: Forwarding

...

Group: 233.252.0.101/32
    Source: 10.0.1.12/32
    Vlan: VLAN-3
    Mesh-group: __all_ces__
        Downstream interface list:
        ae5.0 -(2411)
    Statistics: 0 kBps, 0 pps, 0 packets
    Next-hop ID: 524481
    Route state: Active
    Forwarding state: Forwarding

...

Group: 233.252.0.101/32
    Source: 10.0.1.12/32
    Vlan: VLAN-4
    Mesh-group: __all_ces__
        Downstream interface list:
        ae5.0 -(2411)
    Statistics: 0 kBps, 0 pps, 0 packets
    Next-hop ID: 524481
    Route state: Active
    Forwarding state: Forwarding

...

Group: 233.252.0.101/32
    Source: 10.0.1.12/32
    Vlan: VLAN-5
    Mesh-group: __all_ces__
        Downstream interface list:
        ae5.0 -(2411)
    Statistics: 0 kBps, 0 pps, 0 packets
    Next-hop ID: 524481
    Route state: Active
    Forwarding state: Forwarding
...

Group: 233.252.0.101/32
    Source: 10.0.1.12/32
    Vlan: VLAN-6
    Mesh-group: __all_ces__
        Downstream interface list:
        ae5.0 -(2411)
    Statistics: 0 kBps, 0 pps, 0 packets
    Next-hop ID: 524481
    Route state: Active
    Forwarding state: Forwarding

...

Group: 233.252.0.101/32
    Source: 10.0.1.12/32
    Vlan: VLAN-7
    Mesh-group: __all_ces__
        Downstream interface list:
        ae5.0 -(2411)
    Statistics: 0 kBps, 0 pps, 0 packets
    Next-hop ID: 524481
    Route state: Active
    Forwarding state: Forwarding
```

```
...

Group: 233.252.0.101/32
    Source: 10.0.1.12/32
    Vlan: VLAN-8
    Mesh-group: __all_ces__
        Downstream interface list:
        ae5.0 -(2411)
    Statistics: 0 kBps, 0 pps, 0 packets
    Next-hop ID: 524481
    Route state: Active
    Forwarding state: Forwarding

...
```

## Use Case #2: Internal Source to Internal and External Receivers with IGMPv2—ASM

In use case #2, we configure the topology in Figure 95 on page 639 with a tenant VRF called VRF-26. This use case includes:

- Intra-VLAN and inter-VLAN multicast flows using IGMPv2.
- A single-homed internal multicast source and internal multicast receivers.
- An external multicast receiver in an external PIM domain. As a result:
  - The ingress leaf device advertises EVPN Type 10 S-PMSI A-D routes for each internal multicast source and group (S,G).
  - The OISM border leaf PEG devices, BL-1 and BL-2, receive the EVPN Type 10 route. The BL device that is the PIM designated router (DR) for the SBD, BL-2 in this case, sends a PIM register message for the (S,G) on its PEG interface to the PIM router in the external PIM domain.
  - In response to the PIM register message, the PIM router sends a PIM Join message to either of the multihomed attached OISM PEG devices, BL-1 or BL-2. In this case, BL-2 receives the PIM Join message. .
  - The OISM PEG device that receives the PIM join, BL-2, sends the multicast traffic on its PEG interface toward the external receiver behind the PIM router.

**NOTE**: The test environment uses an MX Series router as the PIM router and PIM RP in an external PIM domain. See "Configure External Multicast PIM Router and PIM RP Router" on page 538 for an example of how to configure an MX Series router as a PIM router and RP with regular OISM; the steps are similar with enhanced OISM.

**Figure 95: Enhanced OISM Use Case #2 Topology—IGMPv2 with Source Behind SL-3 and External Receiver in External PIM Domain**

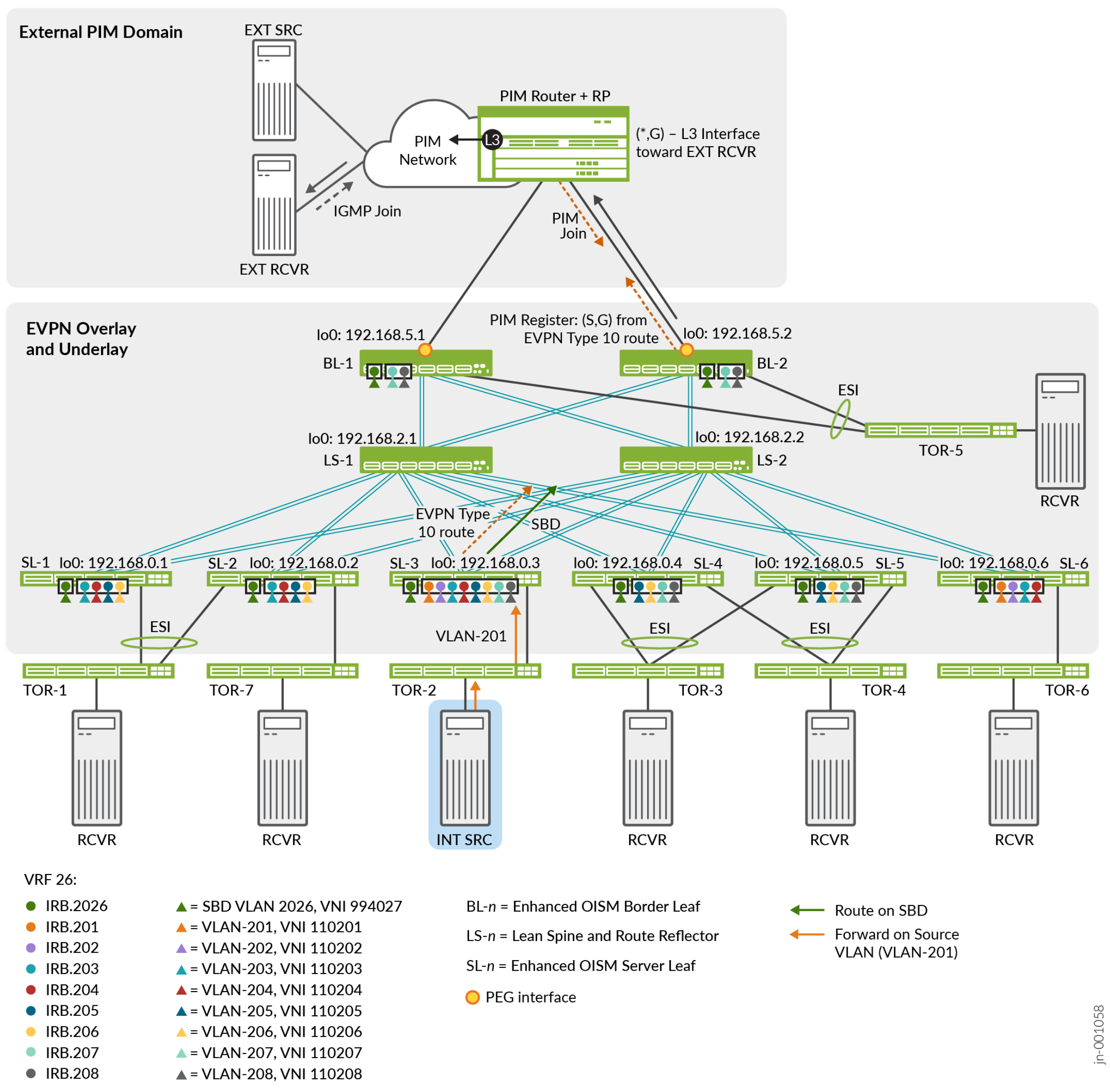


Table 23 on page 640 describes the multicast groups, device roles, configured VLANs, VXLAN VNI mappings for the VLANs, and the corresponding IRB interfaces for each VLAN.

**Table 23: Use Case#2 Elements for Internal Source to Internal and External Receivers with IGMPv2**

| Role | Device | Configured Revenue VLANs | Configured IRB Interfaces | VXLAN VNI Mappings |
|---|---|---|---|---|
| SBD for VRF-26 on all enhanced OISM leaf devices: VLAN-2026, irb.2026, VNI 994027 | | | | |
| Multicast source VLAN: VLAN-201, Source Host IP address: 10.0.201.12 | | | | |
| IGMPv2—ASM multicast groups: 233.252.0.71 – 233.252.0.73 for intra-VLAN (L2), and 233.252.0.171 – 233.252.0.173 for inter-VLAN (L3) | | | | |
| Source | TOR-2—Single-homed to SL-3 | VLAN-201 - VLAN-208 | irb.201 - irb.208 | VNI 110201 - VNI 110208 |
| Receivers | TOR-1—Multihomed to SL-1[1] and SL-2[1] | VLAN-203- VLAN-206 | irb.203- irb.206 | VNI 110203- VNI 110206 |
| | TOR-3—Multihomed to SL-4[2] and SL-5[2] | VLAN-205 - VLAN-208 | irb.205 - irb.208 | VNI 110205 - VNI 110208 |
| | TOR-4—Multihomed to SL-4[2] and SL-5[2] | VLAN-205 - VLAN-208 | irb.205 - irb.208 | VNI 110205 - VNI 110208 |
| | TOR-5—Multihomed to BL-1[3] and BL-2[3] | VLAN-207 - VLAN-208 | irb.207 - irb.208 | VNI 110207 - VNI 110208 |
| | TOR-6—Single-homed to SL-6 | VLAN-201 - VLAN-204 | irb.201 - irb.204 | VNI 110201 - VNI 110204 |
| | EXT RCVR in External PIM domain | VLAN-3126 | n/a | n/a |

**Table 23: Use Case#2 Elements for Internal Source to Internal and External Receivers with IGMPv2 *(Continued)***

| Role | Device | Configured Revenue VLANs | Configured IRB Interfaces | VXLAN VNI Mappings |
|---|---|---|---|---|

[1] SL-1 and SL-2 are multihoming peers, so configure the same revenue VLANs on SL-1 and SL-2.

[2] SL-4 and SL-5 are multihoming peers, so configure the same revenue VLANs on SL-4 and SL-5.

[3] BL-1 and BL-2 are multihoming peers, so configure the same revenue VLANs on BL-1 and BL-2.

## Configure Use Case #2: Internal Source to Internal and External Receivers with IGMPv2—ASM

Configure the revenue VLANs, SBD, tenant VRF, and multicast protocols specific to the use case in "Use Case #2: Internal Source to Internal and External Receivers with IGMPv2—ASM" on page 638.

The main differences in this use case from use case #1 are that here:

- We use enhanced OISM for multicast flows with IGMPv2 (ASM) from a single-homed source behind SL-3.
- SL-3 hosts revenue VLANs VLAN-201 through VLAN-208, and the other leaf devices host different subsets of those revenue VLANs.
- We use VRF-26, and configure the SBD for this VRF as VLAN-2026.

1. On each OISM leaf device, configure the revenue VLANs the device hosts, their corresponding IRB interfaces, and VNI mappings in the same way as "Configure Use Case #1: Internal Source and Receivers with Multihoming Peer Receiver and IGMPv3—SSM" on page 592 for steps 1 through 3, but use the values in Table 23 on page 640, and Table 24 on page 642 below for the IRB interface addresses and virtual gateway addresses.

**Table 24: Use Case #2 IRB Addresses and Virtual Gateway Addresses**

| Leaf Device | IRB Interface Unit# | IRB IPv4 Address<br><br>IPv4 VGA | IRB IPv6 Address<br><br>IPv6 VGA |
|---|---|---|---|
| SL-1 | Revenue VLANs—201 through 208 | 10.0.*unit#*.243/24<br><br>10.0.*unit#*.254 | 2001:db8::10:0:*hex-unit#*:243/112<br><br>2001:db8::10:0:*hex-unit#*:254 |
| | SBD—2026 | 10.20.26.243/24<br><br>10.20.26.254 | 2001:db8::10:0:7ea:243/112<br><br>2001:db8::10:0:7ea:254 |
| SL-2 | Revenue VLANs—201 through 208 | 10.0.*unit#*.244/24<br><br>10.0.*unit#*.254 | 2001:db8::10:0:*hex-unit#*:244/112<br><br>2001:db8::10:0:*hex-unit#*:254 |
| | SBD—2026 | 10.20.26.244/24<br><br>10.20.26.254 | 2001:db8::10:0:7ea:244/112<br><br>2001:db8::10:0:7ea:254 |
| SL-3 | Revenue VLANs—205 through 208 | 10.0.*unit#*.245/24<br><br>10.0.*unit#*.254 | 2001:db8::10:0:*hex-unit#*:245/112<br><br>2001:db8::10:0:*hex-unit#*:254 |
| | SBD—2026 | 10.20.26.245/24<br><br>10.20.26.254 | 2001:db8::10:0:7ea:245/112<br><br>2001:db8::10:0:7ea:254 |
| SL-4 | Revenue VLANs—201 through 204 | 10.0.*unit#*.246/24<br><br>10.0.*unit#*.254 | 2001:db8::10:0:*hex-unit#*:246/112<br><br>2001:db8::10:0:*hex-unit#*:254 |
| | SBD—2026 | 10.20.26.246/24<br><br>10.20.26.254 | 2001:db8::10:0:7ea:246/112<br><br>2001:db8::10:0:7ea:254 |

**Table 24: Use Case #2 IRB Addresses and Virtual Gateway Addresses *(Continued)***

| Leaf Device | IRB Interface Unit# | IRB IPv4 Address<br><br>IPv4 VGA | IRB IPv6 Address<br><br>IPv6 VGA |
|---|---|---|---|
| SL-5 | Revenue VLANs—201 through 204 | 10.0.*unit#*.247/24<br><br>10.0.*unit#*.254 | 2001:db8::10:0:*hex-unit#*:247/112<br><br>2001:db8::10:0:*hex-unit#*:254 |
| | SBD—2026 | 10.20.26.247/24<br><br>10.20.26.254 | 2001:db8::10:0:7ea:247/112<br><br>2001:db8::10:0:7ea:254 |
| SL-6 | Revenue VLANs—203 through 206 | 10.0.*unit#*.248/24<br><br>10.0.*unit#*.254 | 2001:db8::10:0:*hex-unit#*:248/112<br><br>2001:db8::10:0:*hex-unit#*:254 |
| | SBD—2026 | 10.20.26.248/24<br><br>10.20.26.254 | 2001:db8::10:0:7ea:248/112<br><br>2001:db8::10:0:7ea:254 |
| BL-1 | Revenue VLANs—207 and 208 | 10.0.*unit#*.241/24<br><br>10.0.*unit#*.254 | 2001:db8::10:0:*hex-unit#*:241/112<br><br>2001:db8::10:0:*hex-unit#*:254 |
| | SBD—2026 | 10.20.26.241/24<br><br>10.20.26.254 | 2001:db8::10:0:7ea:241/112<br><br>2001:db8::10:0:7ea:254 |
| BL-2 | Revenue VLANs—207 and 208 | 10.0.*unit#*.242/24<br><br>10.0.*unit#*.254 | 2001:db8::10:0:*hex-unit#*:242/112<br><br>2001:db8::10:0:*hex-unit#*:254 |
| | SBD—2026 | 10.20.26.242/24<br><br>10.20.26.254 | 2001:db8::10:0:7ea:242/112<br><br>2001:db8::10:0:7ea:254 |

2. This use case tests IGMPv2 ASM flows. IGMPv2 is enabled by default on all interfaces on which you configure PIM, so you don't need to explicitly enable IGMP in this case. Each OISM leaf device will use IGMPv2 on the IRB interfaces for which you did not explicitly configure IGMPv3 with the `[edit protocols igmp interface` *`interface-name`* `version 3` statement.
3. On each OISM leaf device, enable IGMP snooping for IGMPv2 in the MACVRF-1 instance for the revenue VLANs the device hosts, and the SBD.

   For simplicity, you can enable IGMP snooping for all VLANs in the MAC-VRF instance with the following command:

```
set routing-instances MACVRF-1 protocols igmp-snooping vlan all proxy
```

   **NOTE**: With the `vlan all` option above, if you need IGMP snooping with IGMPv3 for some of the VLANs, you can explicitly enable IGMP snooping with the `evpn-ssm-reports-only` option only for those VLANs, as we do in use case #1. The remaining VLANs will use IGMP snooping with IGMPv2.

   Alternatively, on each device you can enable IGMP snooping explicitly for the SBD and the VLANs that device hosts.

   **All OISM leaf devices:**

```
set routing-instances MACVRF-1 protocols igmp-snooping vlan VLAN-2026
```

   **SL-1 and SL-2—VLANs 203 through 206:**

```
set routing-instances MACVRF-1 protocols igmp-snooping vlan VLAN-203
set routing-instances MACVRF-1 protocols igmp-snooping vlan VLAN-204
set routing-instances MACVRF-1 protocols igmp-snooping vlan VLAN-205
set routing-instances MACVRF-1 protocols igmp-snooping vlan VLAN-206
```

   **SL-3—VLANs 201 through 208:**

```
set routing-instances MACVRF-1 protocols igmp-snooping vlan VLAN-201
set routing-instances MACVRF-1 protocols igmp-snooping vlan VLAN-202
set routing-instances MACVRF-1 protocols igmp-snooping vlan VLAN-203
set routing-instances MACVRF-1 protocols igmp-snooping vlan VLAN-204
set routing-instances MACVRF-1 protocols igmp-snooping vlan VLAN-205
set routing-instances MACVRF-1 protocols igmp-snooping vlan VLAN-206
set routing-instances MACVRF-1 protocols igmp-snooping vlan VLAN-207
set routing-instances MACVRF-1 protocols igmp-snooping vlan VLAN-208
```

SL-4 and SL-5—VLANs 205 through 208:

```
set routing-instances MACVRF-1 protocols igmp-snooping vlan VLAN-205
set routing-instances MACVRF-1 protocols igmp-snooping vlan VLAN-206
set routing-instances MACVRF-1 protocols igmp-snooping vlan VLAN-207
set routing-instances MACVRF-1 protocols igmp-snooping vlan VLAN-208
```

SL-6—VLANs 201 through 204:

```
set routing-instances MACVRF-1 protocols igmp-snooping vlan VLAN-201
set routing-instances MACVRF-1 protocols igmp-snooping vlan VLAN-202
set routing-instances MACVRF-1 protocols igmp-snooping vlan VLAN-203
set routing-instances MACVRF-1 protocols igmp-snooping vlan VLAN-204
```

BL-1 and BL-2—VLANs 207 and 208:

```
set routing-instances MACVRF-1 protocols igmp-snooping vlan VLAN-207
set routing-instances MACVRF-1 protocols igmp-snooping vlan VLAN-208
```

4. On each OISM leaf device, configure a loopback logical interface for the VRF instance. We include this interface in the VRF instance, VRF-26, in the next step.

   For the loopback logical interface configuration, we use the following conventions:

   - The logical unit number matches the VRF instance number.
   - The last octet of the interface's IP address also matches the VRF instance number.

   SL-1:

```
set interfaces lo0 unit 26 family inet address 10.3.1.26/32 primary
```

   SL-2:

```
set interfaces lo0 unit 26 family inet address 10.3.2.26/32 primary
```

SL-3:

```
set interfaces lo0 unit 26 family inet address 10.4.3.26/32 primary
```

SL-4:

```
set interfaces lo0 unit 26 family inet address 10.7.4.26/32 primary
```

SL-5:

```
set interfaces lo0 unit 26 family inet address 10.7.5.26/32 primary
```

SL-6:

```
set interfaces lo0 unit 26 family inet address 10.8.6.26/32 primary
```

BL-1:

```
set interfaces lo0 unit 26 family inet address 10.1.1.26/32 primary
```

BL-2:

```
set interfaces lo0 unit 26 family inet address 10.1.2.26/32 primary
```

5. Configure the tenant VRF instance named VRF-26 on each of the OISM leaf devices.

   Include the same elements for the server leaf and border leaf devices in the VRF configuration as we describe in use case #1, Step 7, with the following differences for this use case:

   - Identify the OISM SBD IRB interface associated with this VRF instance, which is `irb.26`.

   - On the border leaf devices, the L3 PEG interface for VRF-26 is ae3.25 (logical unit = *VRF instance number - 1*).

- On each device, include the loopback logical interface for this VRF instance that you configure in Step 4 above.
- Use the same route target for this VRF instance on all of the OISM leaf devices—`target:100:26`.
- Configure an RD following the same convention as in use case #1, in which the first part of the RD value mirrors the device loopback (unit 0) IP address, and the second part matches the VRF instance number—*`device-loopback-unit-0-address`*`:26`.

All server leaf devices SL-1 through SL-6:

```
set routing-instances VRF-26 instance-type vrf
set routing-instances VRF-26 routing-options graceful-restart
set routing-instances VRF-26 protocols evpn oism supplemental-bridge-domain-irb irb.2026
set routing-instances VRF-26 protocols ospf area 1.1.1.1 interface irb.2026 priority 0
set routing-instances VRF-26 protocols ospf area 1.1.1.1 interface all passive
set routing-instances VRF-26 protocols pim passive
set routing-instances VRF-26 protocols pim interface all
set routing-instances VRF-26 protocols pim interface irb.2026 accept-remote-source
set routing-instances VRF-26 protocols pim disable-packet-register
set routing-instances VRF-26 interface lo0.26
set routing-instances VRF-26 interface irb.2026
set routing-instances VRF-26 route-distinguisher device-loopback-unit-0-address:26
set routing-instances VRF-26 vrf-target target:100:26
```

Border leaf devices BL-1 and BL-2:

```
set policy-options policy-statement export-direct term t1 from protocol direct
set policy-options policy-statement export-direct term t1 then accept
set routing-instances VRF-26 instance-type vrf
set routing-instances VRF-26 routing-options graceful-restart
set routing-instances VRF-26 protocols evpn oism supplemental-bridge-domain-irb irb.2026
set routing-instances VRF-26 protocols ospf area 1.1.1.1 interface ae3.25 mtu 9178
set routing-instances VRF-26 protocols ospf area 1.1.1.1 interface irb.2026
set routing-instances VRF-26 protocols ospf area 1.1.1.1 interface all passive
set routing-instances VRF-26 protocols ospf export export-direct
set routing-instances VRF-26 protocols pim rp static address pim-rp-IP-address
set routing-instances VRF-26 protocols pim interface irb.207 distributed-dr
set routing-instances VRF-26 protocols pim interface irb.208 distributed-dr
set routing-instances VRF-26 protocols pim interface lo0.26
set routing-instances VRF-26 protocols pim interface ae3.25
set routing-instances VRF-26 protocols pim interface irb.2026 family inet bfd-liveness-
```

```
detection minimum-interval 1000
set routing-instances VRF-26 protocols pim interface irb.2026 family inet bfd-liveness-
detection multiplier 3
set routing-instances VRF-26 protocols pim interface irb.2026 stickydr
set routing-instances VRF-26 protocols pim interface irb.2026 hello-interval 30
set routing-instances VRF-26 protocols pim interface irb.2026 accept-remote-source
set routing-instances VRF-26 protocols pim disable-packet-register
set routing-instances VRF-26 interface lo0.26
set routing-instances VRF-26 interface ae3.25
set routing-instances VRF-26 interface irb.2026
set routing-instances VRF-26 route-distinguisher device-loopback-unit-0-address:26
set routing-instances VRF-26 vrf-target target:100:26
```

Then in the VRF instance configuration on each of the leaf devices, add the IRB interfaces only for the revenue VLANs the device hosts:

**SL-1 and SL-2:**

```
set routing-instances VRF-26 interface irb.203
set routing-instances VRF-26 interface irb.204
set routing-instances VRF-26 interface irb.205
set routing-instances VRF-26 interface irb.206
```

**SL-3:**

```
set routing-instances VRF-26 interface irb.201
set routing-instances VRF-26 interface irb.202
set routing-instances VRF-26 interface irb.203
set routing-instances VRF-26 interface irb.204
set routing-instances VRF-26 interface irb.205
set routing-instances VRF-26 interface irb.206
set routing-instances VRF-26 interface irb.207
set routing-instances VRF-26 interface irb.208
```

**SL-4 and SL-5:**

```
set routing-instances VRF-26 interface irb.205
set routing-instances VRF-26 interface irb.206
set routing-instances VRF-26 interface irb.207
set routing-instances VRF-26 interface irb.208
```

SL-6:

```
set routing-instances VRF-26 interface irb.201
set routing-instances VRF-26 interface irb.202
set routing-instances VRF-26 interface irb.203
set routing-instances VRF-26 interface irb.204
```

BL-1 and BL-2:

```
set routing-instances VRF-26 interface irb.207
set routing-instances VRF-26 interface irb.208
```

## Verify Use Case #2: Internal Source to Internal and External Receivers with IGMPv2—ASM

Verify enhanced OISM operation for use case #2 where we have a single-homed internal source behind SL-3 sending IGMPv2 multicast traffic. SL-2 is a multihoming peer of SL-1, and also connects to an internal receiver on TOR-7. SL-3 through SL-6, BL-1, and BL-2 connect to internal receivers. An external receiver also subscribes to the source traffic through an external PIM router and RP by way of OISM PEG devices BL-1 and BL-2. See the topology in Figure 95 on page 639, and the configuration in "Configure Use Case #2: Internal Source to Internal and External Receivers with IGMPv2—ASM" on page 641.

In this case, we focus on how to verify enhanced OISM operation with an external receiver. SL-6, as the ingress OISM leaf device, advertises EVPN Type 10 S-PMSI A-D routes for the multicast source and groups (S,G). BL-1 and BL-2 receive the EVPN Type 10 route on the SBD. In this example, BL-2 is the PIM DR for the SBD, so BL-2 sends a PIM register message for the (S,G) on its PEG interface to the PIM router and RP. The PIM router sends a PIM Join back to either of the multihomed attached OISM PEG devices, in this case BL-2. BL-2 sends multicast traffic from that (S,G) on its PEG interface toward the external receiver.

See "Use Case #2: Internal Source to Internal and External Receivers with IGMPv2—ASM" on page 638 for all of the parameters in this use case.

1. Run the `show pim join extensive instance VRF-26` command on SL-3, the ingress leaf device, to see the joined multicast groups, source address, upstream neighbors, and downstream neighbors. For more details on how enhanced OISM north-south traffic flow works, see *How Enhanced OISM Works*.

   Note the following in the output:

   - **Group**—Multicast groups 233.252.0.71 through 233.252.0.73 for intra-VLAN traffic (L2 forwarding), and 233.252.0.171 through 233.252.0.173 for inter-VLAN (L3 routing) traffic.

   - **Source**—The multicast source behind SL-3 has source address 10.0.201.12.

   - **Upstream Interface, Downstream neighbors**—The source VLAN is VLAN-201. SL-3 receives the traffic on irb.201 (the upstream interface). With enhanced OISM, SL-3 routes the multicast traffic to the other leaf devices only on the SBD, and the receiving leaf devices all receive the traffic on the SBD. As a result, the upstream interface would be irb.2026, the SBD IRB interface, for the receiving devices. Between the multihoming peer receiver devices BL-1 and BL-2, BL-2 received the PIM Join in this case, so BL-2 routes the traffic from the SBD toward the external receiver on its PEG interface.

   **SL-3 in the source role for the first multicast group (output is similar for the other groups):**

```
user@SL-3> show pim join extensive instance VRF-26 source 10.0.201.12 sg 233.252.0.71
Instance: PIM.VRF-26 Family: INET
R = Rendezvous Point Tree, S = Sparse, W = Wildcard

Group: 233.252.0.71
    Source: 10.0.201.12
    Flags: sparse,spt
    Upstream interface: irb.201
    Upstream neighbor: Direct
    Upstream state: Local Source, Local RP, No Prune to RP
    Keepalive timeout: 318
    Uptime: 01:09:47
    Downstream neighbors:
        Interface: irb.2026
            10.20.26.245 State: Join Flags: S   Timeout: Infinity
            Uptime: 01:09:47 Time since last Join: 01:09:47
    Number of downstream interfaces: 1
    Number of downstream neighbors: 1
```

2. Use the `show evpn oism spmsi-ad` command to see the source and group (S,G) information in the EVPN Type 10 S-PMSI A-D routes that SL-3 advertises and BL-2 receives.

For example, we show the output of this command for the source address in this use case (10.0.201.12) and the first intra-VLAN multicast group (233.252.0.71). The command and output are similar for the other multicast groups in this use case.

**SL-3:**

```
user@SL-3> show evpn oism spmsi-ad extensive source 10.0.201.12 group 233.252.0.71
Instance: MACVRF-1
  VN Identifier: 994027
    Group          Source        Local   Remote
    233.252.0.71   10.0.201.12     1       6
```

**BL-2:**

```
user@BL-2> show evpn oism spmsi-ad extensive source 10.0.201.12 group 233.252.0.71
Instance: MACVRF-1
  VN Identifier: 994027
    Group          Source        Local   Remote
    233.252.0.71   10.0.201.12     1       6
```

3. Run the `show route table MACVRF-1.evpn.0 match-prefix 10* extensive` command to see the details about the EVPN Type 10 routes advertised by SL-3, the device that hosts the multicast source.

   For example, we show the EVPN Type 10 route table entries on SL-3 (the source device), BL-2 (a receiver and PEG device), and SL-2 (another receiver). We include results for the first multicast group in this use case, 233.252.0.71. The commands and output are similar for the other leaf devices and multicast groups in this use case.

   **SL-3 in the source device role:**

```
user@SL-3> show route table MACVRF-1.evpn.0 match-prefix 10* extensive | find 10:192.168.0.3
10:192.168.0.3:33301::994027::10.0.201.12::233.252.0.71::192.168.0.3/520 (1 entry, 1
announced)
        *EVPN   Preference: 170
                Next hop type: Indirect, Next hop index: 0
                Address: 0x8252214
                Next-hop reference count: 8671
                Kernel Table Id: 0
                Protocol next hop: 192.168.0.3
                Indirect next hop: 0x0 - INH Session ID: 0
                Indirect next hop: INH non-key opaque: 0x0 INH key opaque: 0x0
                State: <Active Int Ext>
                Age: 16:12
                Validation State: unverified
                Task: MACVRF-1-evpn
                Announcement bits (1): 2-rt-export
                AS path: I
                Communities: encapsulation:vxlan(0x8)
                Thread: junos-main

```

SL-2 in the receiving device role:

```
user@SL-2> show route table MACVRF-1.evpn.0 match-prefix 10* extensive | find 10:192.168.0.3
10:192.168.0.3:33301::994027::10.0.201.12::233.252.0.71::192.168.0.3/520 (2 entries, 1
announced)
        *BGP    Preference: 170/-101
                Route Distinguisher: 192.168.0.3:33301
                Next hop type: Indirect, Next hop index: 0
                Address: 0x825ec94
                Next-hop reference count: 1488
                Kernel Table Id: 0
                Source: 192.168.2.2
                Protocol next hop: 192.168.0.3
                Indirect next hop: 0x2 no-forward INH Session ID: 0
                Indirect next hop: INH non-key opaque: 0x0 INH key opaque: 0x0
                State: <Secondary Active Ext>
                Peer AS: 4200000022
                Age: 18:37      Metric2: 0
                Validation State: unverified
                Task: BGP_4200000022_42000000.192.168.2.2
                Announcement bits (1): 0-MACVRF-1-evpn
                AS path: 4200000022 4200000013 I
                Communities: target:33300:1 encapsulation:vxlan(0x8)
                Import Accepted
                Localpref: 100
                Router ID: 192.168.2.2
                Primary Routing Table: bgp.evpn.0
                Thread: junos-main
                Indirect next hops: 1
                        Protocol next hop: 192.168.0.3 ResolvState: Resolved
                        Indirect next hop: 0x2 no-forward INH Session ID: 0
                        Indirect next hop: INH non-key opaque: 0x0 INH key opaque: 0x0
                        Indirect path forwarding next hops: 2
                                Next hop type: Router
                                Next hop: 172.16.2.0 via ae1.0
                                Session Id: 0
                                Next hop: 172.16.4.0 via ae2.0
                                Session Id: 0
                                192.168.0.3/32 Originating RIB: inet.0
                                  Node path count: 1
                                  Forwarding nexthops: 2
                                        Next hop type: Router
                                        Next hop: 172.16.2.0 via ae1.0
                                        Session Id: 0
                                        Next hop: 172.16.4.0 via ae2.0
                                        Session Id: 0
           BGP    Preference: 170/-101
                  Route Distinguisher: 192.168.0.3:33301
                  Next hop type: Indirect, Next hop index: 0
                  Address: 0x825ec94
                  Next-hop reference count: 1488
                  Kernel Table Id: 0
                  Source: 192.168.2.1
                  Protocol next hop: 192.168.0.3
                  Indirect next hop: 0x2 no-forward INH Session ID: 0
                  Indirect next hop: INH non-key opaque: 0x0 INH key opaque: 0x0
                  State: <Secondary Ext>
                  Inactive reason: Active preferred
                  Peer AS: 4200000021
                  Age: 18:37      Metric2: 0
                  Validation State: unverified
                  Task: BGP_4200000021_42000000.192.168.2.1
                  AS path: 4200000021 4200000013 I
                  Communities: target:33300:1 encapsulation:vxlan(0x8)
                  Import Accepted
                  Localpref: 100
                  Router ID: 192.168.2.1
                  Primary Routing Table: bgp.evpn.0
                  Thread: junos-main
                  Indirect next hops: 1
                        Protocol next hop: 192.168.0.3 ResolvState: Resolved
                        Indirect next hop: 0x2 no-forward INH Session ID: 0
                        Indirect next hop: INH non-key opaque: 0x0 INH key opaque: 0x0
                        Indirect path forwarding next hops: 2
                                Next hop type: Router
                                Next hop: 172.16.2.0 via ae1.0
                                Session Id: 0
                                Next hop: 172.16.4.0 via ae2.0
                                Session Id: 0
                                192.168.0.3/32 Originating RIB: inet.0
                                  Node path count: 1
                                  Forwarding nexthops: 2
                                        Next hop type: Router
                                        Next hop: 172.16.2.0 via ae1.0
                                        Session Id: 0
                                        Next hop: 172.16.4.0 via ae2.0
                                        Session Id: 0
```

BL-2 in the receiving device role and as a PEG device:

```
user@BL-2> show route table MACVRF-1.evpn.0 match-prefix 10* extensive | find 10:192.168.0.3
10:192.168.0.3:33301::994027::10.0.201.12::233.252.0.71::192.168.0.3/520 (2 entries, 1
announced)
        *BGP    Preference: 170/-101
                Route Distinguisher: 192.168.0.3:33301
                Next hop type: Indirect, Next hop index: 0
                Address: 0x8254114
                Next-hop reference count: 1488
                Kernel Table Id: 0
                Source: 192.168.2.2
                Protocol next hop: 192.168.0.3
                Indirect next hop: 0x2 no-forward INH Session ID: 0
                Indirect next hop: INH non-key opaque: 0x0 INH key opaque: 0x0
                State: <Secondary Active Ext>
                Peer AS: 4200000022
                Age: 19:33      Metric2: 0
                Validation State: unverified
                Task: BGP_4200000022_42000000.192.168.2.2
                Announcement bits (1): 0-MACVRF-1-evpn
                AS path: 4200000022 4200000013 I
                Communities: target:33300:1 encapsulation:vxlan(0x8)
                Import Accepted
                Localpref: 100
                Router ID: 192.168.2.2
                Primary Routing Table: bgp.evpn.0
                Thread: junos-main
                Indirect next hops: 1
                        Protocol next hop: 192.168.0.3 ResolvState: Resolved
                        Indirect next hop: 0x2 no-forward INH Session ID: 0
                        Indirect next hop: INH non-key opaque: 0x0 INH key opaque: 0x0
                        Indirect path forwarding next hops: 2
                                Next hop type: Router
                                Next hop: 172.16.12.0 via ae1.0
                                Session Id: 0
                                Next hop: 172.16.14.0 via ae2.0
                                Session Id: 0
                                192.168.0.3/32 Originating RIB: inet.0
                                  Node path count: 1
                                  Forwarding nexthops: 2
                                        Next hop type: Router
                                        Next hop: 172.16.12.0 via ae1.0
                                        Session Id: 0
                                        Next hop: 172.16.14.0 via ae2.0
                                        Session Id: 0
         BGP    Preference: 170/-101
                Route Distinguisher: 192.168.0.3:33301
                Next hop type: Indirect, Next hop index: 0
                Address: 0x8254114
                Next-hop reference count: 1488
                Kernel Table Id: 0
                Source: 192.168.2.1
                Protocol next hop: 192.168.0.3
                Indirect next hop: 0x2 no-forward INH Session ID: 0
                Indirect next hop: INH non-key opaque: 0x0 INH key opaque: 0x0
                State: <Secondary Ext Changed>
                Inactive reason: Active preferred
                Peer AS: 4200000021
                Age: 19:33      Metric2: 0
                Validation State: unverified
                Task: BGP_4200000021_42000000.192.168.2.1
                AS path: 4200000021 4200000013 I
                Communities: target:33300:1 encapsulation:vxlan(0x8)
                Import Accepted
                Localpref: 100
                Router ID: 192.168.2.1
                Primary Routing Table: bgp.evpn.0
                Thread: junos-main
                Indirect next hops: 1
                        Protocol next hop: 192.168.0.3 ResolvState: Resolved
```

```
                          Indirect next hop: 0x2 no-forward INH Session ID: 0
                          Indirect next hop: INH non-key opaque: 0x0 INH key opaque: 0x0
                          Indirect path forwarding next hops: 2
                                  Next hop type: Router
                                  Next hop: 172.16.12.0 via ae1.0
                                  Session Id: 0
                                  Next hop: 172.16.14.0 via ae2.0
                                  Session Id: 0
                                  192.168.0.3/32 Originating RIB: inet.0
                                    Node path count: 1
                                    Forwarding nexthops: 2
                                          Next hop type: Router
                                          Next hop: 172.16.12.0 via ae1.0
                                          Session Id: 0
                                          Next hop: 172.16.14.0 via ae2.0
                                          Session Id: 0
```

## Use Case #3: Internal Source to Internal Receivers with MLDv2—SSM

In use case #3, we configure the topology in Figure 96 on page 657 with a tenant VRF called VRF-56. This use case includes:

- An internal multicast source and internal multicast receivers.
- Inter-VLAN IPv6 multicast flows using MLDv2 (none of the OISM leaf devices host the source VLAN, so all OISM traffic will be inter-VLAN traffic)

Note that we support both IPv4 and IPv6 multicast data traffic with an IPv4 EVPN core. We use the same underlay and overlay peering for MLD multicast traffic as we use for IGMP in use case #1 and use case #2.

**Figure 96: Enhanced OISM Use Case #3 Topology—MLDv2 IPv6 Multicast with Internal Source Behind SL-6 and Internal Receivers**

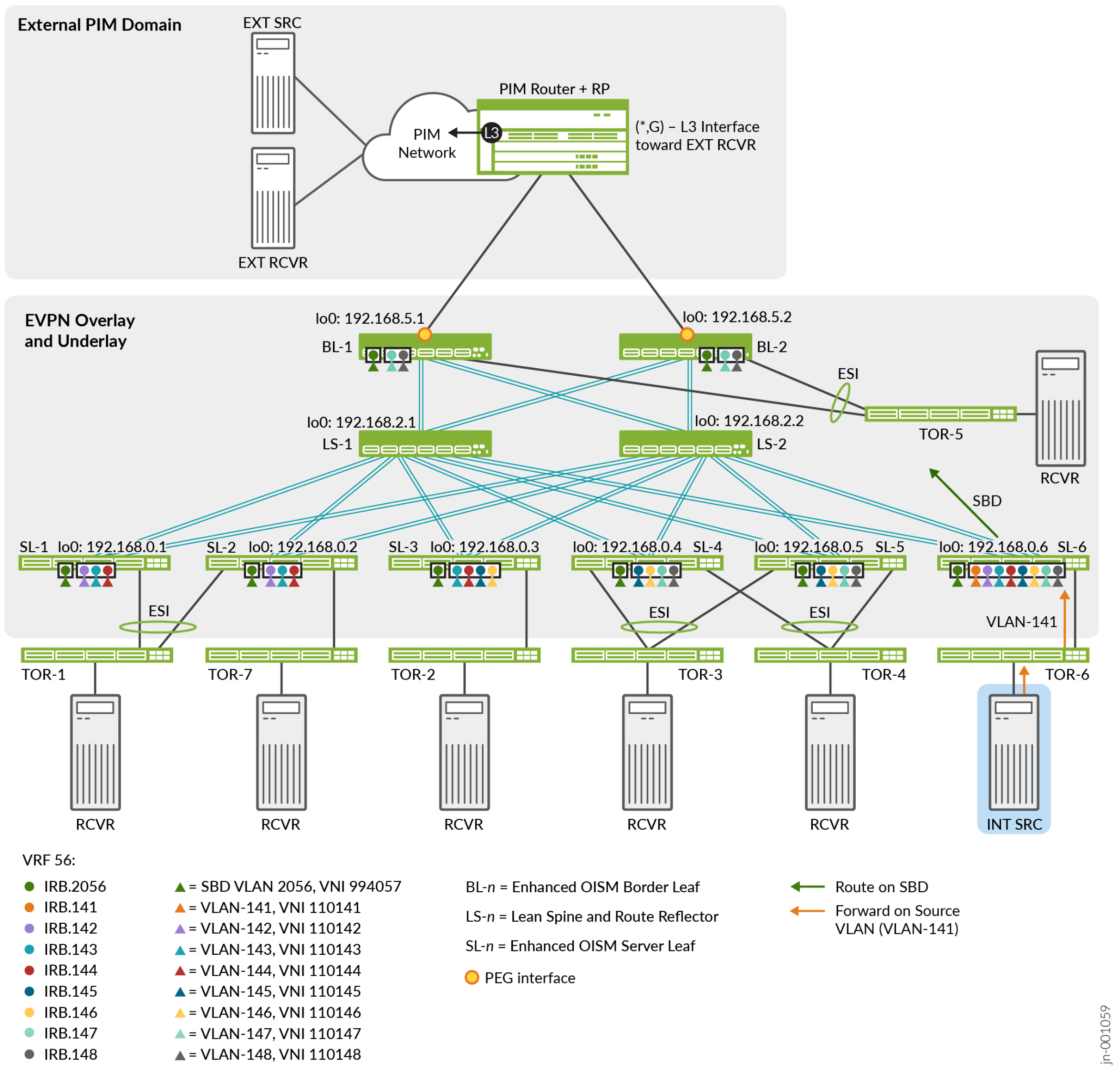


Table 25 on page 658 describes the multicast groups, device roles, configured VLANs, VXLAN VNI mappings for the VLANs, and the corresponding IRB interfaces for each VLAN.

**Table 25: Use Case #3 Elements for Internal Source to Internal Receivers with MLDv2**

| Role | Device | Configured Revenue VLANs | Configured IRB Interfaces | VXLAN VNI Mappings |
|---|---|---|---|---|
| SBD for VRF-56 on all enhanced OISM leaf devices: VLAN-2056, irb.2056, VNI 994057 | | | | |
| Multicast source VLAN: VLAN-141, Source Host IPv6 address: 2001:db8::10:0:8d:0c | | | | |
| MLDv2—SSM multicast groups for inter-VLAN traffic only: ff0e::db8:0:1 – ff0e::db8:0:3 for inter-VLAN (L3) | | | | |
| Source | TOR-6—Single-homed to SL-6 | VLAN-141 - VLAN-148 | irb.141 - irb.148 | VNI 110141 - VNI 110148 |
| Receivers | TOR-1—Multihomed to SL-1[1] and SL-2[1] | VLAN-142 - VLAN-144 | irb.142 - irb.144 | VNI 110142 - VNI 110144 |
| | TOR-7—Single-homed to SL-2[1] | VLAN-142 - VLAN-144 | irb.142 - irb.144 | VNI 110142 - VNI 110144 |
| | TOR-2—Single-homed to SL-3 | VLAN-143-VLAN-146 | irb.143- irb.146 | VNI 110143- VNI 110146 |
| | TOR-3—Multihomed to SL-4[2] and SL-5[2] | VLAN-145 - VLAN-148 | irb.145 - irb.148 | VNI 110145 - VNI 110148 |
| | TOR-4—Multihomed to SL-4[2] and SL-5[2] | VLAN-145 - VLAN-148 | irb.145 - irb.148 | VNI 110145 - VNI 110148 |
| | TOR-5—Multihomed to BL-1[3] and BL-2[3] | VLAN-147 - VLAN-148 | irb.147 - irb.148 | VNI 110147 - VNI 110148 |

Table 25: Use Case #3 Elements for Internal Source to Internal Receivers with MLDv2 *(Continued)*

| Role | Device | Configured Revenue VLANs | Configured IRB Interfaces | VXLAN VNI Mappings |
|---|---|---|---|---|

[1] SL-1 and SL-2 are multihoming peers, so configure the same revenue VLANs on SL-1 and SL-2.

[2] SL-4 and SL-5 are multihoming peers, so configure the same revenue VLANs on SL-4 and SL-5.

[3] BL-1 and BL-2 are multihoming peers, so configure the same revenue VLANs on BL-1 and BL-2.

## Configure Use Case #3: Internal Source and Receivers with MLDv2—SSM

Configure the revenue VLANs, SBD, tenant VRF, and multicast protocols specific to the use case in "Use Case #3: Internal Source to Internal Receivers with MLDv2—SSM" on page 656.

The main differences in this use case from use case #1 are that here:

- We use enhanced OISM for IPv6 multicast flows with MLDv2 from a single-homed source behind SL-6.
- SL-6 hosts revenue VLANs VLAN-141 through VLAN-148, and the other leaf devices host different subsets of revenue VLANs VLAN-142 through VLAN-148.
- We configure a VRF named VRF-56, and configure the SBD for this VRF using VLAN-2056.

1. On each OISM leaf device, configure the revenue VLANs the device hosts, their corresponding IRB interfaces, and VNI mappings in the same way as "Configure Use Case #1: Internal Source and Receivers with Multihoming Peer Receiver and IGMPv3—SSM" on page 592 for steps 1 through 3, but use the values in Table 25 on page 658, and refer to Table 26 on page 660 below for the IRB interface addresses and virtual gateway addresses.

**Table 26: Use Case #3 IRB Addresses and Virtual Gateway Addresses**

| Leaf Device | IRB Interface Unit# | IRB IPv4 Address<br><br>IPv4 VGA | IRB IPv6 Address<br><br>IPv6 VGA |
|---|---|---|---|
| SL-1 | Revenue VLANs—142 through 144 | 10.0.*unit#*.243/24<br><br>10.0.*unit#*.254 | 2001:db8::10:0:*hex-unit#*:243/112<br><br>2001:db8::10:0:*hex-unit#*:254 |
| | SBD—2056 | 10.20.56.243/24<br><br>10.20.56.254 | 2001:db8::10:0:808:243/112<br><br>2001:db8::10:0:808:254 |
| SL-2 | Revenue VLANs—142 through 144 | 10.0.*unit#*.244/24<br><br>10.0.*unit#*.254 | 2001:db8::10:0:*hex-unit#*:244/112<br><br>2001:db8::10:0:*hex-unit#*:254 |
| | SBD—2056 | 10.20.56.244/24<br><br>10.20.56.254 | 2001:db8::10:0:808:244/112<br><br>2001:db8::10:0:808:254 |
| SL-3 | Revenue VLANs—143 through 146 | 10.0.*unit#*.245/24<br><br>10.0.*unit#*.254 | 2001:db8::10:0:*hex-unit#*:245/112<br><br>2001:db8::10:0:*hex-unit#*:254 |
| | SBD—2056 | 10.20.56.245/24<br><br>10.20.56.254 | 2001:db8::10:0:808:245/112<br><br>2001:db8::10:0:808:254 |
| SL-4 | Revenue VLANs—145 through 148 | 10.0.*unit#*.246/24<br><br>10.0.*unit#*.254 | 2001:db8::10:0:*hex-unit#*:246/112<br><br>2001:db8::10:0:*hex-unit#*:254 |
| | SBD—2056 | 10.20.56.246/24<br><br>10.20.56.254 | 2001:db8::10:0:808:246/112<br><br>2001:db8::10:0:808:254 |

**Table 26: Use Case #3 IRB Addresses and Virtual Gateway Addresses *(Continued)***

| Leaf Device | IRB Interface Unit# | IRB IPv4 Address<br><br>IPv4 VGA | IRB IPv6 Address<br><br>IPv6 VGA |
|---|---|---|---|
| SL-5 | Revenue VLANs—145 through 148 | 10.0.*unit#*.247/24<br><br>10.0.*unit#*.254 | 2001:db8::10:0:*hex-unit#*:247/112<br><br>2001:db8::10:0:*hex-unit#*:254 |
| | SBD—2056 | 10.20.56.247/24<br><br>10.20.56.254 | 2001:db8::10:0:808:247/112<br><br>2001:db8::10:0:808:254 |
| SL-6 | Revenue VLANs—141 through 148 | 10.0.*unit#*.248/24<br><br>10.0.*unit#*.254 | 2001:db8::10:0:*hex-unit#*:248/112<br><br>2001:db8::10:0:*hex-unit#*:254 |
| | SBD—2056 | 10.20.56.248/24<br><br>10.20.56.254 | 2001:db8::10:0:808:248/112<br><br>2001:db8::10:0:808:254 |
| BL-1 | Revenue VLANs—147 and 148 | 10.0.*unit#*.241/24<br><br>10.0.*unit#*.254 | 2001:db8::10:0:*hex-unit#*:241/112<br><br>2001:db8::10:0:*hex-unit#*:254 |
| | SBD—2056 | 10.20.56.241/24<br><br>10.20.56.254 | 2001:db8::10:0:808:241/112<br><br>2001:db8::10:0:808:254 |
| BL-2 | Revenue VLANs—147 and 148 | 10.0.*unit#*.242/24<br><br>10.0.*unit#*.254 | 2001:db8::10:0:*hex-unit#*:242/112<br><br>2001:db8::10:0:*hex-unit#*:254 |
| | SBD—2056 | 10.20.56.242/24<br><br>10.20.56.254 | 2001:db8::10:0:808:242/112<br><br>2001:db8::10:0:808:254 |

2. This use case tests MLDv2 SSM flows, so enable MLD version 2 on the IRB interfaces for the revenue VLANs and the SBD in this use case.

   **All OISM leaf devices—SBD:**

   ```
   set protocols mld interface irb.2056 version 2
   ```

   **SL-1 and SL-2 (multihoming peer devices)—irb.142 through irb.144**:

   ```
   set protocols mld interface irb.142 version 2
   set protocols mld interface irb.143 version 2
   set protocols mld interface irb.144 version 2
   ```

   **SL-3—irb.143 through irb.146**:

   ```
   set protocols mld interface irb.143 version 2
   set protocols mld interface irb.144 version 2
   set protocols mld interface irb.145 version 2
   set protocols mld interface irb.146 version 2
   ```

   **SL-4 and SL-5 (multihoming peer devices)—irb.145 through irb.148**:

   ```
   set protocols mld interface irb.145 version 2
   set protocols mld interface irb.146 version 2
   set protocols mld interface irb.147 version 2
   set protocols mld interface irb.148 version 2
   ```

   **SL-6—irb.141 through irb.148**:

   ```
   set protocols mld interface irb.141 version 2
   set protocols mld interface irb.142 version 2
   set protocols mld interface irb.143 version 2
   set protocols mld interface irb.144 version 2
   set protocols mld interface irb.145 version 2
   set protocols mld interface irb.146 version 2
   set protocols mld interface irb.147 version 2
   set protocols mld interface irb.148 version 2
   ```

BL-1 and BL-2 (multihoming peer devices)—irb.147 and irb.148:

```
set protocols mld interface irb.147 version 2
set protocols mld interface irb.148 version 2
```

3. IGMP snooping is enabled by default on the OISM leaf devices. So on each OISM leaf device, disable IGMP snooping and enable MLD snooping for MLDv2 in the MACVRF-1 instance for the SBD and the revenue VLANs the device hosts.

   In EVPN-VXLAN fabrics, we support MLDv1 traffic with ASM reports only. We support MLDv2 traffic with SSM reports only. As a result, when you enable MLD snooping for MLDv2 traffic, you must include the SSM-specific *evpn-ssm-reports-only* configuration option. See *Supported IGMP or MLD Versions and Group Membership Report Modes* for more on ASM and SSM support with EVPN-VXLAN.

   All OISM leaf devices—SBD:

```
set routing-instances MACVRF-1 protocols igmp-snooping vlan VLAN-2056 disable
set routing-instances MACVRF-1 protocols mld-snooping vlan VLAN-2056 evpn-ssm-reports-only
```

   SL-1 and SL-2—VLANs 142 through 144:

```
set routing-instances MACVRF-1 protocols igmp-snooping vlan VLAN-142 disable
set routing-instances MACVRF-1 protocols igmp-snooping vlan VLAN-143 disable
set routing-instances MACVRF-1 protocols igmp-snooping vlan VLAN-144 disable
set routing-instances MACVRF-1 protocols mld-snooping vlan VLAN-142 evpn-ssm-reports-only
set routing-instances MACVRF-1 protocols mld-snooping vlan VLAN-143 evpn-ssm-reports-only
set routing-instances MACVRF-1 protocols mld-snooping vlan VLAN-144 evpn-ssm-reports-only
```

   SL-3—VLANs 143 through 146:

```
set routing-instances MACVRF-1 protocols igmp-snooping vlan VLAN-143 disable
set routing-instances MACVRF-1 protocols igmp-snooping vlan VLAN-144 disable
set routing-instances MACVRF-1 protocols igmp-snooping vlan VLAN-145 disable
set routing-instances MACVRF-1 protocols igmp-snooping vlan VLAN-146 disable
set routing-instances MACVRF-1 protocols mld-snooping vlan VLAN-143 evpn-ssm-reports-only
set routing-instances MACVRF-1 protocols mld-snooping vlan VLAN-144 evpn-ssm-reports-only
set routing-instances MACVRF-1 protocols mld-snooping vlan VLAN-145 evpn-ssm-reports-only
set routing-instances MACVRF-1 protocols mld-snooping vlan VLAN-146 evpn-ssm-reports-only
```

SL-4 and SL-5—VLANs 145 through 148:

```
set routing-instances MACVRF-1 protocols igmp-snooping vlan VLAN-145 disable
set routing-instances MACVRF-1 protocols igmp-snooping vlan VLAN-146 disable
set routing-instances MACVRF-1 protocols igmp-snooping vlan VLAN-147 disable
set routing-instances MACVRF-1 protocols igmp-snooping vlan VLAN-148 disable
set routing-instances MACVRF-1 protocols mld-snooping vlan VLAN-145 evpn-ssm-reports-only
set routing-instances MACVRF-1 protocols mld-snooping vlan VLAN-146 evpn-ssm-reports-only
set routing-instances MACVRF-1 protocols mld-snooping vlan VLAN-147 evpn-ssm-reports-only
set routing-instances MACVRF-1 protocols mld-snooping vlan VLAN-148 evpn-ssm-reports-only
```

SL-6—VLANs 141 through 148:

```
set routing-instances MACVRF-1 protocols igmp-snooping vlan VLAN-141 disable
set routing-instances MACVRF-1 protocols igmp-snooping vlan VLAN-142 disable
set routing-instances MACVRF-1 protocols igmp-snooping vlan VLAN-143 disable
set routing-instances MACVRF-1 protocols igmp-snooping vlan VLAN-144 disable
set routing-instances MACVRF-1 protocols igmp-snooping vlan VLAN-145 disable
set routing-instances MACVRF-1 protocols igmp-snooping vlan VLAN-146 disable
set routing-instances MACVRF-1 protocols igmp-snooping vlan VLAN-147 disable
set routing-instances MACVRF-1 protocols igmp-snooping vlan VLAN-148 disable
set routing-instances MACVRF-1 protocols mld-snooping vlan VLAN-141 evpn-ssm-reports-only
set routing-instances MACVRF-1 protocols mld-snooping vlan VLAN-142 evpn-ssm-reports-only
set routing-instances MACVRF-1 protocols mld-snooping vlan VLAN-143 evpn-ssm-reports-only
set routing-instances MACVRF-1 protocols mld-snooping vlan VLAN-144 evpn-ssm-reports-only
set routing-instances MACVRF-1 protocols mld-snooping vlan VLAN-145 evpn-ssm-reports-only
set routing-instances MACVRF-1 protocols mld-snooping vlan VLAN-146 evpn-ssm-reports-only
set routing-instances MACVRF-1 protocols mld-snooping vlan VLAN-147 evpn-ssm-reports-only
set routing-instances MACVRF-1 protocols mld-snooping vlan VLAN-148 evpn-ssm-reports-only
```

BL-1 and BL-2—VLANs 147 and 148:

```
set routing-instances MACVRF-1 protocols igmp-snooping vlan VLAN-147 disable
set routing-instances MACVRF-1 protocols igmp-snooping vlan VLAN-148 disable
set routing-instances MACVRF-1 protocols mld-snooping vlan VLAN-147 evpn-ssm-reports-only
set routing-instances MACVRF-1 protocols mld-snooping vlan VLAN-148 evpn-ssm-reports-only
```

**NOTE**: MLDv1 is the default version when you enable MLD and MLD snooping. If you have a use case with MLDv1—ASM flows, you configure MLD without the version option, and configure MLD snooping without the evpn-ssm-reports-only option, as follows:

```
set protocols mld interface irb.unit#
set routing-instances macvrf-instance protocols mld-snooping vlan vlan-name
```

4. On each OISM leaf device, configure a loopback logical interface for the VRF instance. We include this interface in the VRF instance, VRF-56, in the next step.

   For the loopback logical interface configuration, we use the following conventions:

   - The logical unit number matches the VRF instance number.
   - The last octet of the interface's IP address also matches the VRF instance number.

   SL-1:

   ```
   set interfaces lo0 unit 56 family inet address 10.3.1.56/32 primary
   ```

   SL-2:

   ```
   set interfaces lo0 unit 56 family inet address 10.3.2.56/32 primary
   ```

   SL-3:

   ```
   set interfaces lo0 unit 56 family inet address 10.4.3.56/32 primary
   ```

   SL-4:

   ```
   set interfaces lo0 unit 56 family inet address 10.7.4.56/32 primary
   ```

SL-5:

```
set interfaces lo0 unit 56 family inet address 10.7.5.56/32 primary
```

SL-6:

```
set interfaces lo0 unit 56 family inet address 10.8.6.56/32 primary
```

BL-1:

```
set interfaces lo0 unit 56 family inet address 10.1.1.56/32 primary
```

BL-2:

```
set interfaces lo0 unit 56 family inet address 10.1.2.56/32 primary
```

5. Configure the tenant VRF instance named VRF-56 on each of the OISM leaf devices.

   Include the same elements for the server leaf and border leaf devices in the VRF configuration as we describe in use case #1, Step 7, with the following differences for this use case:

   - Identify the OISM SBD IRB interface associated with this VRF instance, which is `irb.56`.
   - For IPv6 multicast traffic, also configure an OSPF version 3 (OSPFv3) area on the IRB interface for the SBD (in OSPF active mode) and the IRB interfaces for the hosted VLANs (in passive mode).
   - On the border leaf devices, the L3 PEG interface for VRF-56 is ae3.55 (logical unit = *VRF instance number - 1*).
   - On each device, include the loopback logical interface for this VRF instance that you configure in Step 4 above.
   - Use the same route target for this VRF instance on all of the OISM leaf devices—`target:100:56`.
   - Configure an RD following the same convention as in use case #1, in which the first part of the RD value mirrors the device loopback (unit 0) IP address, and the second part matches the VRF instance number—*`device-loopback-unit-0-address`*`:56`.

All server leaf devices SL-1 through SL-6:

```
set routing-instances VRF-56 instance-type vrf
set routing-instances VRF-56 routing-options graceful-restart
set routing-instances VRF-56 protocols evpn oism supplemental-bridge-domain-irb irb.2056
set routing-instances VRF-56 protocols ospf area 1.1.1.1 interface irb.2056 priority 0
set routing-instances VRF-56 protocols ospf area 1.1.1.1 interface all passive
set routing-instances VRF-56 protocols ospf3 area 1.1.1.1 interface irb.2056 priority 0
set routing-instances VRF-56 protocols ospf3 area 1.1.1.1 interface all passive
set routing-instances VRF-56 protocols pim passive
set routing-instances VRF-56 protocols pim interface all
set routing-instances VRF-56 protocols pim interface irb.2056 accept-remote-source
set routing-instances VRF-56 interface lo0.56
set routing-instances VRF-56 interface irb.2056
set routing-instances VRF-56 route-distinguisher device-loopback-unit-0-address:56
set routing-instances VRF-56 vrf-target target:100:56
```

Border leaf devices BL-1 and BL-2:

```
set policy-options policy-statement export-direct term t1 from protocol direct
set policy-options policy-statement export-direct term t1 then accept
set routing-instances VRF-56 instance-type vrf
set routing-instances VRF-56 routing-options graceful-restart
set routing-instances VRF-56 protocols evpn oism supplemental-bridge-domain-irb irb.2056
set routing-instances VRF-56 protocols ospf area 1.1.1.1 interface ae3.55 mtu 9178
set routing-instances VRF-56 protocols ospf area 1.1.1.1 interface irb.2056set routing-
instances VRF-56 protocols ospf area 1.1.1.1 interface all passive
set routing-instances VRF-56 protocols ospf export export-direct
set routing-instances VRF-56 protocols ospf3 area 1.1.1.1 interface irb.2056 priority 0
set routing-instances VRF-56 protocols ospf3 area 1.1.1.1 interface all passive
set routing-instances VRF-56 protocols pim rp static address pim-rp-IP-address
set routing-instances VRF-56 protocols pim interface irb.147 distributed-dr
set routing-instances VRF-56 protocols pim interface irb.148 distributed-dr
set routing-instances VRF-56 protocols pim interface lo0.56
set routing-instances VRF-56 protocols pim interface ae3.55
set routing-instances VRF-56 protocols pim interface irb.2056 family inet bfd-liveness-
detection minimum-interval 1000
set routing-instances VRF-56 protocols pim interface irb.2056 family inet bfd-liveness-
detection multiplier 3
set routing-instances VRF-56 protocols pim interface irb.2056 stickydr
set routing-instances VRF-56 protocols pim interface irb.2056 hello-interval 30
set routing-instances VRF-56 protocols pim interface irb.2056 accept-remote-source
set routing-instances VRF-56 protocols pim disable-packet-register
set routing-instances VRF-56 interface lo0.56
set routing-instances VRF-56 interface ae3.55
set routing-instances VRF-56 interface irb.2056
set routing-instances VRF-56 route-distinguisher device-loopback-unit-0-address:56
set routing-instances VRF-56 vrf-target target:100:56
```

Then in the VRF instance configuration on each of the leaf devices, add the IRB interfaces only for the revenue VLANs the device hosts:

**SL-1 and SL-2—irb.142 through irb.144:**

```
set routing-instances VRF-56 interface irb.142
set routing-instances VRF-56 interface irb.143
set routing-instances VRF-56 interface irb.144
```

**SL-3—irb.143 through irb.146:**

```
set routing-instances VRF-56 interface irb.143
set routing-instances VRF-56 interface irb.144
set routing-instances VRF-56 interface irb.145
set routing-instances VRF-56 interface irb.146
```

**SL-4 and SL-5—irb.145 through irb.148:**

```
set routing-instances VRF-56 interface irb.145
set routing-instances VRF-56 interface irb.146
set routing-instances VRF-56 interface irb.147
set routing-instances VRF-56 interface irb.148
```

**SL-6—irb.141 through irb.148:**

```
set routing-instances VRF-56 interface irb.141
set routing-instances VRF-56 interface irb.142
set routing-instances VRF-56 interface irb.143
set routing-instances VRF-56 interface irb.144
set routing-instances VRF-56 interface irb.145
set routing-instances VRF-56 interface irb.146
set routing-instances VRF-56 interface irb.147
set routing-instances VRF-56 interface irb.148
```

BL-1 and BL-2—irb.147 and irb.148:

```
set routing-instances VRF-56 interface irb.147
set routing-instances VRF-56 interface irb.148
```

## Verify Use Case #3: Internal Source and Receivers with MLDv2—SSM

Verify enhanced OISM operation for use case #3, in which the internal source is behind SL-6 sending IPv6 multicast flows for inter-VLAN multicast traffic with MLDv2—SSM. SL-1 through SL-5, BL-1, and BL-2 are internal receivers. See the topology in Figure 96 on page 657, and the configuration in "Configure Use Case #3: Internal Source and Receivers with MLDv2—SSM" on page 659.

Verification for this use case is similar to the verification in use case #1 but with an IPv6 source address and IPv6 multicast traffic. We include verification commands here on SL-6 as the OISM leaf device sending the source traffic, and on SL-1 and SL-2 as the OISM leaf devices receiving the traffic. On the other receiving devices, based on the revenue VLANs they host, you'll see output similar to what we include here for SL-1 and SL-2.

1. Run the `show vlans` command for the revenue VLANs and the SBD to see the VTEPs between the leaf devices.

   For example, on SL-6, you see the following:

   - For VLAN-141—No VTEPS. SL-6 doesn't need any VTEPs to other OISM leaf devices for the source VLAN because:

     - This use case is for inter-VLAN traffic only, so no other leaf devices host the source VLAN.

     - SL-6 has no multihoming peer leaf devices , so SL-6 doesn't need to send any multicast flows to a multihoming peer on the source VLAN.

   - For VLAN-142—2 VTEPs for the leaf devices that host VLAN-142 (SL-1 and SL-2)

   - For VLAN-143—3 VTEPs for the leaf devices that host VLAN-143 (SL-1, SL-2, and SL-3)

   - For VLAN-144—3 VTEPs for the leaf devices that host VLAN-144 (SL-1, SL-2, and SL-3)

   - For VLAN-145—3 VTEPs for the leaf devices that host VLAN-145 (SL-3, SL-4, and SL-5)

   - For VLAN-146—3 VTEPs for the leaf devices that host VLAN-146 (SL-3, SL-4, and SL-5)

- For VLAN-147—4 VTEPs for the leaf devices that host VLAN-147 (SL-4, SL-5, BL-1, and BL-2)
- For VLAN-148—4 VTEPs for the leaf devices that host VLAN-148 (SL-4, SL-5, BL-1, and BL-2)

SL-6:

```
user@SL-6> show vlans 141

Routing instance        VLAN name           Tag          Interfaces
MACVRF-1                VLAN-141            141
                                                         ae3.0*

user@SL-6> show vlans 142
Routing instance        VLAN name           Tag          Interfaces
MACVRF-1                VLAN-142            142
                                                         ae3.0*
                                                         esi.32245*
                                                         vtep-133.32803*
                                                         vtep-133.32806*

ruser@SL-6> show vlans 143
Routing instance        VLAN name           Tag          Interfaces
MACVRF-1                VLAN-143            143
                                                         ae3.0*
                                                         esi.32245*
                                                         vtep-133.32802*
                                                         vtep-133.32803*
                                                         vtep-133.32806*

user@SL-6> show vlans 144
Routing instance        VLAN name           Tag          Interfaces
MACVRF-1                VLAN-144            144
                                                         ae3.0*
                                                         esi.32245*
                                                         vtep-133.32802*
                                                         vtep-133.32803*
                                                         vtep-133.32806*

user@SL-6> show vlans 145

Routing instance        VLAN name           Tag          Interfaces
MACVRF-1                VLAN-145            145
                                                         ae3.0*
```

```
                                                          vtep-133.32802*
                                                          vtep-133.32805*
                                                          vtep-133.32807*

user@SL-6> show vlans 146
Routing instance        VLAN name             Tag          Interfaces
MACVRF-1                VLAN-146              146
                                                          ae3.0*
                                                          vtep-133.32802*
                                                          vtep-133.32805*
                                                          vtep-133.32807*

user@SL-6> show vlans 147

Routing instance        VLAN name             Tag          Interfaces
MACVRF-1                VLAN-147              147
                                                          ae3.0*
                                                          esi.42632*
                                                          vtep-133.32804*
                                                          vtep-133.32805*
                                                          vtep-133.32807*
                                                          vtep-133.32808*

user@SL-6> show vlans 148

Routing instance        VLAN name             Tag          Interfaces
MACVRF-1                VLAN-148              148
                                                          ae3.0*
                                                          esi.42632*
                                                          vtep-133.32804*
                                                          vtep-133.32805*
                                                          vtep-133.32807*
                                                          vtep-133.32808*
```

SL-2 (SL-1 and SL-2 host only VLAN-142, VLAN-143, and VLAN-144):

```
user@SL-2> show vlans 141

user@SL-2> show vlans 142

Routing instance        VLAN name             Tag          Interfaces
```

```
MACVRF-1                VLAN-142              142
                                                           ae3.0*
                                                           ae5.0*
                                                           esi.18026*
                                                           vtep-5.32773*
                                                           vtep-5.32774*

user@SL-2> show vlans 143

Routing instance        VLAN name             Tag          Interfaces
MACVRF-1                VLAN-143              143
                                                           ae3.0*
                                                           ae5.0*
                                                           esi.18026*
                                                           vtep-5.32772*
                                                           vtep-5.32773*
                                                           vtep-5.32774*

user@SL-2> show vlans 144

Routing instance        VLAN name             Tag          Interfaces
MACVRF-1                VLAN-144              144
                                                           ae3.0*
                                                           ae5.0*
                                                           esi.18026*
                                                           vtep-5.32772*
                                                           vtep-5.32773*
                                                           vtep-5.32774*

user@SL-2> show vlans 145

user@SL-2> show vlans 146

user@SL-2> show vlans 147

user@SL-2> show vlans 148
```

2. Run the `show vlans` command for the SBD (VLAN-2056) on each leaf device to see the SBD VTEPs between the leaf devices.

   You should see VTEPs to each of the other leaf devices (seven VTEPs in this case).

SL-6:

```
user@SL-6> show vlans 2056

Routing instance        VLAN name            Tag          Interfaces
MACVRF-1                VLAN-2056            2056
                                                          vtep-133.32802*
                                                          vtep-133.32803*
                                                          vtep-133.32804*
                                                          vtep-133.32805*
                                                          vtep-133.32806*
                                                          vtep-133.32807*
                                                          vtep-133.32808*
```

SL-2 (output is similar for SL-1):

```
user@SL-2> show vlans 2056

Routing instance        VLAN name            Tag          Interfaces
MACVRF-1                VLAN-2056            2056
                                                          vtep-5.32769*
                                                          vtep-5.32770*
                                                          vtep-5.32771*
                                                          vtep-5.32772*
                                                          vtep-5.32773*
                                                          vtep-5.32774*
                                                          vtep-5.32775*
```

3. Run the `show interfaces irb.`*`unit#`* `terse` command on each of the leaf devices to verify the revenue VLAN IRB interfaces are up, and the SBD IRB interface is up.

   Table 26 on page 660 lists the addresses of the IRB interfaces in this use case.

   **SL-6 (output is similar on other OISM leaf devices for the SBD and the revenue VLANs they host):**

```
user@SL-6> show interfaces terse | grep 141
irb.141                 up    up   inet     10.0.141.248/24

user@SL-6> show interfaces terse | grep 142
irb.142                 up    up   inet     10.0.142.248/24
```

```
user@SL-6> show interfaces terse | grep 143
irb.143                 up    up   inet     10.0.143.248/24

user@SL-6> show interfaces terse | grep 144
irb.144                 up    up   inet     10.0.144.248/24

user@SL-6> show interfaces terse | grep 145
irb.145                 up    up   inet     10.0.145.248/24

user@SL-6> show interfaces terse | grep 146
irb.146                 up    up   inet     10.0.146.248/24

user@SL-6> show interfaces terse | grep 147
irb.147                 up    up   inet     10.0.147.248/24

user@SL-6> show interfaces terse | grep 148
irb.148                 up    up   inet     10.0.148.248/24

user@SL-6> show interfaces terse | grep 2056
irb.2056                up    up   inet     10.20.56.248/24
```

4. Run the `show ospf3 neighbor` command for the VRF instance in this use case, VRF-56, on each leaf device.

   The output shows the OSPFv3 neighbors, where the neighbor `ID` is the neighbor device's logical loopback address for the VRF (lo0.56). The `Neighbor-address` is the automatically-assigned IPv6 link local address.

   **SL-6:**

```
user@SL-6> show ospf3 neighbor instance VRF-56
ID                Interface               State     Pri   Dead
10.3.2.56         irb.2056                2Way        0     35
  Neighbor-address fe80::8243:3f08:8e2:5d80
10.1.2.56         irb.2056                Full      128     32
  Neighbor-address fe80::8a30:3708:8a2:c9dd
10.3.1.56         irb.2056                2Way        0     30
  Neighbor-address fe80::8243:3f08:8e2:4980
10.4.3.56         irb.2056                2Way        0     39
  Neighbor-address fe80::8a30:3708:876:f630
10.7.4.56         irb.2056                2Way        0     39
  Neighbor-address fe80::9a49:2508:883:f49a
```

```
10.7.5.56        irb.2056              2Way        0     39
  Neighbor-address fe80::d699:6c08:87a:7ef7
```

SL-1:

```
user@SL-1> show ospf3 neighbor instance VRF-56
ID               Interface             State     Pri   Dead
10.3.2.56        irb.2056              2Way        0     30
  Neighbor-address fe80::8243:3f08:8e2:5d80
10.1.2.56        irb.2056              Full      128     36
  Neighbor-address fe80::8a30:3708:8a2:c9dd
10.7.5.56        irb.2056              2Way        0     33
  Neighbor-address fe80::d699:6c08:87a:7ef7
10.8.6.56        irb.2056              2Way        0     39
  Neighbor-address fe80::d699:6c08:8cc:a8e
10.7.4.56        irb.2056              2Way        0     39
  Neighbor-address fe80::9a49:2508:883:f49a
10.4.3.56        irb.2056              2Way        0     34
  Neighbor-address fe80::8a30:3708:876:f630
```

SL-2:

```
user@SL-2> show ospf3 neighbor instance VRF-56
ID               Interface             State     Pri   Dead
10.7.4.56        irb.2056              2Way        0     31
  Neighbor-address fe80::9a49:2508:883:f49a
10.8.6.56        irb.2056              2Way        0     38
  Neighbor-address fe80::d699:6c08:8cc:a8e
10.4.3.56        irb.2056              2Way        0     37
  Neighbor-address fe80::8a30:3708:876:f630
10.3.1.56        irb.2056              2Way        0     35
  Neighbor-address fe80::8243:3f08:8e2:4980
10.1.2.56        irb.2056              Full      128     39
  Neighbor-address fe80::8a30:3708:8a2:c9dd
10.7.5.56        irb.2056              2Way        0     35
  Neighbor-address fe80::d699:6c08:87a:7ef7
```

5. Run the `show pim join extensive instance VRF-56` command on the leaf devices to see the joined IPv6 multicast groups, IPv6 source address, upstream neighbors, and downstream neighbors. For more details on how enhanced OISM east-west traffic flow works, see *How Enhanced OISM Works*.

   We show output from this command for IPv6 multicast traffic on SL-6 as the source leaf device, and SL-2 as a receiving leaf device. Output is similar on the other receiving leaf devices for the VLANs they host. Note the following in the output:

   - **Group**—We use multicast groups ff0e::db8:0:1 through ff0e::db8:0:3 for inter-VLAN (L3 routing) traffic. We truncate the output after the first two multicast groups; the output for the last group is similar.

   - **Source**—The MLDv2 SSM multicast source is behind SL-6 with source address 2001:db8::10:0:8d:0c.

   - **Upstream Interface, Downstream neighbors**—The source VLAN is VLAN-141, but in this case (unlike use case #1), because SL-6 has no multihoming peer leaf devices, the upstream interface is the SBD IRB interface, irb.2056. With enhanced OISM, SL-6 routes all multicast traffic to the other leaf devices on the SBD, and the receiving leaf devices receive the traffic on the SBD. Then the receiving leaf devices route the traffic from the SBD to the destination VLANs.

   **SL-6 in the source role:**

```
user@SL-6> show pim join extensive instance VRF-56
Instance: PIM.VRF-56 Family: INET
R = Rendezvous Point Tree, S = Sparse, W = Wildcard

Instance: PIM.VRF-56 Family: INET6
R = Rendezvous Point Tree, S = Sparse, W = Wildcard

Group: ff0e::db8:0:1
    Source: 2001:db8::10:0:8d:0c
    Flags: sparse,spt
    Upstream interface: irb.141
    Upstream neighbor: Direct
    Upstream state: Local Source, Local RP
    Keepalive timeout:
    Uptime: 00:40:33
    Downstream neighbors:
        Interface: irb.2056
            fe80::d699:6c08:8cc:a8e State: Join Flags: S   Timeout: Infinity
            Uptime: 00:40:33 Time since last Join: 00:40:33
    Number of downstream interfaces: 1
    Number of downstream neighbors: 1
```

```
Group: ff0e::db8:0:2
    Source: 2001:db8::10:0:8d:0c
    Flags: sparse,spt
    Upstream interface: irb.141
    Upstream neighbor: Direct
    Upstream state: Local Source, Local RP
    Keepalive timeout:
    Uptime: 00:40:33
    Downstream neighbors:
        Interface: irb.2056
            fe80::d699:6c08:8cc:a8e State: Join Flags: S   Timeout: Infinity
            Uptime: 00:40:33 Time since last Join: 00:40:33
    Number of downstream interfaces: 1
    Number of downstream neighbors: 1

...
```

**SL-1 in the receiver role (SL-1 and SL-2 share an ESI and host VLANs 142 through 144; SL-2 is the designated forwarder [DF] for the ESI):**

```
user@SL-1> show pim join extensive instance VRF-56
Instance: PIM.VRF-56 Family: INET
R = Rendezvous Point Tree, S = Sparse, W = Wildcard

Instance: PIM.VRF-56 Family: INET6
R = Rendezvous Point Tree, S = Sparse, W = Wildcard

Group: ff0e::db8:0:1
    Source: 2001:db8::10:0:8d:0c
    Flags: sparse,spt
    Upstream interface: irb.2056
    Upstream neighbor: fe80::8243:3f08:8e2:4980
    Upstream state: Local Source, Local RP
    Keepalive timeout:
    Uptime: 6d 04:40:25
    Downstream neighbors:
        Interface: irb.142
            fe80::8243:3f01:bae2:4980 State: Join Flags: S   Timeout: Infinity
            Uptime: 6d 04:40:25 Time since last Join: 5d 06:15:17
        Interface: irb.143
```

```
            fe80::8243:3f01:bbe2:4980 State: Join Flags: S   Timeout: Infinity
            Uptime: 6d 04:40:17 Time since last Join: 5d 06:15:17
        Interface: irb.144
            fe80::8243:3f01:bce2:4980 State: Join Flags: S   Timeout: Infinity
            Uptime: 6d 04:40:21 Time since last Join: 5d 06:15:17
        Interface: irb.2056
            fe80::8243:3f08:8e2:4980 State: Join Flags: S   Timeout: Infinity
            Uptime: 6d 03:41:58 Time since last Join: 5d 06:15:17
    Number of downstream interfaces: 4
    Number of downstream neighbors: 4

Group: ff0e::db8:0:2
    Source: 2001:db8::10:0:8d:0c
    Flags: sparse,spt
    Upstream interface: irb.2056
    Upstream neighbor: fe80::8243:3f08:8e2:4980
    Upstream state: Local Source, Local RP
    Keepalive timeout:
    Uptime: 6d 04:40:25
    Downstream neighbors:
        Interface: irb.142
            fe80::8243:3f01:bae2:4980 State: Join Flags: S   Timeout: Infinity
            Uptime: 6d 04:40:25 Time since last Join: 5d 06:15:17
        Interface: irb.143
            fe80::8243:3f01:bbe2:4980 State: Join Flags: S   Timeout: Infinity
            Uptime: 6d 04:40:17 Time since last Join: 5d 06:15:17
        Interface: irb.144
            fe80::8243:3f01:bce2:4980 State: Join Flags: S   Timeout: Infinity
            Uptime: 6d 04:40:21 Time since last Join: 5d 06:15:17
        Interface: irb.2056
            fe80::8243:3f08:8e2:4980 State: Join Flags: S   Timeout: Infinity
            Uptime: 6d 03:41:58 Time since last Join: 5d 06:15:17
    Number of downstream interfaces: 4
    Number of downstream neighbors: 4

...
```

SL-2 in the receiver role (SL-1 and SL-2 share an ESI and host VLANs 142 through 144; SL-2 is the designated forwarder [DF] for the ESI):

```
user@SL-2> show pim join extensive instance VRF-56
Instance: PIM.VRF-56 Family: INET
R = Rendezvous Point Tree, S = Sparse, W = Wildcard

Instance: PIM.VRF-56 Family: INET6
R = Rendezvous Point Tree, S = Sparse, W = Wildcard

Group: ff0e::db8:0:1
    Source: 2001:db8::10:0:8d:0c
    Flags: sparse,spt
    Upstream interface: irb.2056
    Upstream neighbor: fe80::8243:3f08:8e2:5d80
    Upstream state: Local Source, Local RP
    Keepalive timeout:
    Uptime: 00:11:38
    Downstream neighbors:
        Interface: irb.2056
            fe80::8243:3f08:8e2:5d80 State: Join Flags: S   Timeout: Infinity
            Uptime: 00:11:38 Time since last Join: 00:11:29
        Interface: irb.142
            fe80::8243:3f01:bae2:5d80 State: Join Flags: S   Timeout: Infinity
            Uptime: 00:11:38 Time since last Join: 00:11:38
        Interface: irb.143
            fe80::8243:3f01:bbe2:5d80 State: Join Flags: S   Timeout: Infinity
            Uptime: 00:11:38 Time since last Join: 00:11:38
        Interface: irb.144
            fe80::8243:3f01:bce2:5d80 State: Join Flags: S   Timeout: Infinity
            Uptime: 00:11:38 Time since last Join: 00:11:38
    Number of downstream interfaces: 4
    Number of downstream neighbors: 4

Group: ff0e::db8:0:2
    Source: 2001:db8::10:0:8d:0c
    Flags: sparse,spt
    Upstream interface: irb.2056
    Upstream neighbor: fe80::8243:3f08:8e2:5d80
    Upstream state: Local Source, Local RP
    Keepalive timeout:
    Uptime: 00:11:38
    Downstream neighbors:
        Interface: irb.2056
            fe80::8243:3f08:8e2:5d80 State: Join Flags: S   Timeout: Infinity
            Uptime: 00:11:38 Time since last Join: 00:11:29
        Interface: irb.142
            fe80::8243:3f01:bae2:5d80 State: Join Flags: S   Timeout: Infinity
            Uptime: 00:11:38 Time since last Join: 00:11:38
        Interface: irb.143
            fe80::8243:3f01:bbe2:5d80 State: Join Flags: S   Timeout: Infinity
            Uptime: 00:11:38 Time since last Join: 00:11:38
        Interface: irb.144
            fe80::8243:3f01:bce2:5d80 State: Join Flags: S   Timeout: Infinity
            Uptime: 00:11:38 Time since last Join: 00:11:38
    Number of downstream interfaces: 4
    Number of downstream neighbors: 4

...
```

6. On each device, use the `show mld snooping membership vlan` *`vlan-name`* `virtual-switch MACVRF-1` command to verify the multicast (S,G) source and group information learned with MLD snooping on the device for the hosted VLANs.

   For example, on SL-2 as a receiving leaf device:

```
user@SL-2> show mld snooping membership vlan VLAN-142 virtual-switch MACVRF-1
Instance: MACVRF-1

Vlan: VLAN-142

Learning-Domain: default
Interface: ae3.0, Groups: 0

Learning-Domain: default
Interface: ae5.0, Groups: 3
    Group: ff0e::db8:0:1
        Group mode: Include
        Source: 2001:db8::10:0:8d:0c
        Source timeout: 191
        Last reported by: fe80::11:0:1ba:2
        Group timeout:       0 Type: Dynamic
    Group: ff0e::db8:0:2
        Group mode: Include
        Source: 2001:db8::10:0:8d:0c
        Source timeout: 191
        Last reported by: fe80::11:0:1ba:2
        Group timeout:       0 Type: Dynamic
    Group: ff0e::db8:0:3
        Group mode: Include
        Source: 2001:db8::10:0:8d:0c
        Source timeout: 191
        Last reported by: fe80::11:0:1ba:2
        Group timeout:       0 Type: Dynamic

user@SL-2> show mld snooping membership vlan VLAN-143 virtual-switch MACVRF-1
Instance: MACVRF-1

Vlan: VLAN-143

Learning-Domain: default
Interface: ae3.0, Groups: 3
    Group: ff0e::db8:0:1
        Group mode: Include
        Source: 2001:db8::10:0:8d:0c
        Source timeout: 254
        Last reported by: fe80::11:0:1ba:3
        Group timeout:       0 Type: Dynamic
    Group: ff0e::db8:0:2
        Group mode: Include
        Source: 2001:db8::10:0:8d:0c
        Source timeout: 254
        Last reported by: fe80::11:0:1ba:3
        Group timeout:       0 Type: Dynamic
    Group: ff0e::db8:0:3
        Group mode: Include
        Source: 2001:db8::10:0:8d:0c
        Source timeout: 254
        Last reported by: fe80::11:0:1ba:3
        Group timeout:       0 Type: Dynamic

Learning-Domain: default
Interface: ae5.0, Groups: 3
    Group: ff0e::db8:0:1
        Group mode: Include
        Source: 2001:db8::10:0:8d:0c
        Source timeout: 193
        Last reported by: fe80::11:0:1ba:3
        Group timeout:       0 Type: Dynamic
    Group: ff0e::db8:0:2
        Group mode: Include
        Source: 2001:db8::10:0:8d:0c
        Source timeout: 193
        Last reported by: fe80::11:0:1ba:3
        Group timeout:       0 Type: Dynamic
    Group: ff0e::db8:0:3
        Group mode: Include
        Source: 2001:db8::10:0:8d:0c
        Source timeout: 193
        Last reported by: fe80::11:0:1ba:3
        Group timeout:       0 Type: Dynamic

...
```

7. Run the `show multicast snooping route source-prefix 2001:db8::10:0:8d:0c instance VRF-56 extensive` command on the leaf devices to check for the expected routes in the multicast snooping route table with the outgoing interface for the IPv6 multicast (S,G) traffic in this use case.

   We include the output on SL-1 and SL-2 as receiving leaf devices. The output here shows that SL-1 and SL-2 receive the traffic on the SBD for the subscribed (S,G) entries and route the traffic toward the receivers on the destination VLANs they host.

   **SL-1 (SL-1 and SL-2 host VLANs 142 through 144):**

```
user@SL-1> show multicast snooping route source-prefix 2001:db8::10:0:8d:0c instance MACVRF-1
Nexthop Bulking: OFF

             Family: INET6

Group: ff0e::db8:0:1/128
    Source: 2001:db8::10:0:8d:0c/128
    Vlan: VLAN-2056
        Downstream interface list:
        evpn-core-nh

Group: ff0e::db8:0:2/128
    Source: 2001:db8::10:0:8d:0c/128
    Vlan: VLAN-2056
        Downstream interface list:
```

```
        evpn-core-nh

Group: ff0e::db8:0:3/128
    Source: 2001:db8::10:0:8d:0c/128
    Vlan: VLAN-2056
        Downstream interface list:
        evpn-core-nh

Group: ff0e::db8:0:1/128
    Source: 2001:db8::10:0:8d:0c/128
    Vlan: VLAN-142
        Downstream interface list:
        ae3.0

Group: ff0e::db8:0:2/128
    Source: 2001:db8::10:0:8d:0c/128
    Vlan: VLAN-142
        Downstream interface list:
        ae3.0

Group: ff0e::db8:0:3/128
    Source: 2001:db8::10:0:8d:0c/128
    Vlan: VLAN-142
        Downstream interface list:
        ae3.0

Group: ff0e::db8:0:1/128
    Source: 2001:db8::10:0:8d:0c/128
    Vlan: VLAN-143
        Downstream interface list:
        ae3.0

Group: ff0e::db8:0:2/128
    Source: 2001:db8::10:0:8d:0c/128
    Vlan: VLAN-143
        Downstream interface list:
        ae3.0

Group: ff0e::db8:0:3/128
    Source: 2001:db8::10:0:8d:0c/128
    Vlan: VLAN-143
        Downstream interface list:
        ae3.0
```

```

Group: ff0e::db8:0:1/128
    Source: 2001:db8::10:0:8d:0c/128
    Vlan: VLAN-144
        Downstream interface list:
        ae3.0

Group: ff0e::db8:0:2/128
    Source: 2001:db8::10:0:8d:0c/128
    Vlan: VLAN-144
        Downstream interface list:
        ae3.0

Group: ff0e::db8:0:3/128
    Source: 2001:db8::10:0:8d:0c/128
    Vlan: VLAN-144
        Downstream interface list:
        ae3.0

```

SL-2 (SL-1 and SL-2 host VLANs 142 through 144, and SL-2 has an additional single-homed receiver on interface ae5 to TOR-7):

```
user@SL-1> show multicast snooping route source-prefix 2001:db8::10:0:8d:0c instance
MACVRF-1
Nexthop Bulking: OFF

             Family: INET6

Group: ff0e::db8:0:1/128
    Source: 2001:db8::10:0:8d:0c/128
    Vlan: VLAN-2056
        Downstream interface list:
        evpn-core-nh

Group: ff0e::db8:0:2/128
    Source: 2001:db8::10:0:8d:0c/128
    Vlan: VLAN-2056
        Downstream interface list:
        evpn-core-nh

Group: ff0e::db8:0:3/128
```

```
    Source: 2001:db8::10:0:8d:0c/128
    Vlan: VLAN-2056
        Downstream interface list:
        evpn-core-nh

Group: ff0e::db8:0:1/128
    Source: 2001:db8::10:0:8d:0c/128
    Vlan: VLAN-142
        Downstream interface list:
        ae3.0 ae5.0

Group: ff0e::db8:0:2/128
    Source: 2001:db8::10:0:8d:0c/128
    Vlan: VLAN-142
        Downstream interface list:
        ae3.0 ae5.0

Group: ff0e::db8:0:3/128
    Source: 2001:db8::10:0:8d:0c/128
    Vlan: VLAN-142
        Downstream interface list:
        ae3.0 ae5.0

Group: ff0e::db8:0:1/128
    Source: 2001:db8::10:0:8d:0c/128
    Vlan: VLAN-143
        Downstream interface list:
        ae3.0 ae5.0

Group: ff0e::db8:0:2/128
    Source: 2001:db8::10:0:8d:0c/128
    Vlan: VLAN-143
        Downstream interface list:
        ae3.0 ae5.0

Group: ff0e::db8:0:3/128
    Source: 2001:db8::10:0:8d:0c/128
    Vlan: VLAN-143
        Downstream interface list:
        ae3.0 ae5.0

Group: ff0e::db8:0:1/128
    Source: 2001:db8::10:0:8d:0c/128
    Vlan: VLAN-144
        Downstream interface list:
        ae3.0 ae5.0

Group: ff0e::db8:0:2/128
    Source: 2001:db8::10:0:8d:0c/128
    Vlan: VLAN-144
        Downstream interface list:
        ae3.0 ae5.0

Group: ff0e::db8:0:3/128
    Source: 2001:db8::10:0:8d:0c/128
    Vlan: VLAN-144
        Downstream interface list:
        ae3.0 ae5.0
```

## Configuring VMTO

This section describes how to configure virtual machine traffic optimization (VMTO). For overview information about VMTO, see Ingress Virtual Machine Traffic Optimization for EVPN section in "Data Center Fabric Blueprint Architecture Components" on page 9.

To enable VMTO for EVPN:

- Enable VMTO on the spine:

```
set routing-instances VRF-1 protocols evpn remote-ip-host-routes
```

To verify that VMTO is working:

- Display the routing table.

```
host@spine> show route table VRF-1.inet.0 protocol evpn


VRF-1.inet.0: 45 destinations, 45 routes (45 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both
```

```
10.0.37.1/32       *[EVPN/7] 1d 21:44:43
                    >  via irb.37
10.0.37.2/32       *[EVPN/7] 1d 15:26:27
```

### VMTO — Release History

Table 27 on page 687 provides a history of all of the features in this section and their support within this reference design.

**Table 27: VMTO Release History**

| Release | Description |
|---|---|
| 19.1R2 | QFX10002-60C and QFX5120-32C switches running Junos OS Release 19.1R2 and later releases in the same release train also support all features documented in this section. |
| 18.4R2 | QFX5120-48Y switches running Junos OS Release 18.4R2 and later releases in the same release train support all features documented in this section. |
| 18.1R3-S3 | All devices in the reference design that support Junos OS Release 18.1R3-S3 and later releases in the same release train also support all features documented in this section. |

### RELATED DOCUMENTATION

*Ingress Virtual Machine Traffic Optimization*

# DHCP Relay Design and Implementation

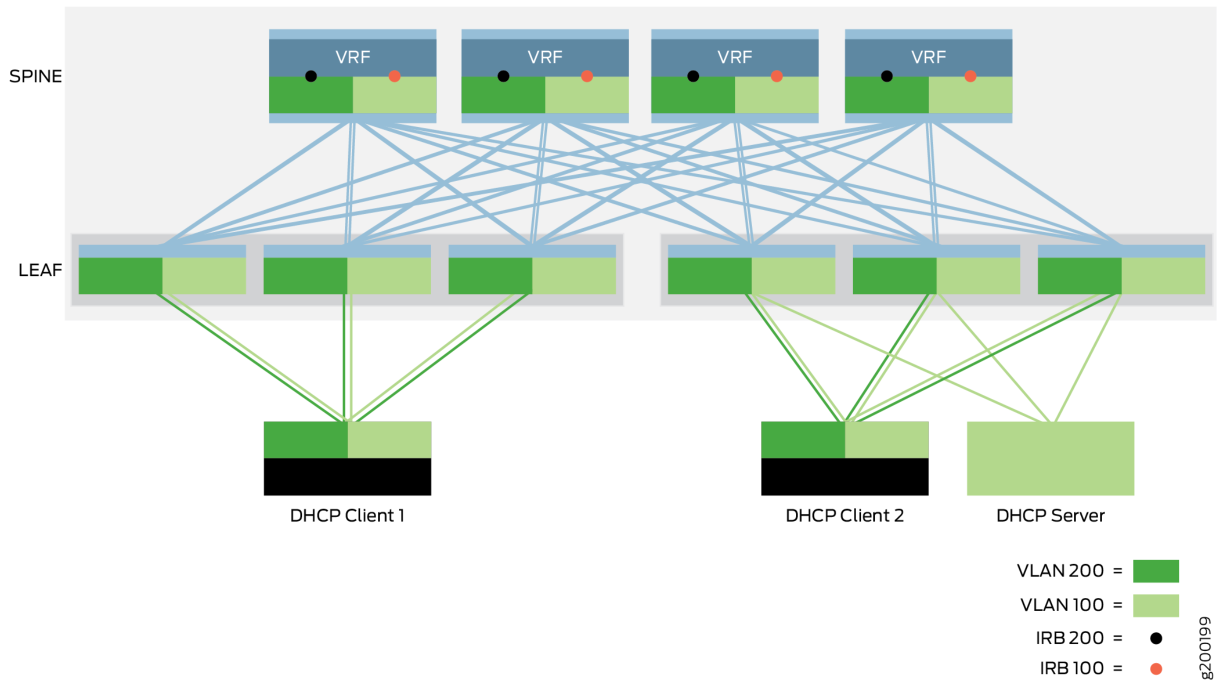
**IN THIS SECTION**

For an overview of the DHCP Relay implementation in this design, see the DHCP Relay section in "Data Center Fabric Blueprint Architecture Components" on page 9.

DHCP Relay was validated on centrally-routed bridging overlays (see "Centrally-Routed Bridging Overlay Design and Implementation" on page 142) in use cases like those described below. DHCP Relay has also been validated with edge-routed bridging overlays (see "Edge-Routed Bridging Overlay Design and Implementation" on page 206) starting in Junos OS Releases 18.4R2-S5 and 19.1R1. You can refer to Configure a DHCP Relay in EVPN-VXLAN Fabric Architecture for a network configuration example of a DHCP Relay use case with an edge-routed bridging overlay.

Figure 97 on page 688 provides a sample DHCP server and client placement in this design.

**Figure 97: DHCP Server and Client Setup**

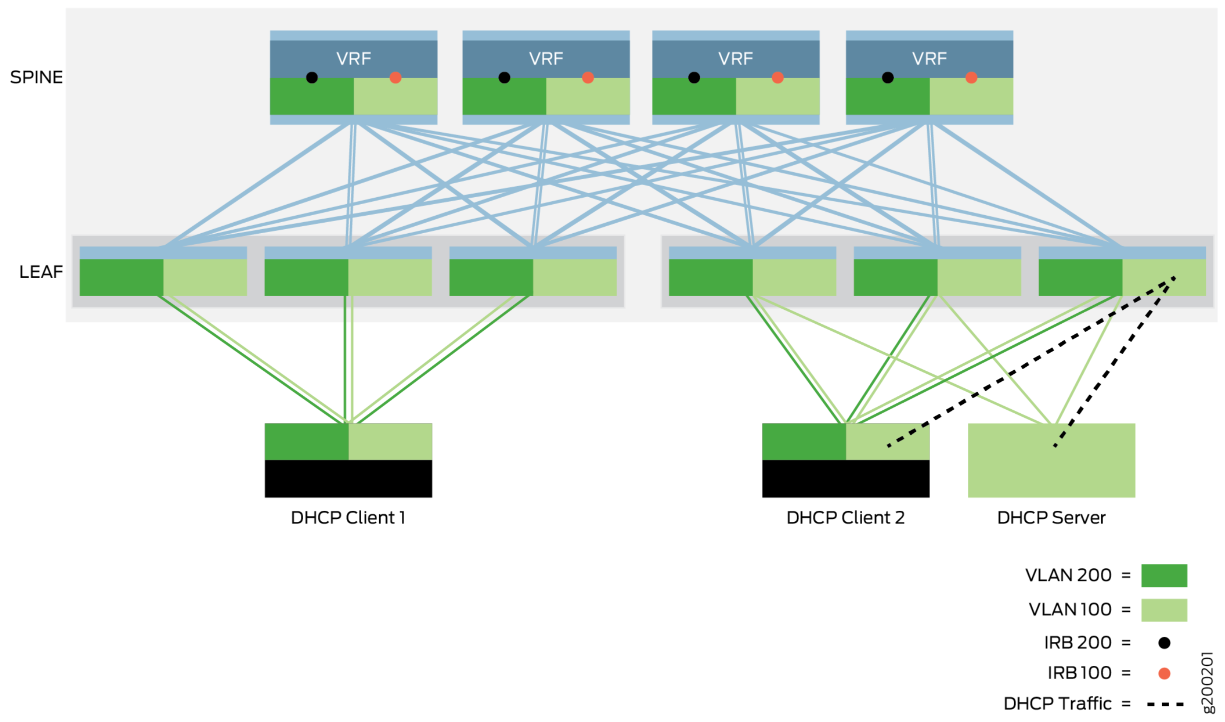


This section covers the following DHCP Relay scenarios:

## Enabling DHCP Relay: DHCP Client and Server in Same VLAN and Connected to Same Leaf Device

Figure 98 on page 689 shows how DHCP traffic might traverse the overlay network when the DHCP client and server connect to access interfaces on the same leaf device and are in the same VLAN.

Figure 98: DHCP Relay—Same Leaf Device and Same VLAN

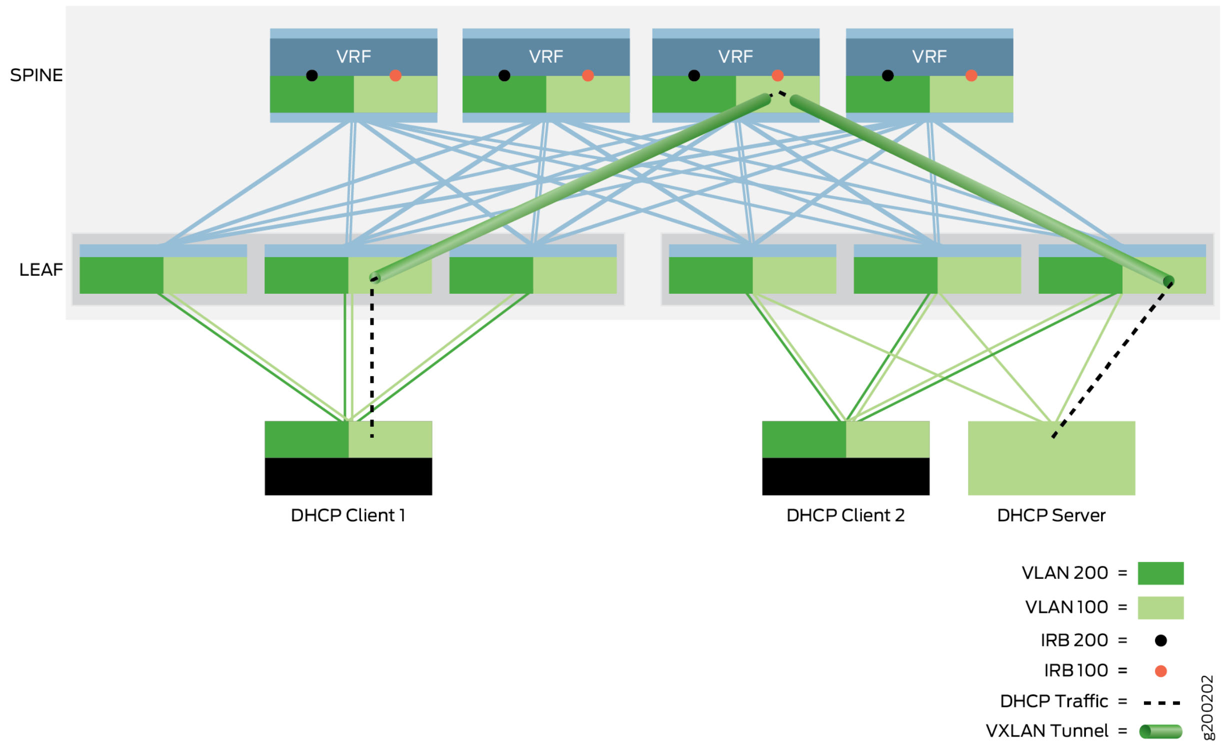


DHCP traffic is handled like any other intra-VLAN traffic in this scenario. It could also have been passed in `VNI_10000` on leaf device 10 or 11. No additional configuration of the leaf or spine devices is required to support this traffic exchange.

## Enabling DHCP Relay: DHCP Client and Server in Same VLAN and Connected to Different Leaf Devices

Figure 99 on page 690 shows how DHCP traffic might be forwarded when the DHCP client and server connect to access interfaces on non-connected leaf devices that have interfaces in the same VLAN.

**Figure 99: DHCP Relay—Different Leaf Device and Same VLAN**

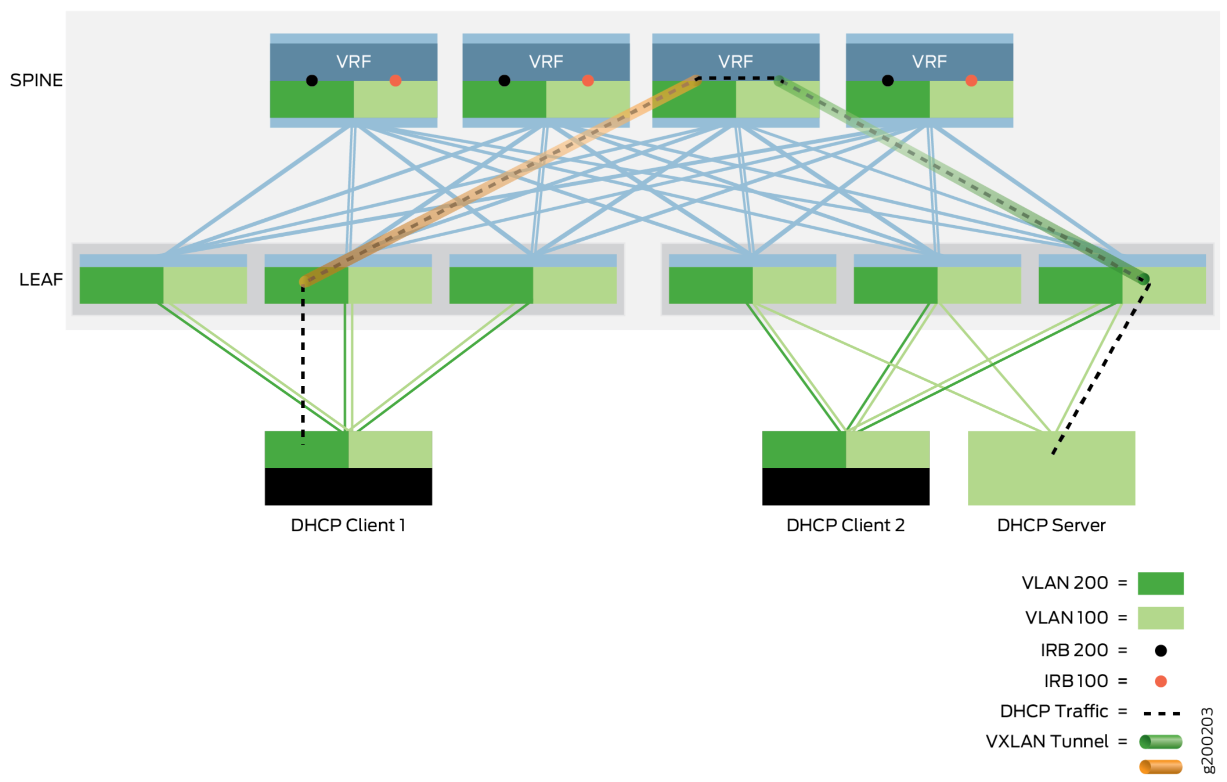


In this scenario, the DHCP traffic remains in `VNI_10000` and is passed over the topology using a VXLAN tunnel. No additional configuration of the leaf or spine devices is required to support this traffic exchange.

## Enabling DHCP Relay: DHCP Client and Server in Different VLANs

Figure 100 on page 691 shows how DHCP traffic is forwarded when the DHCP client and server connect to access interfaces in different VLANs.

**Figure 100: DHCP Relay—Different VLANs**



When the DHCP client and server are in different VNI, DHCP traffic is forwarded to a spine device. The DHCP traffic is forwarded between VNIs via the IRB interfaces on the spine device and then passed to the destination DHCP client or server.

The IRB interfaces on the spine devices must be configured to support DHCP Relay.

To prepare the network to support DHCP relay when the DHCP client and server are in different VLANs:

1. Ensure the centrally-routed bridging overlay is configured. See "Centrally-Routed Bridging Overlay Design and Implementation" on page 142.
2. Enable DHCP relay in a routing instance with the forward only option. The forward only option ensures that DHCP packets are forwarded on the switch but that no DHCP server client bindings are created.

   *Spine Device*:

   ```
   set routing-instances VRF_1 forwarding-options dhcp-relay forward-only
   ```
3. Create and activate the DHCP relay server group. The DHCP relay server group include one or more DHCP servers—individually identified by IP address—and a user-defined name for the servers.

In this reference design, one DHCP server—10.1.0.211—is assigned into a DHCP server group named *DHCP_SERVER_GROUP_1*.

*Spine Device*:

```
set routing-instances VRF_1 forwarding-options dhcp-relay server-group DHCP_SERVER_GROUP_1
10.1.0.211
set routing-instances VRF_1 forwarding-options dhcp-relay active-server-group
DHCP_SERVER_GROUP_1
```

4. Associate the server group with the IRB interfaces on the spine devices.

   In this topology, irb.100 and irb.200 relay DHCP traffic between VNI_10000 and VNI_20000.

   *Spine Devices*:

```
set routing-instances VRF_1 forwarding-options dhcp-relay group RELAY_DHCP_SERVER_GROUP_1
interface irb.100
set routing-instances VRF_1 forwarding-options dhcp-relay group RELAY_DHCP_SERVER_GROUP_1
interface irb.200
```

5. Confirm that the DHCP hosts received an IP address by reviewing the ARP table on the spine device.

```
user@spine-1> show arp no-resolve
MAC Address       Address         Interface         Flags
0e:ad:11:00:00:0d 10.1.0.214 irb.100 [.local..260] none
0e:ad:11:00:00:0e 10.1.0.215 irb.100 [.local..260] none
0e:ad:11:00:00:10 10.1.1.219 irb.200 [.local..260] none
0e:ad:11:00:00:0f 10.1.1.220 irb.200 [.local..260] none
```

## DHCP Relay — Release History

Table 28 on page 693 provides a history of all of the features in this section and their support within this reference design.

**Table 28: DHCP Relay Release History**

| Release | Description |
|---|---|
| 19.1R2 | QFX10002-60C and QFX5120-32C switches running Junos OS Release 19.1R2 and later releases in the same release train and serve as spine devices also support all features documented in this section.<br><br>We have not validated DHCP Relay between VLANs in edge-routed bridging overlays for this reference design. |
| 18.4R2 | QFX5120-48Y switches running Junos OS Release 18.4R2 and later releases in the same release train support all features documented in this section. |
| 18.1R3-S3 | QFX5110 switches running Junos OS Release 18.1R3-S3 and later releases in the same release train support all features documented in this section. |
| 17.3R3-S1 | All spine devices in the reference design that support Junos OS Release 17.3R3-S1 and later releases in the same release train also support all features documented in this section.<br><br>We have not validated DHCP Relay between VLANs in edge-routed bridging overlays for this reference design. |

### RELATED DOCUMENTATION

*Configuring DHCP and BOOTP Relay*

## Verifying Proxy ARP and ARP Suppression for the Edge-Routed Bridging Overlay

**IN THIS SECTION**

## Verifying Proxy ARP and ARP Suppression for the Edge-Routed Bridging Overlay

Proxy ARP and ARP suppression are enabled by default on all QFX Series switches that can act as leaf devices in an edge-routed bridging overlay. (For a list of these switches, see Table 2 on page 52.) As a result, there is no configuration needed to enable these features.

**NOTE**: In Junos OS releases before Release 19.1R1, you could turn off proxy ARP and ARP suppression on EX Series and QFX Series switches using the `no-arp-suppression` configuration statement. That statement is no longer supported starting in Junos OS Release 19.1R1, so proxy ARP and ARP suppression is always enabled.

To verify proxy ARP is working on supported devices and ARP suppression prevents other leaf devices from seeing the ARP requests, perform the following:

1. Select a remote end system entry to verify that proxy ARP is enabled on a supported leaf device. In this example, Leaf 10 is a QFX10002 switch.

```
user@leaf-10> show arp no-resolve expiration-time | match 10.1.4.201
02:0c:10:04:02:01 10.1.4.201      irb.500 [.local..9]      none 1157
```

2. Verify that the entry was learned from a remote leaf device. In this case, the ARP entry was learned from Leaf 4 and 5.

```
user@leaf-10> show evpn database mac-address 02:0c:10:04:02:01 extensive
Instance: default-switch

VN Identifier: 50000, MAC address:: 02:0c:10:04:02:01
  Source: 00:00:00:00:00:00:51:10:00:01, Rank: 1, Status: Active
    Remote origin: 192.168.1.4  # Leaf 4
    Remote origin: 192.168.1.5  # Leaf 5
    Timestamp: Sep 11 00:37:32 (0x59b63d3c)
    State: <Remote-To-Local-Adv-Done>
    IP address: 10.1.4.201
      Remote origin: 192.168.1.4
      Remote origin: 192.168.1.5
    IP address: 2001:db8::10:1:4:201
      Flags: <Proxy>
      Remote origin: 192.168.1.4
      Remote origin: 192.168.1.5    History db:
```

```
      Time                 Event
      Sep 11 00:37:33 2017  Active ESI unchanged (00:00:00:00:00:00:51:10:00:01)
      Sep 11 00:37:33 2017  Updating output state (change flags 0x0)
      Sep 11 00:37:33 2017  Advertisement route cannot be created (no local state present)
      Sep 11 00:37:33 2017  Updating output state (change flags 0x0)
      Sep 11 00:37:33 2017  Advertisement route cannot be created (no local source present)
      Sep 11 00:37:33 2017  IP host route cannot be created (No remote host route for non-
MPLS instance)
      Sep 11 00:37:33 2017  Updating output state (change flags 0x4000 <IP-Peer-Added>)
      Sep 11 00:37:33 2017  Creating MAC+IP advertisement route for proxy
      Sep 11 00:37:33 2017  IP host route cannot be created (No remote host route for non-
MPLS instance)
      Sep 11 00:37:33 2017  Clearing change flags <IP-Peer-Added>
```

3. Send an ARP Request and verify that the ARP reply is generated.

```
user@leaf-10> monitor traffic interface irb no-resolve
verbose output suppressed, use <detail> or <extensive> for full protocol decode
Address resolution is OFF.
Listening on irb, capture size 96 bytes



00:43:01.881508  In arp who-has 10.1.4.201 tell 10.1.4.202
00:43:01.881569 Out arp reply 10.1.4.201 is-at 02:0c:10:04:02:01
00:43:02.081404  In arp who-has 10.1.4.201 tell 10.1.4.202

## The next entry is the MAC address from the operational
mode command issued in Step 2.
00:43:02.081466 Out arp reply 10.1.4.201 is-at 02:0c:10:04:02:01
```

4. Verify that ARP is suppressed in the remote leaf device. Note: There is no ARP request connecting Leaf 4 to any other leaf in the segment.

```
user@leaf-4> monitor traffic interface irb no-resolve
verbose output suppressed, use <detail> or <extensive> for full protocol decode
Address resolution is OFF.
Listening on irb, capture size 96 bytes

^C
0 packets received by filter
0 packets dropped by kernel
```

## Proxy ARP and ARP Suppression for an Edge-Routed Bridging Overlay — Release History

Table 29 on page 696 provides a history of all of the features in this section and their support within this reference design.

**Table 29: Proxy ARP and ARP Suppression in an Edge-Routed Bridging Overlay Release History**

| Release | Description |
|---|---|
| 19.1R2 | QFX10002-60C and QFX5120-32C switches running Junos OS Release 19.1R2 and later releases in the same release train also support all features documented in this section. |
| 19.1R1 | The `no-arp-suppression` statement is deprecated starting in Junos OS Release 19.1R1. You can no longer turn off proxy ARP and ARP suppression. |
| 18.4R2 | QFX5120-48Y switches running Junos OS Release 18.4R2 and later releases in the same release train support all features documented in this section. |
| 18.1R3-S3 | All devices in the reference design that support Junos OS Release 18.1R3-S3 and later releases in the same release train also support all features documented in this section. |

RELATED DOCUMENTATION

*EVPN Proxy ARP and ARP Suppression, and Proxy NDP and NDP Suppression*

# Configuring Layer 2 Port Security Features on Ethernet-Connected End Systems

**IN THIS SECTION**

This section shows how to configure the following Layer 2 port security features. For overview information about these features, see "Layer 2 Port Security Features on Ethernet-Connected End Systems" on page 45 in "Data Center Fabric Blueprint Architecture Components" on page 9.

## Configuring Storm Control

In this sample configuration, storm control rate limits BUM traffic on server-facing aggregated Ethernet interfaces. If the amount of BUM traffic exceeds 6% of the available bandwidth on the interface, storm control drops it to prevent broadcast storms.

To enable storm control:

1. Create a storm control profile and specify the percentage of bandwidth available to BUM traffic.

   ```
   set forwarding-options storm-control-profiles STORM-CONTROL all bandwidth-percentage 6
   ```

2. Apply the storm control profile to an ingress Layer 2 interface. After you apply the profile to an interface, the interface resides in the default switch interface.

   ```
   set interfaces ae11 unit 0 family ethernet-switching storm-control STORM-CONTROL
   ```

## Verifying Storm Control

To verify storm control activity, filter system log messages related to storm control:

```
user@leaf10> show log messages | match storm
Sep 27 11:35:34 leaf1-qfx5100 l2ald[1923]: L2ALD_ST_CTL_IN_EFFECT: ae11.0: storm control in
effect on the port
```

## Configuring Port Security Using MAC Filtering

To configure MAC filtering, you create firewall filters in which you specify one or more of the supported match conditions. See *MAC Filtering, Storm Control, and Port Mirroring Support in an EVPN-VXLAN Environment* for a list of match conditions supported on QFX5110 switches and QFX10000 switches. You then apply the firewall filter to a Layer 2 interface.

To configure MAC filtering:

1. Create a firewall filter for an ingress interface.

```
set firewall family ethernet-switching filter L2-INGRESS term ARP-REQ from source-mac-address
be:ef:a2:01:00:0a/48
set firewall family ethernet-switching filter L2-INGRESS term ARP-REQ from destination-mac-
address ff:ff:ff:ff:ff:ff/48
set firewall family ethernet-switching filter L2-INGRESS term V4-BROADCAST from source-mac-
address be:ef:a2:01:00:0a/48
set firewall family ethernet-switching filter L2-INGRESS term ARP-REQ from ether-type arp
set firewall family ethernet-switching filter L2-INGRESS term ARP-REQ from user-vlan-id 10
set firewall family ethernet-switching filter L2-INGRESS term ARP-REQ then accept
set firewall family ethernet-switching filter L2-INGRESS term ARP-REQ then count ARP-REQ-CNT
set firewall family ethernet-switching filter L2-INGRESS term V4-BROADCAST from source-mac-
address be:ef:a2:01:00:0a/48
set firewall family ethernet-switching filter L2-INGRESS term V4-BROADCAST from destination-
mac-address ff:ff:ff:ff:ff:ff/48
set firewall family ethernet-switching filter L2-INGRESS term V4-BROADCAST from ether-type
ipv4
set firewall family ethernet-switching filter L2-INGRESS term V4-BROADCAST from user-vlan-id
10
set firewall family ethernet-switching filter L2-INGRESS term V4-BROADCAST then accept
set firewall family ethernet-switching filter L2-INGRESS term V4-BROADCAST then count V4-
```

```
BROADCAST-CNT-IN
set firewall family ethernet-switching filter L2-INGRESS term V6-BROADCAST from source-mac-
address be:ef:a2:01:00:0a/48
set firewall family ethernet-switching filter L2-INGRESS term V6-BROADCAST from destination-
mac-address ff:ff:ff:ff:ff:ff/48
set firewall family ethernet-switching filter L2-INGRESS term V6-BROADCAST from ether-type
ipv6
set firewall family ethernet-switching filter L2-INGRESS term V6-BROADCAST from user-vlan-id
10
set firewall family ethernet-switching filter L2-INGRESS term V6-BROADCAST then accept
set firewall family ethernet-switching filter L2-INGRESS term V6-BROADCAST then count V6-
BROADCAST-CNT-IN
set firewall family ethernet-switching filter L2-INGRESS term PKT_IN_V4 from source-mac-
address be:ef:a2:01:00:0a/48
set firewall family ethernet-switching filter L2-INGRESS term PKT_IN_V4 from destination-mac-
address 00:00:5e:00:00:04/48
set firewall family ethernet-switching filter L2-INGRESS term PKT_IN_V4 from source-port 1020
set firewall family ethernet-switching filter L2-INGRESS term PKT_IN_V4 from destination-port
1024
set firewall family ethernet-switching filter L2-INGRESS term PKT_IN_V4 from ip-source-
address 10.0.10.201/32
set firewall family ethernet-switching filter L2-INGRESS term PKT_IN_V4 from ip-destination-
address 10.0.12.201/32
set firewall family ethernet-switching filter L2-INGRESS term PKT_IN_V4 from ip-protocol tcp
set firewall family ethernet-switching filter L2-INGRESS term PKT_IN_V4 from user-vlan-id 10
set firewall family ethernet-switching filter L2-INGRESS term PKT_IN_V4 then accept
set firewall family ethernet-switching filter L2-INGRESS term PKT_IN_V4 then count V4-PKT-CNT-
IN-TCP-FLAG-0x90
set firewall family ethernet-switching filter L2-INGRESS term DEF then accept
set firewall family ethernet-switching filter L2-INGRESS term DEF then count DEF_CNT_IN
```

2. Apply the firewall filter to the ingress of an access interface / Layer 2 interface.

```
set interfaces ae11 unit 0 family ethernet-switching filter input L2-INGRESS
```

3. Create a firewall filter for an egress interface.

```
set firewall family ethernet-switching filter L2-EGRESS term ARP-REP from source-mac-address
00:00:5e:00:00:04/48
set firewall family ethernet-switching filter L2-EGRESS term ARP-REP from destination-mac-
address be:ef:a2:01:00:0a/48
set firewall family ethernet-switching filter L2-EGRESS term ARP-REP from ether-type arp
set firewall family ethernet-switching filter L2-EGRESS term ARP-REP from user-vlan-id 10
set firewall family ethernet-switching filter L2-EGRESS term ARP-REP then accept
set firewall family ethernet-switching filter L2-EGRESS term ARP-REP then count ARP-REP-CNT
set firewall family ethernet-switching filter L2-EGRESS term V4-BROADCAST from source-mac-
address be:ef:a4:03:00:0c/48
set firewall family ethernet-switching filter L2-EGRESS term V4-BROADCAST from destination-
mac-address ff:ff:ff:ff:ff:ff/48
set firewall family ethernet-switching filter L2-EGRESS term V4-BROADCAST from ether-type ipv4
set firewall family ethernet-switching filter L2-EGRESS term V4-BROADCAST from user-vlan-id 12
set firewall family ethernet-switching filter L2-EGRESS term V4-BROADCAST then accept
set firewall family ethernet-switching filter L2-EGRESS term V4-BROADCAST then count V4-
BROADCAST-CNT-OUT
set firewall family ethernet-switching filter L2-EGRESS term V6-BROADCAST from source-mac-
address be:ef:a4:03:00:0c/48
set firewall family ethernet-switching filter L2-EGRESS term V6-BROADCAST from destination-
mac-address ff:ff:ff:ff:ff:ff/48
set firewall family ethernet-switching filter L2-EGRESS term V6-BROADCAST from ether-type ipv6
set firewall family ethernet-switching filter L2-EGRESS term V6-BROADCAST from user-vlan-id 12
set firewall family ethernet-switching filter L2-EGRESS term V6-BROADCAST then accept
set firewall family ethernet-switching filter L2-EGRESS term V6-BROADCAST then count V6-
BROADCAST-CNT-OUT
set firewall family ethernet-switching filter L2-EGRESS term DEF then accept
set firewall family ethernet-switching filter L2-EGRESS term DEF then count DEF_CNT_OUT
```

4. Apply the firewall filter to the egress interface.

```
set interfaces ae11 unit 0 family ethernet-switching filter output L2-EGRESS
```

## Verifying MAC Filtering

1. Verify MAC filtering on the ingress interface.

```
user@leaf10> show firewall filter L2-INGRESS
Filter: L2-INGRESS
Counters:
Name                                                Bytes              Packets
ARP-REQ-CNT                                           640                   10
DEF_CNT_IN                                       44038137                79032
V4-BROADCAST-CNT-IN                                     0                    0
```

```
V4-PKT-CNT-IN-TCP                                7418880               57960
V6-BROADCAST-CNT-IN                              5370880               41960
```

2. Verify MAC filtering on the egress interface.

```
user@leaf10> show firewall filter L2-EGRESS
Filter: L2-EGRESS
Counters:
Name                                               Bytes             Packets
ARP-REP-CNT                                           68                   1
DEF_CNT_OUT                                     17365964              146535
V4-BROADCAST-CNT-OUT                             4171264               32588
V6-BROADCAST-CNT-OUT                             3147264               24588
```

## Configuring Analyzer-Based Port Mirroring

This section shows how to mirror ingress traffic on an underlay interface to another physical port.

The source and destination ports for mirrored traffic are on the same leaf or same spine.

1. Configure an analyzer to mirror ingress traffic on interface ae1.0.

```
set forwarding-options analyzer A1 input ingress interface ae1.0
```

2. Configure the destination interface for the mirrored packets.

```
set forwarding-options analyzer A1 output interface et-0/0/71.0
```

3. Configure the interface that connects to another switch (the uplink interface) to trunk mode and associate it with the appropriate VLAN.

```
set interfaces et-0/0/71 unit 0 family ethernet-switching interface-mode trunk
set interfaces et-0/0/71 unit 0 family ethernet-switching vlan members all
```

## Verifying Port Mirroring

- To verify port mirroring:

```
host> show forwarding-options analyze
r
  Analyzer name                      : A1
  Mirror rate                        : 1
  Maximum packet length              : 0
  State                              : up
  ingress monitored interfaces       : ae1.0
  Output interface                   : et-0/0/71.0
```

## Layer 2 Port Security Features — Release History

Table 30 on page 702 provides a history of all of the features in this section and their support within this reference design.

**Table 30: Layer 2 Port Security Release History**

| Release | Description |
|---|---|
| 19.1R2 | • QFX5120-32C switches running Junos OS Release 19.1R2 and later releases in the same release train support MAC filtering, storm control, and port mirroring and analyzing.<br>• QFX10002-60C switches running Junos OS Release 19.1R2 and later releases in the same release train support MAC filtering. These switches do not support storm control, and port mirroring and analyzing. |
| 18.4R2 | QFX5120-48Y switches running Junos OS Release 18.4R2 and later releases in the same release train support all features documented in this section. |
| 18.1R3-S3 | All devices in the reference design that support Junos OS Release 18.1R3-S3 and later releases in the same release train also support all features documented in this section. |

## RELATED DOCUMENTATION

*MAC Filtering, Storm Control, and Port Mirroring Support in an EVPN-VXLAN Environment*

Remote Port Mirroring for EVPN-VXLAN Fabrics