CONFIGURATION GUIDE

Extreme SLX-OS IP Fabrics Configuration Guide, 18s.1.01

Supporting SLX-OS 18s.1.01

9035268
November 2018

Legal Notice

Extreme Networks, Inc. reserves the right to make changes in specifications and other information contained in this document and its website without prior notice. The reader should in all cases consult representatives of Extreme Networks to determine whether any such changes have been made.

The hardware, firmware, software or any specifications described or referred to in this document are subject to change without notice.

Trademarks

Extreme Networks and the Extreme Networks logo are trademarks or registered trademarks of Extreme Networks, Inc. in the United States and/or other countries.

All other names (including any product names) mentioned in this document are the property of their respective owners and may be trademarks or registered trademarks of their respective companies/owners.

For additional information on Extreme Networks trademarks, please see: www.extremenetworks.com/company/legal/trademarks

Software Licensing

Some software files have been licensed under certain open source or third-party licenses. End-user license agreements and open source declarations can be found at: www.extremenetworks.com/support/policies/software-licensing

Support

For product support, phone the Global Technical Assistance Center (GTAC) at 1-800-998-2408 (toll-free in U.S. and Canada) or +1-408-579-2826. For the support phone number in other countries, visit: http://www.extremenetworks.com/support/contact/

For product documentation online, visit: https://www.extremenetworks.com/documentation/

Contents

Preface

This section discusses the conventions used in this guide, ways to provide feedback, additional help, and other Extreme Networks® publications.

Conventions

This section discusses the conventions used in this guide.

Notes, cautions, and warnings

Notes, cautions, and warning statements may be used in this document. They are listed in the order of increasing severity of potential hazards.

NOTE
A Note provides a tip, guidance, or advice, emphasizes important information, or provides a reference to related information.

ATTENTION
An Attention statement indicates a stronger note, for example, to alert you when traffic might be interrupted or the device might reboot.

CAUTION
A Caution statement alerts you to situations that can be potentially hazardous to you or cause damage to hardware, firmware, software, or data.

DANGER
A Danger statement indicates conditions or situations that can be potentially lethal or extremely hazardous to you. Safety labels are also attached directly to products to warn of these conditions or situations.

Text formatting conventions

Text formatting conventions such as boldface, italic, or Courier font may be used to highlight specific words or phrases.

Format	Description
bold text	Identifies command names.
	Identifies keywords and operands.
	Identifies the names of GUI elements.
	Identifies text to enter in the GUI.
italic text	Identifies emphasis.
	Identifies variables.
	Identifies document titles.

Format	Description
`Courier font`	Identifies CLI output. Identifies command syntax examples.

Command syntax conventions

Bold and italic text identify command syntax components. Delimiters and operators define groupings of parameters and their logical relationships.

Convention	Description
bold text	Identifies command names, keywords, and command options.
italic text	Identifies a variable.
[ ]	Syntax components displayed within square brackets are optional. Default responses to system prompts are enclosed in square brackets.
{ **x** | **y** | **z** }	A choice of required parameters is enclosed in curly brackets separated by vertical bars. You must select one of the options.
x | **y**	A vertical bar separates mutually exclusive elements.
< >	Nonprinting characters, for example, passwords, are enclosed in angle brackets.
...	Repeat the previous element, for example, *member*[*member*...].
\	Indicates a "soft" line break in command examples. If a backslash separates two lines of a command input, enter the entire command at the prompt without the backslash.

Documentation and Training

To find Extreme Networks product guides, visit our documentation pages at:

Current Product Documentation	www.extremenetworks.com/documentation/
Archived Documentation (for earlier versions and legacy products)	www.extremenetworks.com/support/documentation-archives/
Release Notes	www.extremenetworks.com/support/release-notes
Hardware/Software Compatibility Matrices	https://www.extremenetworks.com/support/compatibility-matrices/
White papers, data sheets, case studies, and other product resources	https://www.extremenetworks.com/resources/

Open Source Declarations

Some software files have been licensed under certain open source licenses. More information is available at: www.extremenetworks.com/support/policies/open-source-declaration/.

Training

Extreme Networks offers product training courses, both online and in person, as well as specialized certifications. For more information, visit www.extremenetworks.com/education/.

Getting Help

If you require assistance, contact Extreme Networks using one of the following methods:

- GTAC (Global Technical Assistance Center) for Immediate Support
  - **Phone:** 1-800-998-2408 (toll-free in U.S. and Canada) or +1 408-579-2826. For the support phone number in your country, visit: www.extremenetworks.com/support/contact
  - **Email:** support@extremenetworks.com. To expedite your message, enter the product name or model number in the subject line.
- Extreme Portal — Search the GTAC knowledge base, manage support cases and service contracts, download software, and obtain product licensing, training, and certifications.
- The Hub — A forum for Extreme Networks customers to connect with one another, answer questions, and share ideas and feedback. This community is monitored by Extreme Networks employees, but is not intended to replace specific guidance from GTAC.

Before contacting Extreme Networks for technical support, have the following information ready:

- Your Extreme Networks service contract number and/or serial numbers for all involved Extreme Networks products
- A description of the failure
- A description of any action(s) already taken to resolve the problem
- A description of your network environment (such as layout, cable type, other relevant environmental information)
- Network load at the time of trouble (if known)
- The device history (for example, if you have returned the device before, or if this is a recurring problem)
- Any related RMA (Return Material Authorization) numbers

Subscribing to Service Notifications

You can subscribe to email notifications for product and software release announcements, Vulnerability Notices, and Service Notifications.

1. Go to www.extremenetworks.com/support/service-notification-form.
2. Complete the form with your information (all fields are required).
3. Select the products for which you would like to receive notifications.

   NOTE
   You can modify your product selections or unsubscribe at any time.

4. Click **Submit**.

Providing Feedback to Us

Quality is our first concern at Extreme Networks, and we have made every effort to ensure the accuracy and completeness of this document. We are always striving to improve our documentation and help you work better, so we want to hear from you! We welcome all feedback but especially want to know about:

- Content errors or confusing or conflicting information.
- Ideas for improvements to our documentation so you can find the information you need faster.

- Broken links or usability issues.

If you would like to provide feedback to the Extreme Networks Information Development team, you can do so in two ways:

- Use our short online feedback form at https://www.extremenetworks.com/documentation-feedback/.
- Email us at documentation@extremenetworks.com.

Please provide the publication title, part number, and as much detail as possible, including the topic heading and page number if applicable, as well as your suggestions for improvement.

About This Document

Supported hardware and software

This guide is specific to support for IP Fabrics. In those instances in which procedures or parts of procedures documented here apply to some switches but not to others, this guide identifies exactly which switches are supported for certain features and which are not.

Although many different software and hardware configurations are tested and supported by Extreme Networks, Inc. for SLX-OS, documenting all possible configurations and scenarios is beyond the scope of this document.

The following hardware platforms are supported by this release of SLX-OS:

- ExtremeSwitching SLX 9140
- ExtremeSwitching SLX 9240

What's new in this document

On October 30, 2017, Extreme Networks, Inc. acquired the data center networking business from Brocade Communications Systems, Inc. This document has been updated to remove or replace references to Brocade Communications, Inc. with Extreme Networks., Inc., as appropriate.

The following table includes descriptions of new information added to this guide for the SLX OS 18s.1.01 software release.

TABLE 1 Summary of enhancements in SLX OS release 18s.1.01

Feature	Description	Described in
Embedded Fabric Automation	Embedded Fabric Automation (EFA) supports the automated configuration of an IP Fabric, greatly simplifying the time and effort that would otherwise be required.	Embedded Fabric Automation on page 89

IP Fabrics

Overview of IP Fabrics

Data center networking architectures have evolved with the changing requirements of the modern data center and cloud environments.

The traffic patterns in data center networks are changing rapidly from north-south to east-west. Cloud applications are often multitiered and hosted at different endpoints connected to the network. The communication between these application tiers is a major contributor to the overall traffic in a data center.

These traffic patterns are the primary reasons that data center networks need to evolve into scale-out architectures. Scale-out architectures are built to maximize the throughput for east-west traffic. In addition to providing high east-west throughput, scale-out architectures provide a mechanism to add capacity to the network horizontally, without reducing the provisioned capacity between the existing endpoints. The de-facto industry standard for implementing scale-out architectures is using Clos topologies. These topologies include the 3-stage folded Clos (or leaf-spine topology) and the optimized 5-stage folded Clos. These topologies are described in Physical topologies and scale on page 14.

With Extreme IP Fabrics, support is provided for a Layer 3 Clos deployment for data center sites, where all the links in the Clos topology are Layer 3 links. An Extreme IP Fabric includes the networking architecture, the protocols used to build the network, turnkey automation features used to provision, manage, and monitor the networking infrastructure and the hardware differentiation with Extreme VDX switches.

Because the infrastructure is built on IP, advantages such as loop-free communication using industry-standard routing protocols, ECMP, very high solution scale, and standards-based interoperablility are leveraged.

These are some of the key benefits of deploying a data center site with Extreme IP Fabrics:

- Highly scalable infrastructure: Because the Clos topology is built upon IP protocols, the scale of the infrastructure is very high. The port and rack scales are documented with descriptions of the Extreme IP Fabrics deployment topologies.
- Standards-based and interoperable protocols: Extreme IP Fabrics is built upon industry-standard protocols such as the Border Gateway Protocol (BGP) and Open Shortest Path First (OSPF). These protocols are well understood and provide a solid foundation for a highly scalable solution. In addition, industry-standard overlay control and data plane protocols such as BGP EVPN and Virtual Extensible Local Area Network (VXLAN) are used to extend the Layer 2 domain and extend tenancy domains by enabling Layer 2 communications and virtual machine (VM) mobility.
- Active-active multichassis trunk (MCT) cluster pairs: By supporting MCT cluster pairs on leaf switches, dual-homing of the networking endpoints is supported. This provides higher redundancy. Also, because the links are active-active, the MCT cluster provides higher throughput to the endpoints; these pairs are supported for all interface speeds, and up to 32 links can participate in a pair.
- Layer 2 extensions: In order to enable Layer 2 domain extension across the Layer 3 infrastructure, VXLAN encapsulation is leveraged. The use of VXLAN provides a very large number of Layer 2 domains to support large-scale multitenancy over the infrastructure. In addition, Extreme BGP EVPN network virtualization provides the control plane for the VXLAN, enabling automatic VTEP discovery and control-plane learning of remote MAC addresses and MAC-IP bindings, thus eliminating

broadcast, unknown unicast, and multicast (BUM) traffic. (For details, refer to Controllerless Network Virtualization with BGP EVPN on page 21.)

- Multitenancy at Layers 2 and 3: Extreme IP Fabrics provides multitenancy at Layers 2 and 3, enabling traffic isolation and segmentation across the fabric. Layer 2 multitenancy allows an extended range of up to 8000 Layer 2 domains to exist at each ToR switch, while isolating overlapping 802.1q tenant networks into separate Layer 2 domains. Layer 3 multitenancy with VRFs, multi-VRF routing protocols, and BGP EVPN allow large-scale Layer 3 multitenancy. Specifically, Extreme BGP EVPN Network Virtualization leverages BGP EVPN to provide a control plane for MAC address learning and VRF routing for tenant prefixes and host routes, which reduces BUM traffic and optimizes the traffic patterns in the network.
- Support for unnumbered interfaces: Using Extreme Network OS support for IP unnumbered interfaces, only one IP address per switch is required to configure the routing protocol peering. This significantly reduces the planning and use of IP addresses and simplifies operations.
- Turnkey automation: Extreme automated provisioning dramatically reduces the deployment time of network devices and network virtualization. Prepackaged, server-based automation scripts provision Extreme IP Fabrics devices for service with minimal effort.
- Programmable automation: Extreme server-based automation provides support for common industry automation tools such as Python, Puppet, and YANG model-based REST and NETCONF APIs. Prepackaged PyNOS scripting library and editable automation scripts execute predefined provisioning tasks, while allowing customization for addressing unique requirements to meet technical or business objectives when the enterprise is ready.
- Ecosystem integration: Extreme IP Fabrics integrates with leading industry solutions and products such as VMware vSphere, NSX, and vRealize. Cloud orchestration and control are provided through OpenStack and OpenDaylight-based Extreme SDN Controller support.

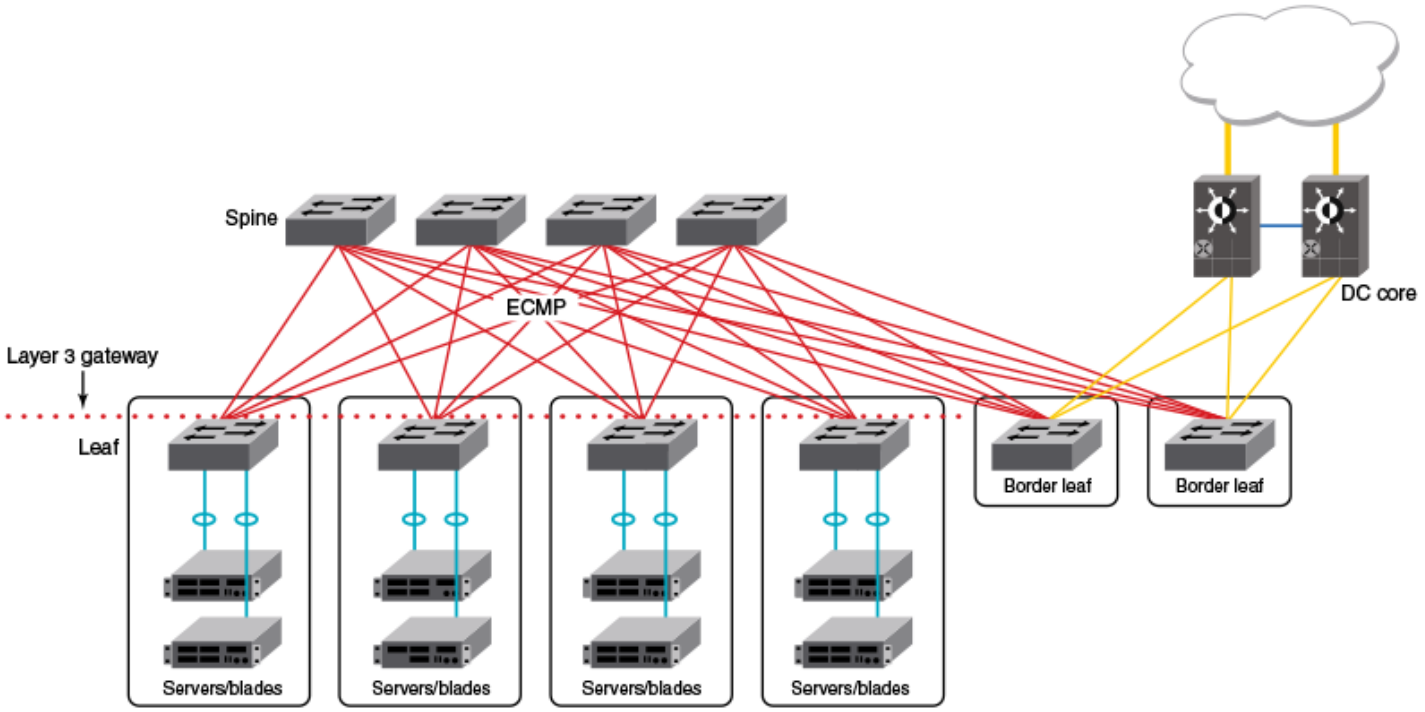
NOTE
For supported features, refer to the Extreme IP Fabrics Data Sheet.

Physical topologies and scale

The basic IP Fabrics physical topologies represent the underlay network and offer different scaling factors.

The following topology illustrates a 3-stage folded Clos topology with leaf nodes and spines, with connectivity to the DC core through border leaf nodes. These nodes provide external connectivity to the data center fabric and also provide connectivity to network services such as firewalls and load balancers.

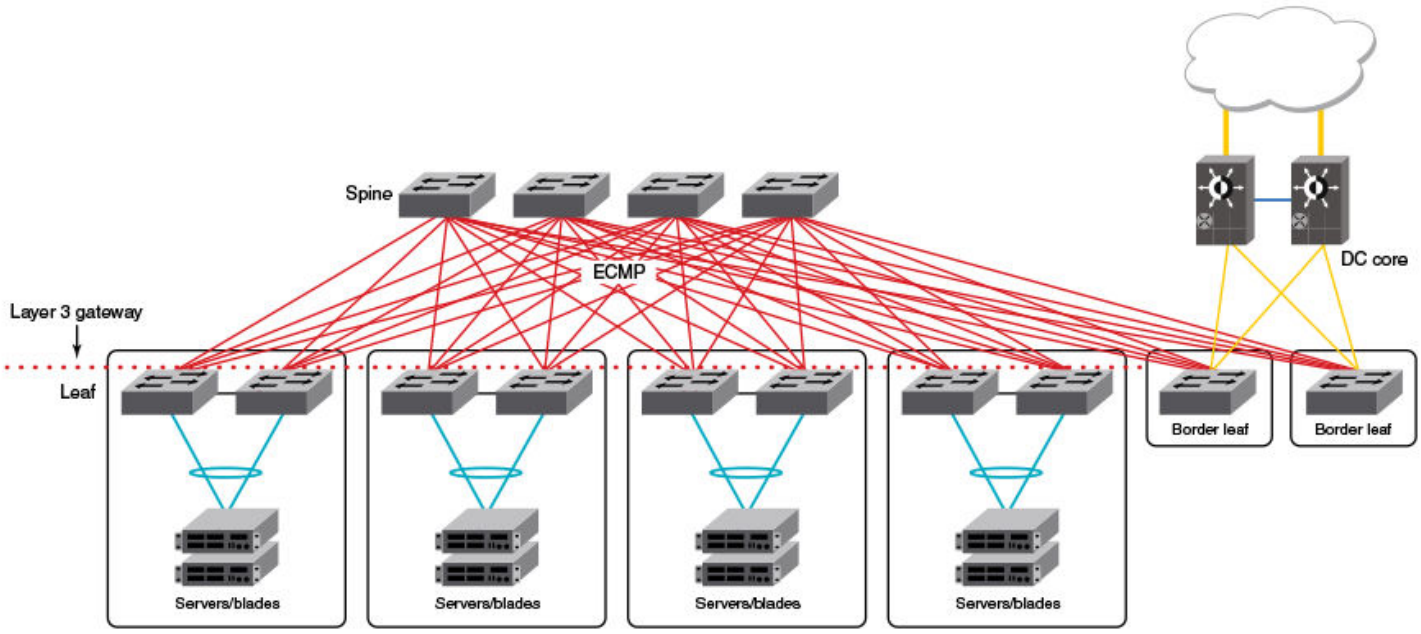
FIGURE 1 A 3-stage folded Clos topology



The following topology is similar to the previous topology, but with support for optional high availability (HA) provided through redundant servers or blades.

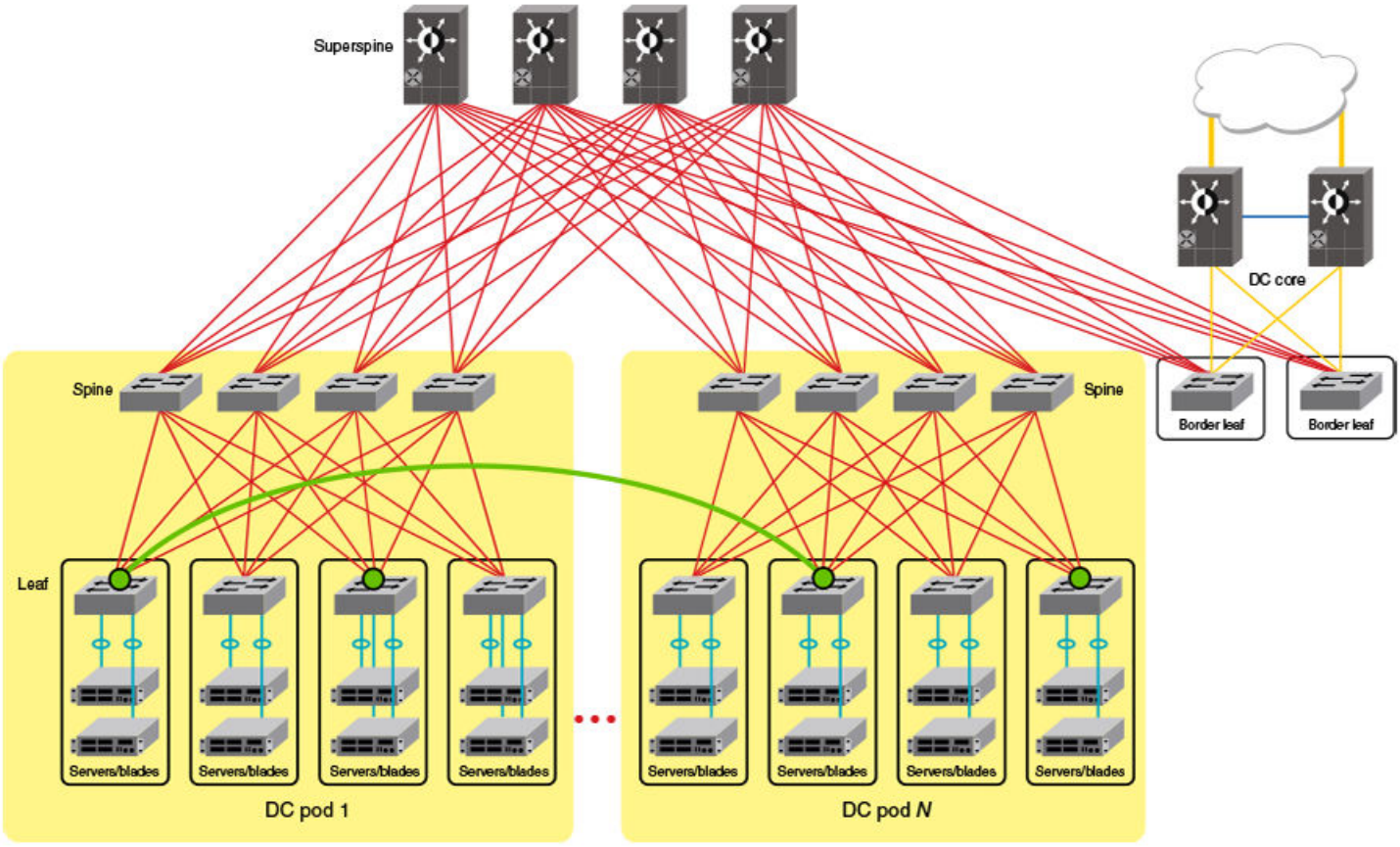
Redundancy is provided by means of MCT cluster pairs.

FIGURE 2 A 3-stage folded Clos topology with paired leaf nodes for redundant servers or blades



The following topology uses superspine nodes to interconnect multiple data center pods. Here the border leaf nodes connect directly to the superspine nodes.

FIGURE 3 Optimized 5-stage folded Clos topology



NOTE

For design considerations and scaling details, refer to the "Extreme Data Center Fabric Architectures" white paper. For details of BGP underlay topologies, refer to Controllerless Network Virtualization with BGP EVPN on page 21.

IP unnumbered interface

In a typical Layer 3 Clos network, routers are directly connected and require the assignment of IP addresses to each pair of routers, typically from the same subnet.

With large numbers of routers and redundant Layer 3 interfaces, a significant number of IP addresses are consumed just to configure the network itself. Using /31 masks reduces the consumption of addresses, but two IP addresses are still consumed per interface. Using unnumbered interfaces greatly reduces the number of IP addresses consumed in the configuration of the network.

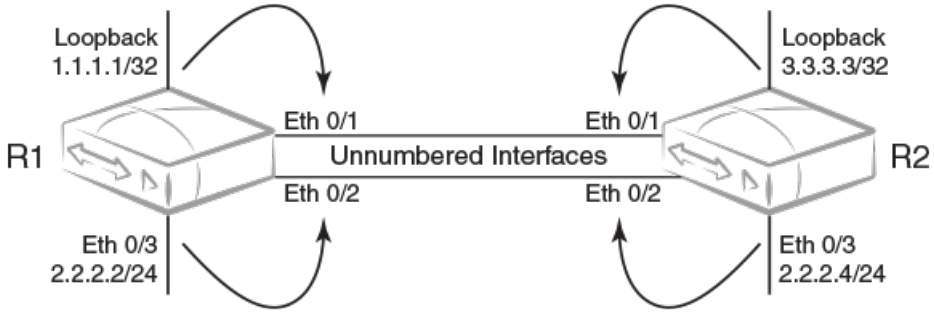
This feature borrows an IP address from another Layer 3 interface already configured on the router. This address is used as a source address in the Layer 3 packets that are sent out the unnumbered interface. The interface from which the IP address is borrowed is called the donor interface. The following points highlight some of the key aspects of the use and functionality of this feature:

- Only physical interfaces can be configured as unnumbered interfaces. Unnumbered interfaces are not supported for virtual Ethenet (VE) or switched virtual interface (SVI) interfaces.
- The donor interface can be any other Layer 3 interface (Ethernet/VE/loopback).

- Unnumbered interface support is provided for IPv4 address family. Consequently, only an IPv4 address can be borrowed for an unnumbered interface.
- Once the interface is configured as an unnumbered interface, it is treated as a point-to-point interface and there can be only one remote neighbor attached to the interface.
- Because no network subnet is associated with the unnumbered interface, ARP is not supported on an unnumbered interface.
- Unnumbered interfaces are not supported in the management VRF, because routing protocols (OSPF/BGP) are not enabled in the management VRF.

The following diagram illustrates the overall concept of unnumbered interfaces and donor interfaces.

FIGURE 4 IP unnumbered and donor interfaces



- Interfaces Ethernet 0/1 and Ethernet 0/2 on R1 and R2 are unnumbered interfaces.
- Interfaces Ethernet 0/1 on R1 and R2 borrow IPv4 addresses from a loopback interface.
- Interfaces Ethernet 0/2 on R1 and R2 borrow IPv4 addresses from physical interface Ethernet 0/3.
- The end points of the unnumbered interface may or may not be in the same subnet.

The following functionalities are required to support Layer 3 over unnumbered interfaces:

- Neighbor discovery
- Host routes to reach neighbors over unnumbered interfaces
- Routes using unnumbered interfaces as next-hops

NOTE
For a simple three-stage IP Fabric, IP unnumbered interfaces can be used. For a five-stage IP Fabric, numbered interfaces are highly recommended, although IP unnumbered interfaces can be used in five-stage IP Fabric deployments if third-party devices are not included in the design. Brocade does not recommend using a mix of unnumbered and numbered interfaces within an IP Fabric.

MTUs on physical and LAG ports

You can configure the size, in bytes, of the maximum transmission unit (MTU) for a Layer 2 packet on a physical or LAG (port-channel) interface bound to one more VLANs. This feature is supported only on the SLX 9140.

IP Fabrics deployments introduce encapsulations on the IP packet, and therefore users must be aware of the IP MTU size in the fabric so that packets are not dropped as a result of MTU checks.

There are no restrictions on the number of Layer 2 MTU profiles within a broadcast domain. Each port, whether physical or LAG, can have a different MTU value, which is set by means of the **mtu** command in interface configuration mode. For physical ports, the MTU is configured on the router port's internal VLAN ID (IVID), which is allocated when the router port is configured. For LAG ports, the Layer 2 MTU is configured on the VLAN to which a virtual Ethernet (VE) interface is bound.

You can figure a nondefault Layer 2 MTU value globally, by means of the global **mtu** command. The configuration on the interface takes precedence.

Note the following examples.

To configure a nondefault Layer 2 MTU globally:

```
device# configure terminal
device(config)# mtu 2000
```

To configure a nondefault Layer 2 MTU on an Ethernet interface:

```
device# configure terminal
device(config)# interface ethernet 0/1
device(conf-if-eth-0/1)# mtu 2000
```

To confirm the running configuration in an example switchport context:

```
device# show running-config interface ethernet 0/1
interface Ethernet 0/31
 speed 40000
 mtu 2000
 switchport
 switchport mode access
 switchport access vlan 1
 no shutdown
```

To configure a nondefault Layer 2 MTU on a port-channel interface:

```
device# configure terminal
device(config)# interface port-channel 10
device(config-Port-channel-10)# mtu 2000
```

To confirm the running configuration in an example switchport context:

```
device# show running-config interface Port-channel 10
interface Port-channel 10
 mtu 2000
 switchport
 switchport mode trunk
 switchport trunk allowed vlan all
 switchport trunk tag native-vlan
```

To revert to the default Layer 2 MTU for the above example:

```
device# configure terminal
device(config)# interface port-channel 10
device(config-Port-channel-10)# no mtu
```

IP-based management cluster

IP-based management cluster builds logical clusters of switches over IP, to manage cluster-wide configurations and distribution services.

This feature supports two-node clustering, to support a two-node leaf architecture for IP Fabrics, and it is required for EVPN deployments. For further information and a configuration example, refer to the "IP-Based Management Cluster" chapter in the *Extreme SLX-OS Layer 2 Switching Configuration Guide*.

In two-node cluster deployments, the overlay gateway configuration is distributed to both nodes in the pair. The CLI configuration must be done from only one node. One of the cluster nodes is elected as the principal node, and the overlay configurations must also be done from the principal node. However, underlay configurations (for example, VLANs, BDs, Layer 3 configurations) are not distributed, and so they must be executed on individual nodes.

Controllerless Network Virtualization with BGP EVPN

Overview of controllerless network virtualization with BGP EVPN

Layer 2 extension mechanisms using VXLAN rely on “flood and learn” mechanisms. These mechanisms are very inefficient, making MAC address convergence longer and resulting in unnecessary flooding.

Also, in a data center environment with VXLAN-based Layer 2 extension mechanisms, a Layer 2 domain and an associated subnet might exist across multiple racks and even across all racks in a data center site. With traditional underlay routing mechanisms, routed traffic destined to a VM or a host belonging to the subnet follows an inefficient path in the network, because the network infrastructure is aware only of the existence of the distributed Layer 3 subnet, but is not aware of the exact location of the hosts behind a leaf switch.

With Extreme BGP EVPN network virtualization, network virtualization is achieved through creation of a VXLAN-based overlay network. Extreme BGP EVPN network virtualization leverages BGP EVPN to provide a control plane for the virtual overlay network. BGP EVPN enables control-plane learning for end hosts behind remote VXLAN tunnel endpoints (VTEPs). This learning includes reachability for Layer 2 MAC addresses and Layer 3 host routes.

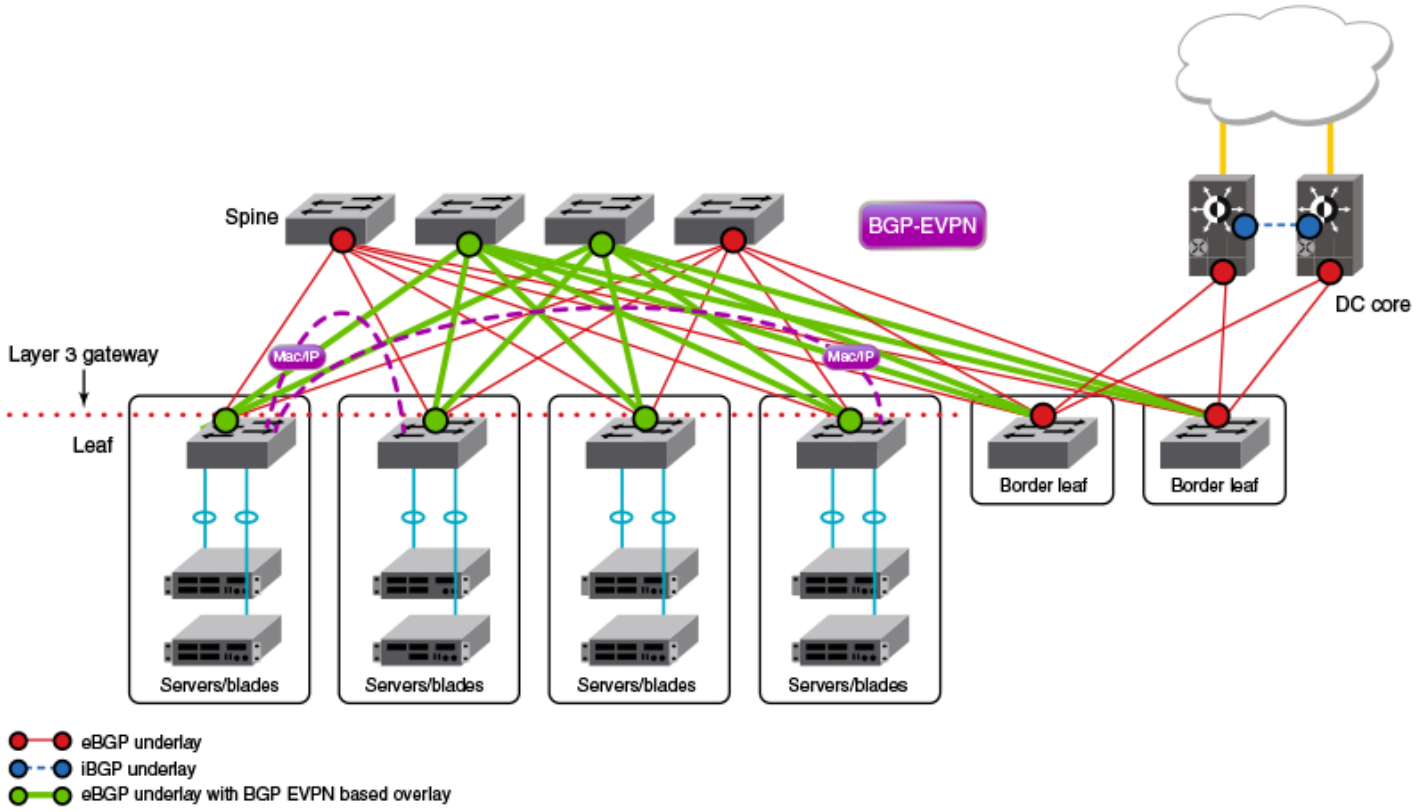
Some key features and benefits of Extreme BGP EVPN network virtualization are summarized as follows:

- Active-active MCT cluster pairs: node pairs for multiswitch port channel for dual homing of network endpoints are supported at the leaf. Both the switches in the pair participate in the BGP EVPN operations and are capable of actively forwarding traffic.

- Static anycast gateway: With static anycast gateway technology, each leaf is assigned the same default gateway IP and MAC addresses for all the connected subnets. This ensures that local traffic is terminated and routed at Layer 3 at the leaf. This also eliminates any suboptimal inefficiencies found with centralized gateways. All leaves are simultaneously active forwarders for all default traffic for which they are enabled. Also, because the static anycast gateway does not rely on any control plane protocol, it can scale to large deployments.
- Efficient VXLAN routing: With the existence of active-active leaf and the static anycast gateway, all traffic is routed and switched at the leaf. Routed traffic from the network endpoints is terminated in the leaf and is then encapsulated in VXLAN header to be sent to the remote site. Similarly, traffic from the remote leaf node is VXLAN-encapsulated and needs to be decapsulated and routed to the destination. This VXLAN routing operation into and out of the tunnel on the leaf switches is enabled in the Extreme platform ASICs. VXLAN routing performed in a single pass is more efficient than with competitive ASICs.
- Data plane IP and MAC learning: With IP host routes and MAC addresses learned from the data plane and advertised with BGP EVPN, the leaf switches are aware of the reachability information for the hosts in the network. Any traffic destined to the hosts takes the most efficient route in the network.
- Layer 2 and Layer 3 multitenancy: BGP EVPN provides control plane for VRF routing as well as for Layer 2 VXLAN extension. BGP EVPN enables a multitenant infrastructure and extends it across the data center site to enable traffic isolation between the Layer 2 and Layer 3 domains, while providing efficient routing and switching between the tenant endpoints.
- Dynamic tunnel discovery: With BGP EVPN, the remote VTEPs are automatically discovered. The resulting VXLAN tunnels are also automatically created. This significantly reduces Operational Expense (OpEx) and eliminates errors in configuration.
- ARP/ND suppression: As the BGP EVPN EVI leaves discover remote IP and MAC addresses, they use this information to populate their local ARP tables. Using these entries, the leaf switches respond to any local ARP queries. This eliminates the need for flooding ARP requests in the network infrastructure.
- Conversational ARP/ND learning: Conversational ARP/ND reduces the number of cached ARP/ND entries by programming only active flows into the forwarding plane. This helps to optimize utilization of hardware resources. In many scenarios, there are software requirements for ARP and ND entries beyond the hardware capacity. Conversational ARP/ND limits storage-in-hardware to active ARP/ND entries; aged-out entries are deleted automatically.
- VM mobility support: If a VM moves behind a leaf switch, with data plane learning, the leaf switch discovers the VM and learns its addressing information. It advertises the reachability to its peers, and when the peers receive the updated information for the reachability of the VM, they update their forwarding tables accordingly. BGP EVPN-assisted VM mobility leads to faster convergence in the network.
- Simpler deployment: With multi-VRF routing protocols, one routing protocol session is required per VRF. With BGP EVPN, VRF routing and MAC address reachability information is propagated over the same BGP sessions as the underlay, with the addition of the L2VPN EVPN address family. This significantly reduces OpEx and eliminates errors in configuration.
- Open standards and interoperability: BGP EVPN is based on the open standard protocol and is interoperable with implementations from other vendors. This allows the BGP EVPN-based solution to fit seamlessly in a multivendor environment

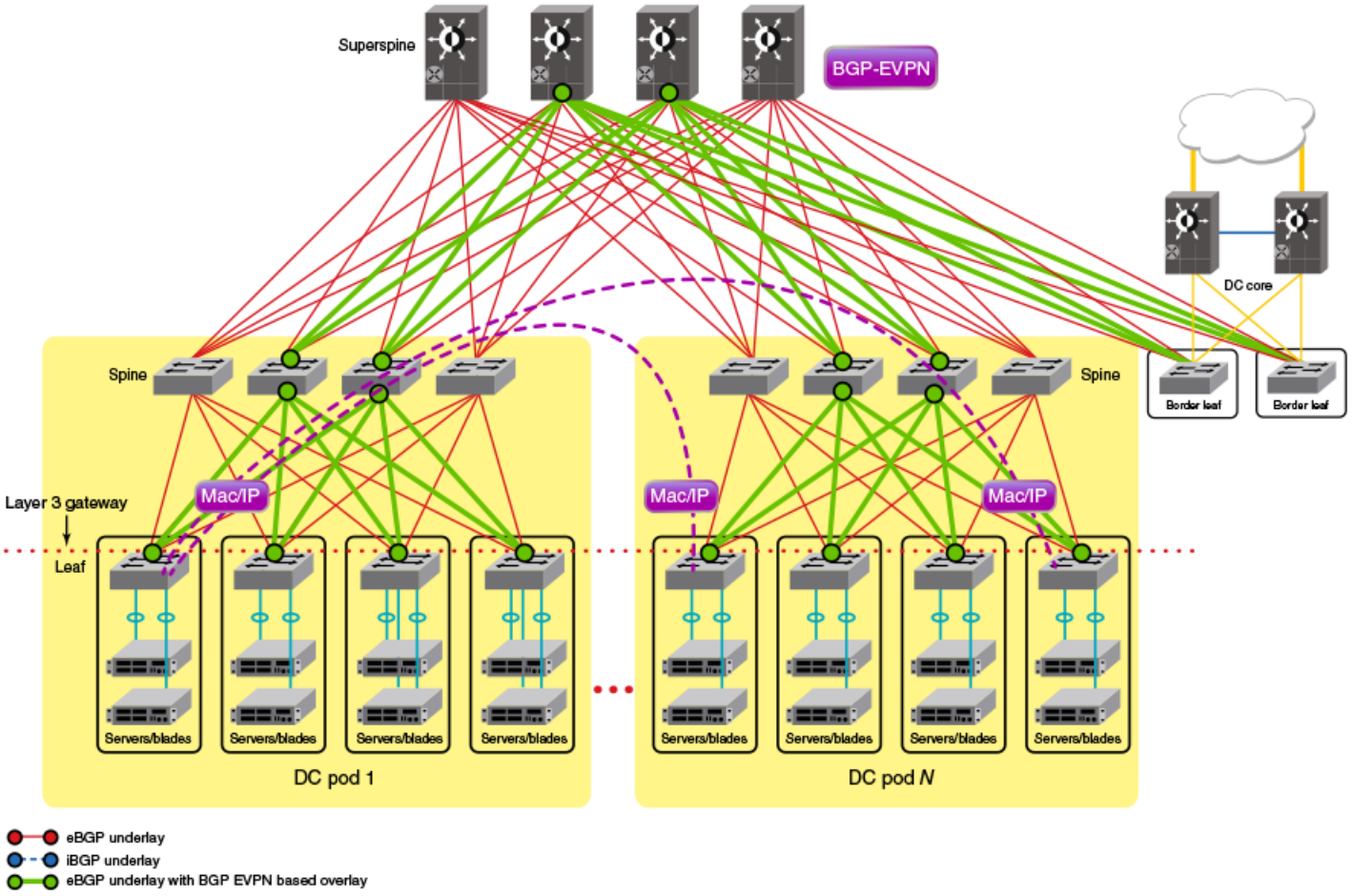
The following illustration depicts a 3-stage Clos topology that uses BGP EVPN. VXLAN tunnels are represented by dashed lines. This topology allows the provider to extend VLANs across racks with support from a Layer 3 underlay. Note that there is no controller, as VTEP autodiscovery is used. VTEPs are automatically discovered across the racks, and all MAC and IP address learning occurs automatically on the control plane. In this case, manual or controller intervention is not required. BGP EVPN can be extended to the border leaf nodes.

FIGURE 5 Overlay for a 3-stage Clos network without a controller



The following figure depicts a topology similar to that shown above, but with an optional optimized 5-stage folded Clos topology (superspine) for data center connectivity. Here the border leaf nodes connect directly to the superspine nodes. BGP EVPN can be extended to the border leaf nodes.

FIGURE 6 Overlay for an optimized 5-stage folded Clos topology without a controller



BGP EVPN control plane

This section summarizes the supported features of the BGP EVPN control plane.

Supported route types

The following table lists the BGP EVPN route types described in RFC 7432, including the Type 5 IP prefix route described in IETF draft "draft-ietf-bess-evpn-prefix-advertisement."

TABLE 2 BGP EVPN route types

Route type	Description
1	Ethernet auto-discovery (A-D) route
2	MAC/IP advertisement route
3	Inclusive multicast route
4	Ethernet segment route
5	IP prefix route

All BGP EVPN route types with VXLAN encapsulation are accepted and reflected by the BGP EVPN spine.

There is **no** termination or origination of the above routes at the spine. Because the spine nodes do not import any of the EVPN routes, they should be configured to retain all EVPN routes received from the leaf nodes and a super spine if one is deployed.

Supported BGP features

The following features are supported for BGP EVPN address-family:

- iBGP and eBGP peering
- IPv4 and IPv6 peer types
- Peer groups with EVPN neighbors
- Ability to negotiate only EVPN address family for IPv4/IPv6 peers
- iBGP route-reflector server
- Keeping next-hop unchanged for EBGP peers
- Ability to retain all EVPN routes without importing them
- Coexistence of all IPv4 and IPv6 BGP features for neighbors negotiating EVPN address family

Supported data-plane encapsulation types

The following data-plane encapsulation types are supported on the platforms in this release:

- VXLAN Virtual Extensible LAN)
- NSH (Network Service Header) (the default)

  NOTE
  MPLS is not supported.

The BGP encapsulation extended community attribute is carried with each route in the BGP control plane to signify the encapsulation to be used to reach the prefix.

Supported service-interface model

The VLAN-based service-interface model described in RFC 7432 is supported. Each VLAN is mapped to a unique virtual network interface (VNI) label. Within the data center, each VNI maps to a unique Ethernet virtual identifier (EVI). (This is referred to as the "single subnet per EVI" option in "draft-ietf-bess-evpn-overlay".) No VLAN translation service is supported. Within the data-center, VNI-to-VLAN mapping is local to each leaf, and the inner VXLAN frames may not carry an Ethernet tag. It is the administrator's responsibility to keep this mapping consistent within the data center.

Bundle service is achieved by means of bridge domains (BDs). Each BD corresponds to a unique EVI, and is given a unique VNI label. The Ethernet-tag field in the BGP EVPN routes is kept at 0, and no tagging of the forwarding frame is required. Advertising the router's logical interface (LIF) information that is associated with the Layer 2 route is not required at the remote end, as the BD's VNI label uniquely identifies the BD after the tunnel termination. In the case of flooding, frames are flooded on all of the LIFs of the BD after tunnel termination.

BGP-based underlay architecture supporting BGP EVPN

This section illustrates the basic BGP underlay architecture that is required to support the BGP EVPN overlay.

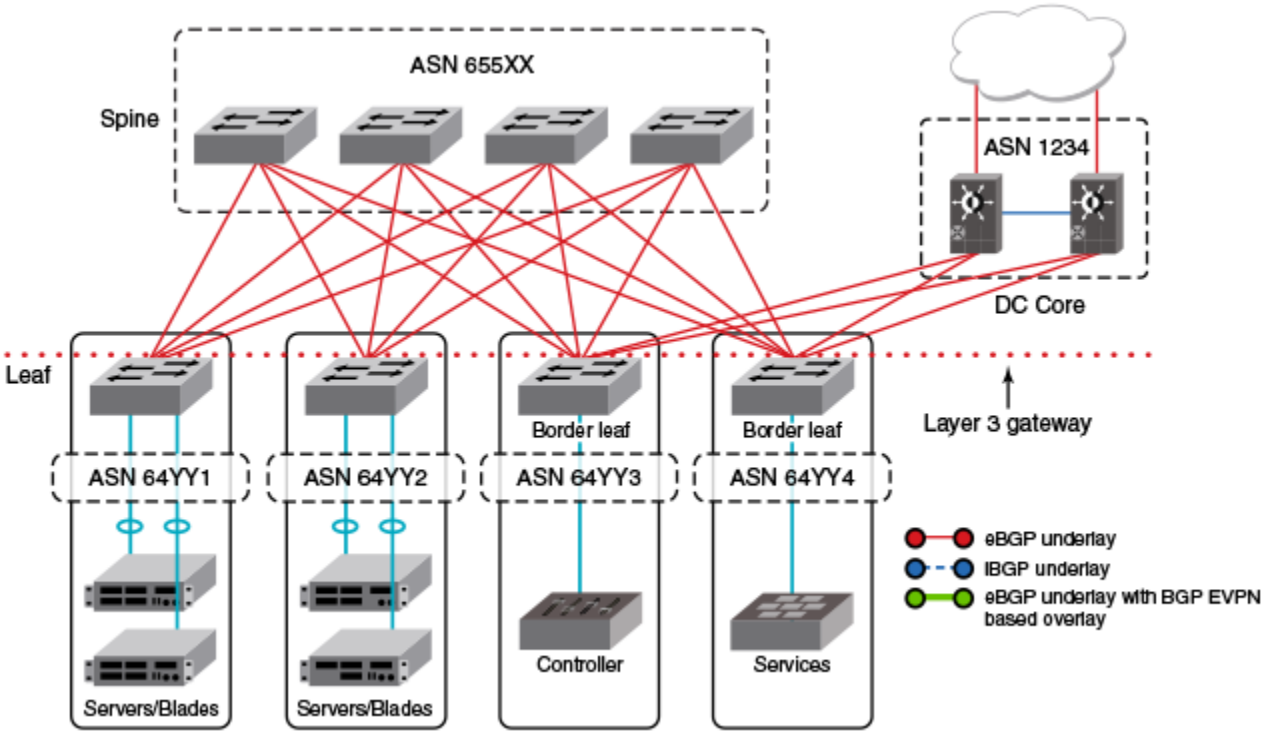
eBGP-based BGP EVPN

When eBGP is used for BGP EVPN in an IP Fabric, spine switches are configured to be in one autonomous system (AS). Each leaf switch or ToR pair is configured to be in a different AS from that of the spine switches, and advertises the routes by using local VTEP IP addresses as the next hop address.

For the BGP EVPN address family, spine switches are configured to advertise these routes to other eBGP peers, leaving the next-hop attribute unchanged.

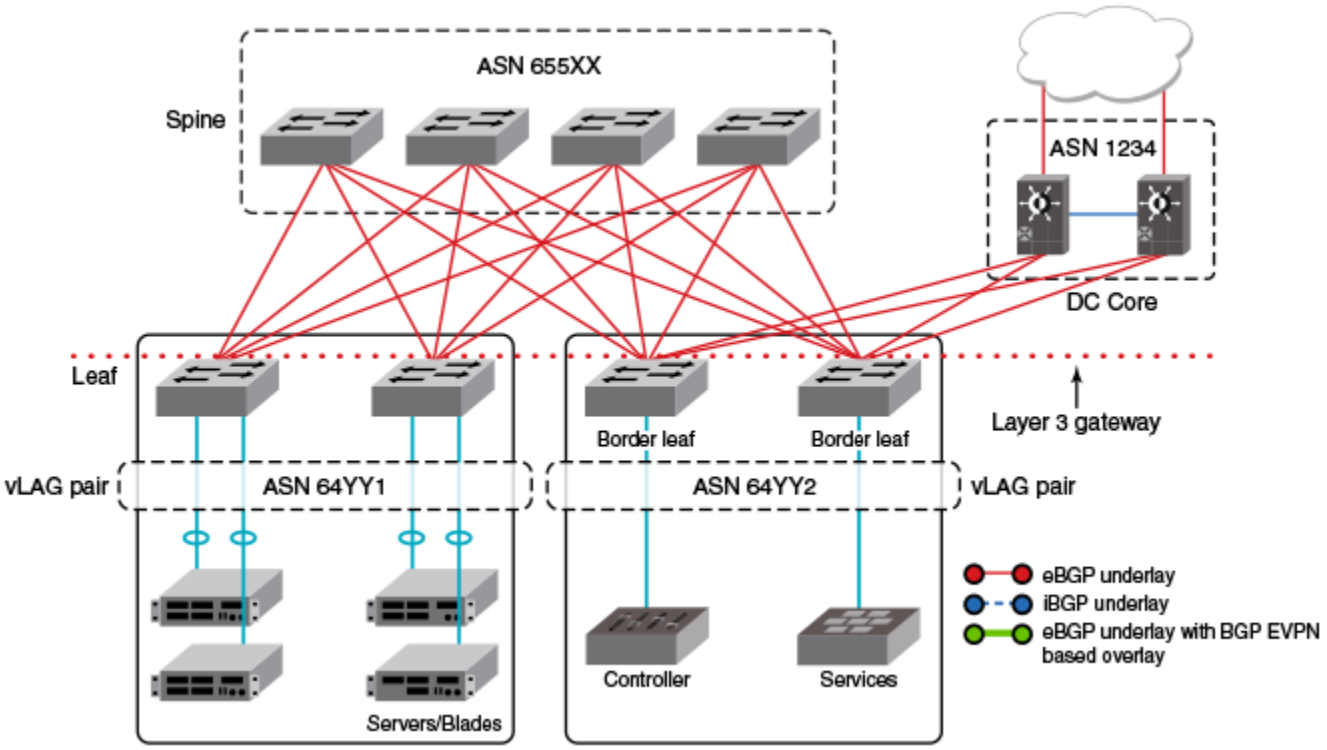
The following figure shows the eBGP underlay configuration in an IP Fabric where the spine switches are configured to be in one AS. Each leaf switch is configured to be in a separate AS.

FIGURE 7 eBGP underlay configuration with leaf switches in a separate autonomous system



The following figure shows the eBGP underlay configuration in an IP Fabric where the spine switches are configured to be in one AS. Leaf 1 and leaf 2 are configured to be in one AS, and leaf 3 and leaf 4 are configured to be in another AS.

FIGURE 8 eBGP underlay configuration with leaf switches in two separate autonomous systems



NOTE

When eBGP is used in conjunction with the IP unnumbered interfaces feature, it is important to ensure that sufficient Time To Live (TTL) values are available for hello messages to establish separate neighbor adjacencies on leaf switches in an IP Fabric. To do so, use the **neighbor ebgp-multihop** command and set the number of maximum hops to 2. Refer to the *Command Reference* for more information.

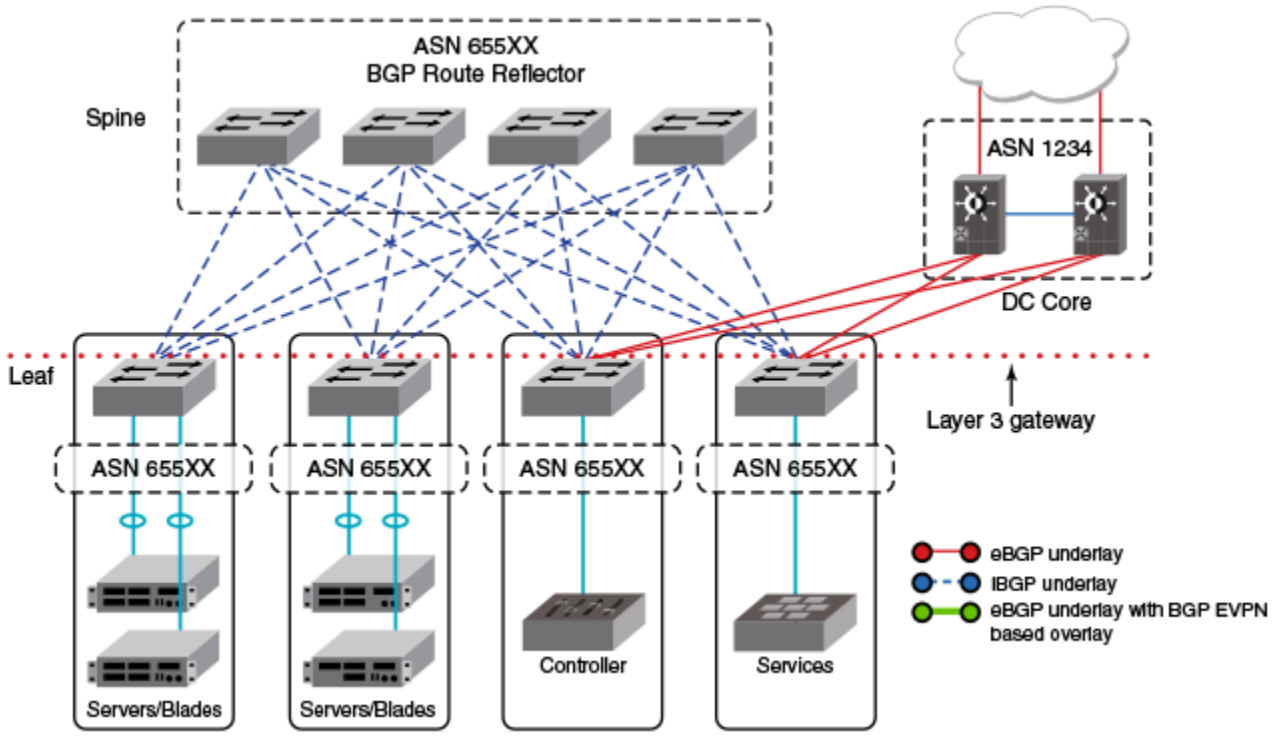
iBGP-based BGP EVPN

When iBGP is used for BGP EVPN in an IP Fabric, spine switches must be configured as route reflectors (RRs). This is because iBGP generally requires a switch to peer with every other device in the IP Fabric. When the spine switch is configured as an RR this situation is avoided.

However, BGP RRs only reflect the best route. Only the single, best prefix is reflected to each leaf, which is configured as an RR client, even if multiple Equal-Cost Multipath (ECMP) routes exist.

The following figure shows the iBGP underlay configuration for an IP Fabric. The entire IP Fabric is managed as a single ASN.

FIGURE 9 iBGP underlay configuration in an IP Fabric



BGP add path

The BGP add path feature provides the ability for multiple paths for the same prefix to be advertised without the new paths implicitly replacing the previous paths. Path diversity is achieved rather than path hiding.

When iBGP is used for BGP EVPN in an IP Fabric, spine switches are configured as route reflectors (RRs) so that a switch is not required to peer with every other device in the IP Fabric. Leaf switches are configured as RR clients.

BGP add path is supported for the BGP IPv4 and IPv6 unicast address families. It is not supported for the BGP L2VPN EVPN address family. BGP add path is not required for the L2VPN EVPN address family as a unique RD at each RBridge ensures that the same prefixes from multiple sources are not suppressed.

NOTE

For more information on BGP add path, refer to the "BGP4" and "BGP4+" chapters in the *Extreme SLX-OS Layer 3 Routing Configuration Guide*.

BGP EVPN NLRI

BGP EVPN distributes Network Layer Reachability Information (NLRI) for a network. BGP EVPN NLRI includes both Layer 2 and Layer 3 reachability information for end hosts residing in the network, and advertises both the MAC and IP addresses of the EVPN VXLAN end hosts.

The following table shows BGP EVPN support for NLRI.

TABLE 3 BGP EVPN support for NLRI

Support for NLRI	Description
Auto-discovery (AD) per ES	An ESI can be associated with more than one EVPN instance. When an interface with an associated ESI and EVPN instance comes online, an auto-discovery (AD) route per ES route is originated and installed in the corresponding EVPN instance. The ESI can be distinguished locally through the IP address of the peer.
MAC and MAC IP	The MAC and MAC-IP addresses of the EVPN VXLAN end hosts are advertised to peers, and the distribution of these addresses through BGP EVPN reduces unknown unicast flooding in the VXLAN.
Ethernet Tag Routes	Multicast Ethernet Tag routes advertise VLAN membership to peers, indicating the type of multicast tunnel on which flooded traffic can be received for a VLAN.
ES-Routes	Multihoming support is provided through the advertisement of BGP Ethernet Segment Routes (ES-Routes). An Ethernet Segment (ES) is the set of links connected to the same multihomed host. An Ethernet Segment Identifier (ESI) is associated with each Ethernet segment and advertised in the Ethernet Segment Route (ES-Route). For BGP EVPN, BGP initiates an ES-Route in the EVPN routing table and advertises it to EVPN neighbors, thus distributing NLRI for the network.
Inclusive Multicast Route (IMR)	BGP EVPN uses this BGP Type 0x3 Route Type to advertise multicast labels for multicast tunnel end-point discovery.
Prefix Routes	IPv4 and IPv6 prefix routes belonging to tenants (VRFs) are exchanged providing inter-subnet connectivity within the data center.

MAC address learning

Data plane (Layer 2) MAC address learning is enabled by default at the end points of the statically configured VXLAN tunnel..

When BGP routing is used, MAC learning in the forwarding plane on the auto-discovered VXLAN tunnel is disabled. Instead, MAC addresses are learned through the BGP EVPN control plane.

EVPN instances

An EVPN instance (EVI) is an EVPN routing and forwarding instance that spans across the leaf nodes participating in that EVPN.

An EVI is created by adding a VLAN/BD under EVPN configuration mode. Each VLAN/BD is assigned a unique EVPN instance ID (EVID) by the device.

Each EVI has a unique route distinguisher (RD) and one or more route targets (RT). RTs control the routes to be imported into and exported from the EVPN instance. An EVPN routing table, containing information about the various routes associated with the EVI, is maintained for each EVI instance.

EVIs can be configured with the following features:

- Route distinguisher (RD): Unique route distinguishers are assigned for an EVI. This value is automatically derived globally by means of the **rd auto** command, so that each EVI has an associated RD that is unique across the entire fabric. RDs can also be manually configured for a specific EVI by means of **rd** command under VLAN/bridge-domain configuration mode under EVPN. .
- Route distinguisher (RD): Unique route distinguishers are assigned for an EVI. This value is automatically derived globally by means of the **rd auto** command, so that each EVI has an associated RD that is unique across the entire fabric. RDs can also be manually configured for a specific virtual network identifier (VNI) under an EVI by means of the **rd (VNI)** command.
- Route targets: The Route Target (RT) extended community attribute, enabling logical grouping of EVPN routes that can be imported or exported, can be specified for an EVI. These route targets can be globally derived automatically by means of the **route-target (EVPN)** command. The **ignore-as** option for the **route-target (EVPN)** command should be used in instances where the leaf nodes are under different autonomous systems and you want to import the routes from one leaf node to another.

Route targets can also be manually configured for a specific EVI by means of the **route-target (VLAN/BD)** command. BGP EVPN uses these route targets to control the advertisement of EVPN routes.

- VLANs/BDs: VLANs and BDs can be added for an EVI by means of the **vlan add** and **bridge-domain add** commands.
- Duplicate MAC detection timer: When the same host MAC address is learned by two different devices, continuous MAC moves are triggered by the traffic originating from these hosts. A duplicate MAC detection timer can be configured, by means of the **duplicate-mac-timer** command, to detect these continuous MAC moves. This timer specifies both a time interval and the maximum threshold of MAC moves that can occur within the configured time interval before the MAC address is treated as a duplicate address and further processing of updates for that MAC address are blocked.

With a VLAN-based service model, each VLAN/BD corresponds to a unique EVPN instance, or MAC-VRF. Each route in EVPN belonging to a MAC-VRF has a unique RD value that differentiates routes from other MAC-VRFs. In addition, there are sets of route targets (export RTs) that are associated with each MAC-VRF; these are applied to the routes while advertising. RTs in the route are matched against the configured import RTs. If any of the RTs match, the route is imported; otherwise it is discarded.

Each VLAN/BD to be extended into an EVPN overlay must be added under the EVPN configuration block, as shown in the following configuration example.

```
evpn leaf1
 vlan 1020
 rd 1020:1
 route-target both 1020:100
!
vlan 3200
 rd 3200:1
 route-target both 3200:100
!
bridge-domain 10
 rd 10:1
 route-target both 10:100
!
bridge-domain 3100
 rd 3100:1
 route-target both 3100:1
!
!
```

Each VLAN/BD added to above EVPN configuration is considered an EVI and is assigned a unique EVID internally. EVIDs for VLANs/BDs are calculated as shown in the following table.

TABLE 4 Calculating EVIDs for VLANBs/BDs

VLAN/BD	EVI value
VLAN (1–4095)	VLANID
BD (1–4096+)	4096 + BD-ID

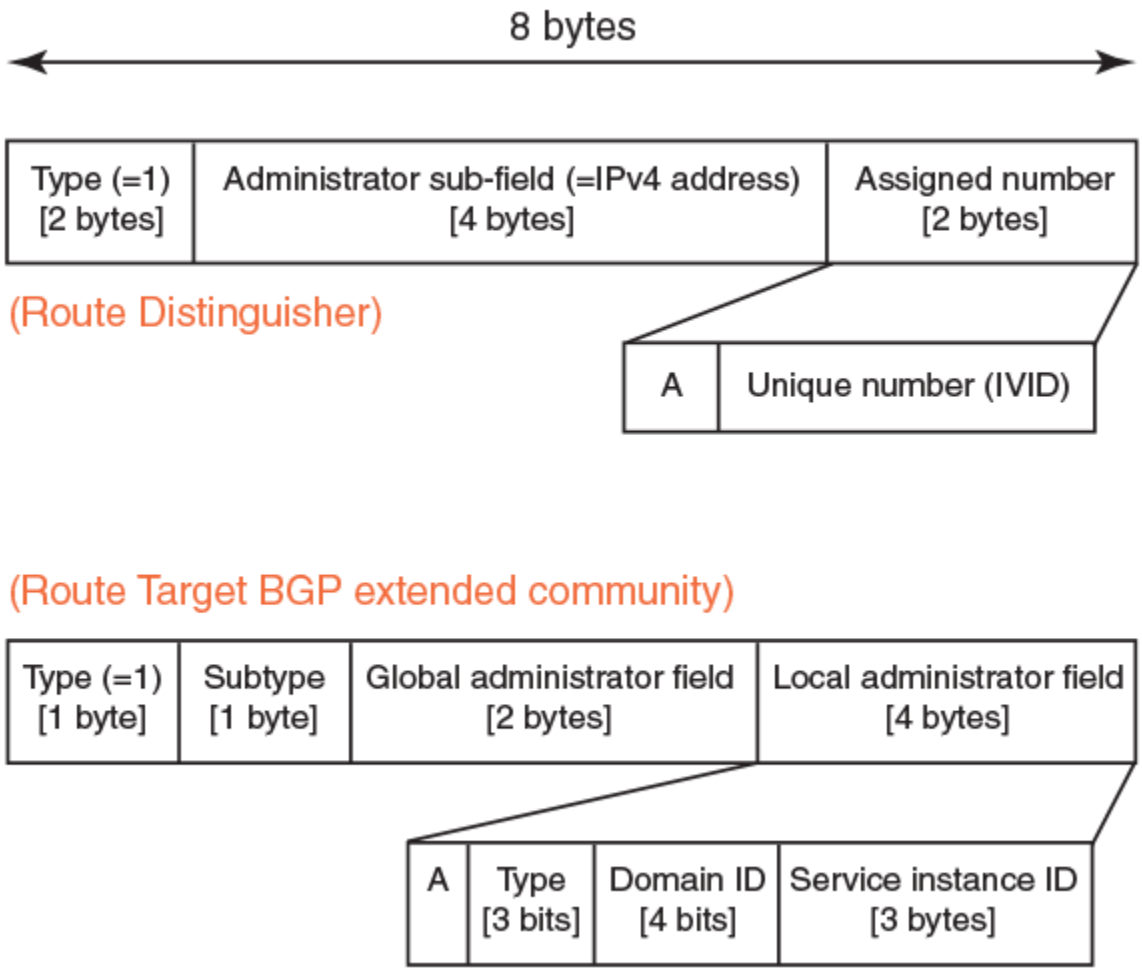
The EVID value in the preceding configuration example is not sent in the Ethernet-tag field in the routes. It is always 0 for VLAN-aware service. However, routes imported into the MAC-VRF table are classified against an EVID, which is shown in the Ethernet-tag field.

Autogeneration of route distinguisher and route target

Because there can be thousands of VLANs/BDs in the network, configuring each of them individually on each router is cumbersome. In order to simplify the configuration, the autogeneration of RD and RTs is recommended.

The RD of EVPN routes is encoded as a Type-1 route-distinguisher as described in RFC4364. The formats of RD and BGP RT extended community are shown in the following figure.]

FIGURE 10 RD and RT formats



The most significant bit "A" of the assigned number field stands for automatic or manual RD. In the case of a manual RD value, this bit is set, and the unique identifier value is set to the internal VLAN ID (IVID) to keep the RD unique across VLANs and bridge domain IDs (BD-IDs). The administrator sub-field of the RD will be set to the BGP router ID.

The global administrator field in the RT is set to the AS number. The local administrator field in the RT is expanded as shown in the preceding figure. Bit-value "A" stands for automatic or manual RT. In case of an automatically derived RT value, this bit is set to 0. The service instance ID field carries the VLAN or VNI value. The Type field is set, according to the value set in service instance ID space, to either 4 (=EVID) or 1 (=VNI). The domain-id value is not used currently and is set to 0. RTs with the following service instance types are attached according to neighbor encapsulation:

TABLE 5 Neighbor encapsulation and RT service types

Neighbor encapsulation	RT service instance type
VXLAN	VXLAN (1)
NSH	EVID (4)

The configuration of VLANs/BDs that use autogenerated RD and RT values is simplified as shown in the following configuration example.

```
evpn leaf1
 rd auto
 route-target both auto ignore-as
 vlan add 1000-2000
 bridge-domain add 1000-2000
!
```

VLAN and bridge domains that are added with range, RD, and RT values are specified as "auto". In scenarios where multiple leaf nodes belong to different ASs, the "ignore-as" option is recommended. When that option is specified, the AS number field in the RT in the route is not compared against local AS number.

BGP routing tables

The EVPN table holds VPN routes belonging to L2VPN EVPN address family. These routes may be received from an EVPN peer, originated locally, or exported from other VRFs in the case of prefix routes.

EVPN route types 1 through 4 received from remote peers are imported only into the MAC-VRF table, except for MACIP host routes that may get converted to IPv4/v6 host routes and imported into the IP-VRF table. EVPN routes received from remote peers are validated against import rules, and are added to the VPN table only if validation passes. The following table lists the import rules for peer types and route types.

TABLE 6 Import rules for peer and route types

Received from peer type	Route type	Import rules
VxLAN peer	MAC/IP, Ethernet Tag	RTs in the route are looked up in the MAC-VRF table. Route is imported if: 1. RT in the route matches RTs configured for VLAN/BD. 2. When VLAN/BD instance is not configured manually, VLAN/BD should exist in the system, and added to EVPN. 3. L2VNI in the label field is same as corresponding VNI for the VLAN/BD. For MAC-IP (ARP/ND) routes, if no match is found in MAC-VRF, RTs in the route will be looked up in IP-VRFs. For each IP-VRF matching the RT, the route is imported in that IP-VRF.
	Ethernet Segment	Route is discarded.
	Auto-Discovery	Route is discarded.
	IP Prefix	RTs in the route are looked up in IP-VRFs. For each IP-VRF matching the RT, route is imported into that IP-VRF.
	Ethernet Segment	Route is imported if RT in the route matches auto ES-RT of the locally configured MCT interface.
	Auto-Discovery	Route is imported in the MAC-VRF if any RT in the route matches any VLAN/BD instance.
	IP Prefix	Same as that of VXLAN peer.

The EVPN table can hold routes even if they are not imported into MAC-VRF or IP-VRF tables. This is required at the spine or border-leaf nodes to reflect the routes to other leaf nodes or to other data centers, respectively. The **retain route-target all** command is used to configure this.

The consolidation of the routes from different sources (RDs) occurs in the MAC-VRF table after the routes pass through route-target checking and import filtering. The following operations are performed exclusively in the MAC-VRF table:

- VTEP discovery
- MAC move detection
- MAC dampening
- ES-based route usage

The following table lists advertisement behavior by route type.

TABLE 7 Advertisement behavior by route type

Route type	Advertised to VXLAN peer?	Advertised to NSH peer?
Auto-discovery	No	Yes
MAC	Yes (without ES)	Yes
MACIP (ARP/ND)	Yes (without ES)	Yes
Inclusive Multicast	Yes	Yes
Ethernet Segment	No	Yes
IP-Prefix (IPv4Prefix/IPv6Prefix)	Yes	Yes

BGP L2VPN EVPN address family

The BGP L2VPN EVPN address-family configuration level provides access to commands that allow you to configure BGP EVPN. The BGP L2VPN EVPN address family uses components of BGP that are independent of the IPv4 and IPv6 unicast address families. Both iBGP and eBGP are supported.

The L2VPN EVPN address family supports the EVPN Subsequent Address Family Identifier (SAFI), an address qualifier that provides additional information about the Network Layer Reachability Information (NLRI) type for a given attribute.

The commands that you enter at this level apply only to the BGP L2VPN EVPN address family. BGP L2VPN EVPN address family capability can be negotiated along with the IPv4 or IPv6 address family. A separate BGP session is not required for the BGP EVPN address family.

To negotiate only the BGP L2VPN EVPN address family on an IPv4 neighbor, you must explicitly deactivate the IPv4 unicast address family using the **no neighbor activate** command.

To negotiate only the BGP L2VPN EVPN address family on an IPv6 neighbor, you must activate the neighbor only under the BGP L2VPN EVPN address family.

The following configuration options are allowed under BGP address-family L2VPN EVPN configuration mode:

```
device(config-bgp-evpn)# ?

Possible completions:

 client-to-client-reflection   Configure client to client route reflection
 graceful-restart              Enables the BGP graceful restart capability
 neighbor                      Specify a neighbor router
 retain                        Retain route targets
 vtep-discovery                Enable VTEP discovery
```

The following neighbor configuration options are allowed under BGP address-family L2VPN EVPN configuration mode:

```
device(config-bgp-evpn)# neighbor 10.1.1.1 ?

Possible completions:

  activate                Allow exchange of route in the current family mode
  allowas-in              Disables the AS_PATH check of the routes learned
                          from the AS
  enable-peer-as-check    Disable routes advertise between peers in same AS
  maximum-prefix
  next-hop-unchanged      Next hop unchanged
  route-map               Apply route map
  route-reflector-client  Configure a neighbor as Route Reflector client
  send-community          Send community attribute to this neighbor
```

Automatic VXLAN tunnel endpoint discovery

VXLAN tunnel endpoint (VTEP) IP addresses are carried in every BGP EVPN route so that the leaf device receiving the BGP EVPN updates triggers VXLAN tunnel creation using this remote VTEP IP address. Remote VTEP discovery is enabled by default for BGP EVPN when the BGP L2VPN EVPN address family is enabled.

BGP devices can determine which VLANs are common between two devices and extend those VLANs over the VXLAN tunnel between the two devices. VTEP IP address is carried in BGP EVPN updates in next-hop network address field of the MP_REACH_NLRI attributes. BGP deletes VXLAN tunnels associated with remote VTEP when all of the routes from remote VTEP are withdrawn. When the **no vtep discovery** command is configured, all dynamically discovered VXLAN tunnels are deleted, and BGP does not create VXLAN tunnels automatically. In addition, EVPN routes are not installed into the system even though statically configured VXLAN tunnels exist.

BGP next-hop-unchanged

The BGP next-hop-unchanged feature is supported for the L2VPN EVPN address family, allowing BGP to send updates to eBGP peers with the next hop in the MP_REACH_NLRI attribute unchanged.

By default, eBGP BGP speakers change the next hop while sending the updates to eBGP neighbors. The BGP next-hop-unchanged feature overrides this behavior so that the next-hop address remains unchanged while updates are sent to eBGP peers. The BGP speaker is forced to retain the next hop address in the BGP updates received from neighbors. BGP next-hop-unchanged should be configured on the spine switch for each EVPN neighbor if the leaf switch is an eBGP peer. BGP next-hop-unchanged should be configured on the leaf switch for each EVPN neighbor if the spine switch is an eBGP peer. This means that eBGP BGP speakers will not change the next hop while sending updates to eBGP neighbors in an IP Fabric. Therefore, by configuring BGP next-hop-unchanged under the BGP L2VPN EVPN address family, changes to the next hop address in BGP EVPN updates are prevented.

NOTE
BGP next-hop-unchanged should be configured on spine nodes for all types of BGP deployments.

BGP retain route target

The BGP retain route target feature allows a spine node to accept all route targets (RTs).

Because spine switches do not have EVPN instances (EVIs) and VRFs configured, BGP EVPN routes are filtered as otherwise none of the RTs in the routes match the local route-target import configuration. When spine switches are configured as RRs, or establish eBGP peering with the leaf nodes, the BGP retain route target feature should be enabled so that the Spine switches do not do route-target filtering. This means that all route targets are retained and the route-target attributes in the EVPN routes are not modified or removed.

BGP retain route target is supported for the BGP L2VPN EVPN address family only and is used in BGP EVPN configurations. It is not supported for the IPv4 and IPv6 unicast address families.

RIBout AS path check

The RIBout AS path check feature enables the outbound AS_PATH check function so that a BGP sender speaker does not send routes with an AS path that contains the ASN of the receiving speaker. If the remote AS of the neighbor appears in the AS path segment of the NLRI, the NLRI is not added to RIBOut of the peer.

When the AS path check is enforced for the sender using the **neighbor enable-peer-as-check** command, it saves the amount of RIBout memory used to store routes that would otherwise be stored in sender's RIBout, advertised to the peer and be discarded by the receiver, if the **neighbor allowas-in** command is not configured. Convergence time is also improved.

BGP legacy features supported for the L2VPN EVPN address family

The following BGP features, already supported for IPv4 and IPv6 unicast address families, are supported for the BGP L2VPN EVPN address family and used in BGP EVPN configurations:

- BGP extended community: The BGP extended community feature filters routes based on a regular expression specified when a route has multiple community values in it. The BGP extended community feature is supported for the L2VPN EVPN address family for both spine and leaf nodes. When a spine or leaf node receives BGP EVPN routes from other spine nodes, it checks the route-target extended community attribute of the routes. If the route-target is the same as the import target of the given EVPN instance on the local leaf node, the leaf node adds the route to the EVPN routing table. If the spine node is configured with the BGP retain route target feature, this checking is bypassed in the spine. If a static MAC address is redistributed to BGP, it is advertised with the "Sticky MAC" extended community attribute. The next hop router MAC address for prefix routes is advertised as the default gateway extended community attribute. The same check is performed for prefix routes; however the comparison is made against the RT configured for VRFs rather than EVIs.
- BGP graceful restart: BGP graceful restart (GR) can be configured for the L2VPN EVPN address family. When GR capability is negotiated, neighboring devices participate in the restart, helping to ensure that no route and topology changes occur in the network for the duration of the restart. GR helps retain the peer routes when a restart occurs during HA failover. When the GR feature is enabled for the L2VPN EVPN address family, existing sessions are not affected and require neighbor reset to negotiate the GR capability.
- BGP peer groups: BGP peer groups can be configured for the L2VPN EVPN address family so that neighbors with the same attributes and parameters can be grouped together.
- BGP route reflection: The BGP route reflection feature is supported for the L2VPN EVPN address family. When iBGP is used for BGP EVPN in an IP Fabric, spine switches are configured as route reflectors (RRs) and leaf switches are configured as route

reflector clients. You can configure a leaf switch as a route reflector client from the spine switch using the **neighbor route-reflector-client** command. Each leaf switch should be configured so that they all belong in the same cluster.

- Client-to-client-reflection: The BGP client-to-client reflection feature is supported for the L2VPN EVPN address family. For iBGP, if the leaf switches are configured as route reflector clients and are connected in a full iBGP mesh, you can disable client-to-client reflection on the spine switch which is configured as an RR. By default, in an IP Fabric, the clients of a route reflector are not required to be fully meshed and the routes from a client are reflected to other clients.
- Disabling the BGP AS_PATH check: A device can be configured so that the AS_PATH check function for routes learned from a specific location is disabled, and routes that contain the recipient BGP speaker's AS number are not rejected. When eBGP is configured for BGP EVPN in an IP Fabric, the AS_PATH check function can be disabled on a leaf switch so that route updates from the spine layer are not discarded by the leaf.

Ethernet Segment Identifiers for BGP routing

An Ethernet Segment is a set of Ethernet links that connect a multihomed device to a BGP router. Ethernet Segments are assigned a unique identifier, referred to as an Ethernet Segment Identifier (ESI). ESIs can be configured in cluster client configuration mode.

Ethernet links that connect single-homed devices are not required to have an ESI value associated with them. For BGP EVPN, each BGP device advertises an Ethernet Segment Route (ES-Route) informing other BGP devices that it is connected to that ES. The other BGP devices can then detect if they are connected to the same ES. The user can use the **esi** command in port-channel configuration mode to configure the port-channel to derive the ESI value automatically by means of Link Aggregation Control Protocol (LACP). ESI value can be manually also configured on the cluster client. This is needed for static port channels where LACP is not involved.

BGP EVPN-based MCT cluster formation

MCT cluster formation leverages BGP EVPN Ethernet Segment (ES) and Auto-Discovery (AD) routes and employs proprietary mechanisms to avoid loops in the cluster in transient configuration scenarios.

There is no separate EVPN instance for MCT and non-MCT services. Therefore, in order to configure MCT even on a standalone MCT cluster, EVPN configuration is required.

Dependence on EVPN instances

Member VLAN/bridge-domain (BD) configuration, by means of the **member-vlan** and **member-bridge-domain** commands, is deprecated under cluster configuration. EVPN configuration is now required (along with BGP neighbor and MCT cluster/client) to form an MCT cluster. An MCT cluster automatically becomes a member of a VLAN/BD configured under EVPN.

NOTE

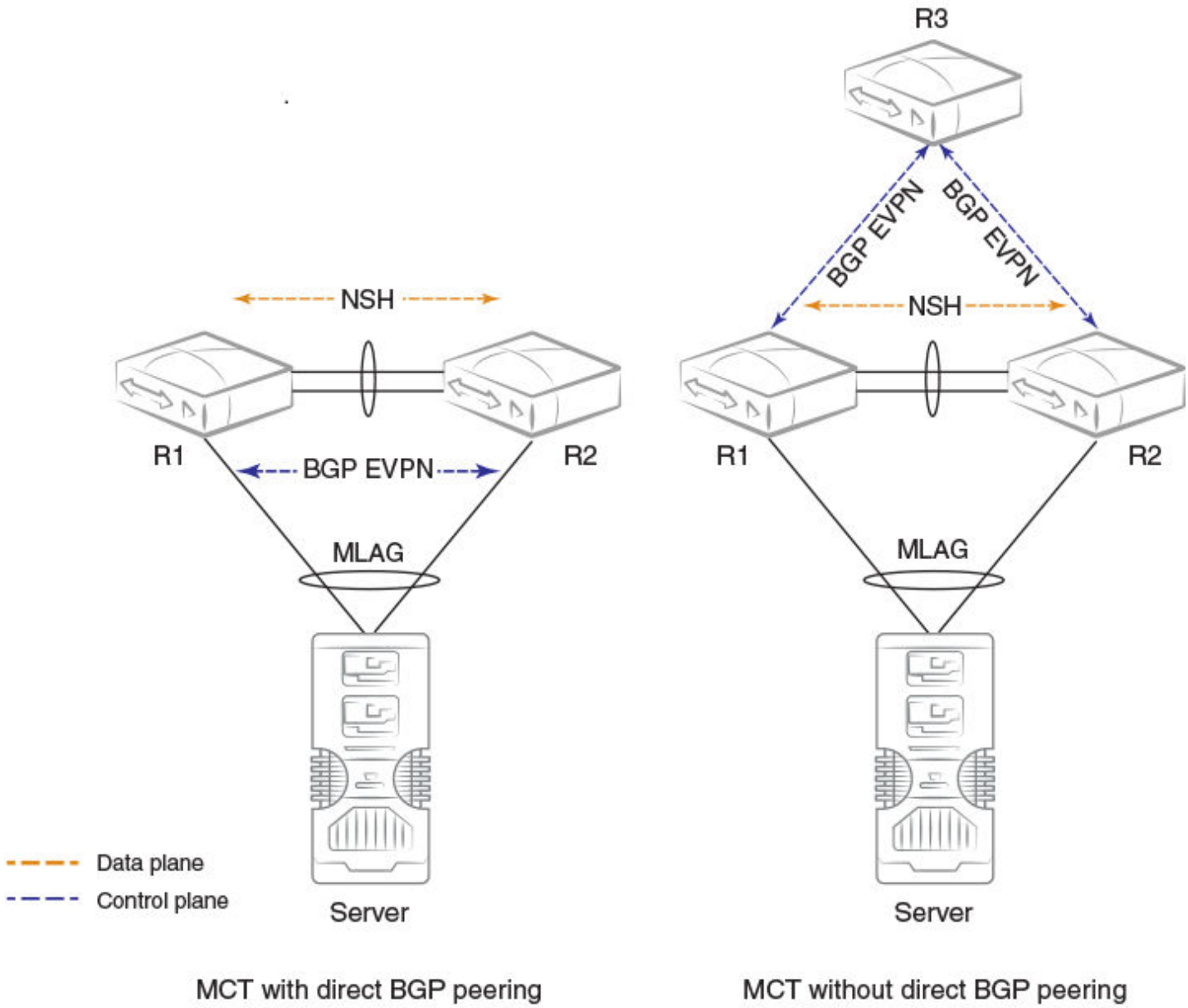
During an upgrade from a previous release, if this configuration was applied, it results in the automatic creation of an EVPN configuration instance with auto-generated RD and RT and auto-specified VLANs/BDs being added to the EVPN configuration. In case of a downgrade to a previous release, the member VLAN/BD configuration is lost. The user must reconfigure the VLAN/BD under MCT cluster configuration.

BGP EVPN peering for an MCT cluster

MCT cluster formation is controlled by BGP EVPN peering between the cluster members, and it should be configured with a platform-specific (NSH) encapsulation type.

The following figure shows typical use cases of MCT. An MCT cluster on a SLX 9140 or SLX 9240 requires a direct Inter-Chassis Link (ICL) between the peers.

FIGURE 11 MCT use cases



NOTE
In the above topology diagrams, only the first topology is supported.

The first topology shows a standalone MCT cluster and is the simplest MCT use case where BGP EVPN peering exists only between the cluster members. An NSH tunnel is established over the ICL link, and the control and data planes are set up over the same path.

The second topology shows separate control-plane and data-plane paths. BGP EVPN peering is not direct between the cluster members. This use case is not supported in the current release.

The third topology shows MCT cluster formation without an ICL link. This use case is not possible with the SLX 9140 or SLX 9240 platforms because a multihop NSH tunnel cannot be established.

Handling of ES/AD routes in BGP

Ethernet segment (ES) and auto-discovery (AD)-per-ES routes are exchanged between MCT peers for Designated Forwarder (DF) election. An ES route is originated when the MCT cluster client is deployed, and a set of AD-per-ES routes is originated when the BGP cluster client interface is operationally up.

ES routes carry BGP ES-import route-target extended community, which is automatically constructed from the ESI value. ES- import RTs are automatically added in EVPN whenever a cluster client interface with a given ES is configured. An ES route with a given ESI value is imported only by those routers that have membership in that ES. Similar to other EVPN routes, ES/AD routes are added to the BGP RIB-in table and are advertised to EVPN neighbors.

AD-per-ES routes are required to carry the RTs of all EVIs that are members of the ES (cluster client interface). Because there can be a large number of EVIs that are members of the ES (4K VLANs + BDs), the AD route may not fit in a single BGP update message that is only 4K bytes. Multiple AD-per-ES routes with unique RDs are generated for each ES, as shown in the following table.

RD index for RD (Router-ID: RD-index)	RTs of the EVIs attached to AD-per-ES route
1	VLAN 1-127
2	VLAN 128-255
3	VLAN 256-383
…	…
32	VLAN 3968-4095
33	BD 1-127
34	BD 128-255
34	BD 256-383
…	…

AD-per-ES routes carry ESI labels in ESI label extended community for split-horizon filtering.

BGP EVPN-based VXLAN overlay

BGP EVPN provides support for control plane signaling and tunnel discovery in a VXLAN overlay network.

Inclusive-multicast routes provide for the extension of VLAN/BD over the tunnels, and MAC, MACIP, and prefix routes provide the signaling of control plane routes for tenants. Forwarding across the tenants or towards the core occurs over VXLAN overlay tunnels. The flooding of BUM traffic is achieved through ingress replication. The following are salient points for BGP EVPN-based VXLAN overlay networks:

- A mesh of tunnels between leaf nodes that extend common EVI or VRF within the data center is required.
- Flooding occurs through Ingress replication. There is no support for multicast.
- Spine nodes provide the load balancing for overlay tunnels, as well as alternate route sources for BGP.

- Logical VTEP (LVTEP) provides inherent load balancing in the underlay network for dual-homed hosts. There is no need for aliasing. Devices in the underlay Layer 3 network may have distinct roles: leaf, spine, border leaf, or colocated border leaf and spine.

Underlay architectures

The underlay architecture in a Layer 3 Clos network can be either iBGP–based or eBGP–based.

iBGP spine and leaf

With iBGP–based underlay, all spine and leaf nodes are in same AS. Spine nodes act as route reflectors and do not terminate any tunnel. On the other hand, leaf nodes originate and terminate all of the tunnels and routes.

eBGP spine and leaf

With eBGP–based underlay, each leaf node is assigned a distinct AS number (ASN). Usually iBGP underlay architectures provide higher BGP scalability, but with less control over the routes getting exchanged. On the other hand, eBGP underlay provides lower BGP scalability but higher control over routes, because policies and filters can be applied on the originating ASNs.

Border leaf architectures

A border leaf is a special leaf node, typically redundant, that sits at the edge of the data center and provides termination, hand-off, or control-plane extension towards the WAN or core. Similarly, it provides the reverse functionality from the WAN/core towards the data center.

Layer 2/Layer 3 handoff

In this scenario, VXLAN tunnels are terminated at the border leaf. Depending on the interconnect technology being used between data centers extending over the WAN (VPLS, VXLAN Layer 2 extension, or any other interconnect scheme), a border leaf bridges the two domains by means of forwarding-plane learning without any control-plane extension. In the case of Layer 3, all VRFs in the data center are terminated at the border leaf and traffic is routed towards the WAN. Similarly, Layer 3 traffic received from the WAN is routed and forwarded over tunnels in the data center. Because Layer 3 routes across multiple VRFs can be imported into a single (interconnecting) VRF, it is not necessary to configure all possible VRFs in a given data center on the border leaf.

Layer 2 and Layer 3 control-plane extension

In this scenario, VXLAN tunnels are extended by means of a border leaf instead of being terminated. This is primarily the spine functionality being provided by the border leaf, except that the control and forwarding planes are extended over the WAN/core in the case of the border leaf.

Service leaf

A border leaf may act as a transparent route reflector/forwarder (with iBGP/eBGP), or it may terminate tunnels along with providing route reflection. In the service leaf model, a border leaf also acts as just another leaf node in the network and may be extending Layer 2, Layer 3, or both services.

VXLAN overlay gateway configuration

An overlay gateway configuration must be present and should remain in the active state in order to exchange routes in the VXLAN overlay network. The following is an example configuration.

```
overlay-gateway gateway1
 type layer2-extension
 ip interface Loopback 1
 map vni auto
 activate
!
interface Loopback 1
```

```
 ip address 192.168.32.10/32
 no shutdown
```

The following are criteria for accepting and originating routes in a VXLAN overlay network:

- An overlay gateway configuration must exist.
- The tunnel type must be Layer 2 extension.
- A source VTEP IP address must be configured. In case the source IP address is obtained from a loopback interface, the loopback interface must be configured.
- The overlay gateway must be activated, by means of the **activate** command in overlay gateway configuration mode.
- The VLAN/bridge domain (BD)-to-VNI mapping (auto or manual) must exist in order to accept or originate routes for specific VLANs/BDs.

The above criteria affect only the exchange of routes between BGP neighbors with VXLAN encapsulation. When the overlay gateway is deactivated or unconfigured, all dynamic tunnels are deleted, the EVPN routes received with VXLAN encapsulation are removed, and the RIB-out entries for the BGP EVPN neighbors configured with VXLAN encapsulation are flushed.

Dynamic VTEP discovery

Dynamic VTEP discovery is a configurable option. It can be enabled under BGP EVPN address family configuration mode as shown in the example below.

```
router-bgp
 address-family l2vpn evpn
    vtep-discovery
!
```

By default, VTEP discovery is enabled. When it is enabled, if a route with BGP encapsulation "extended community" as the VXLAN type is imported into the MAC-VRF table, a VXLAN tunnel (if it does not already exist) is created in the system with the destination IP address as the next-hop address of the route. MAC learning in the forwarding plane on BGP-discovered VXLAN tunnels is automatically disabled.

When the above "vtep-discovery" is disabled, all dynamically discovered VXLAN tunnels are deleted from the system.

The following is an example configuration.

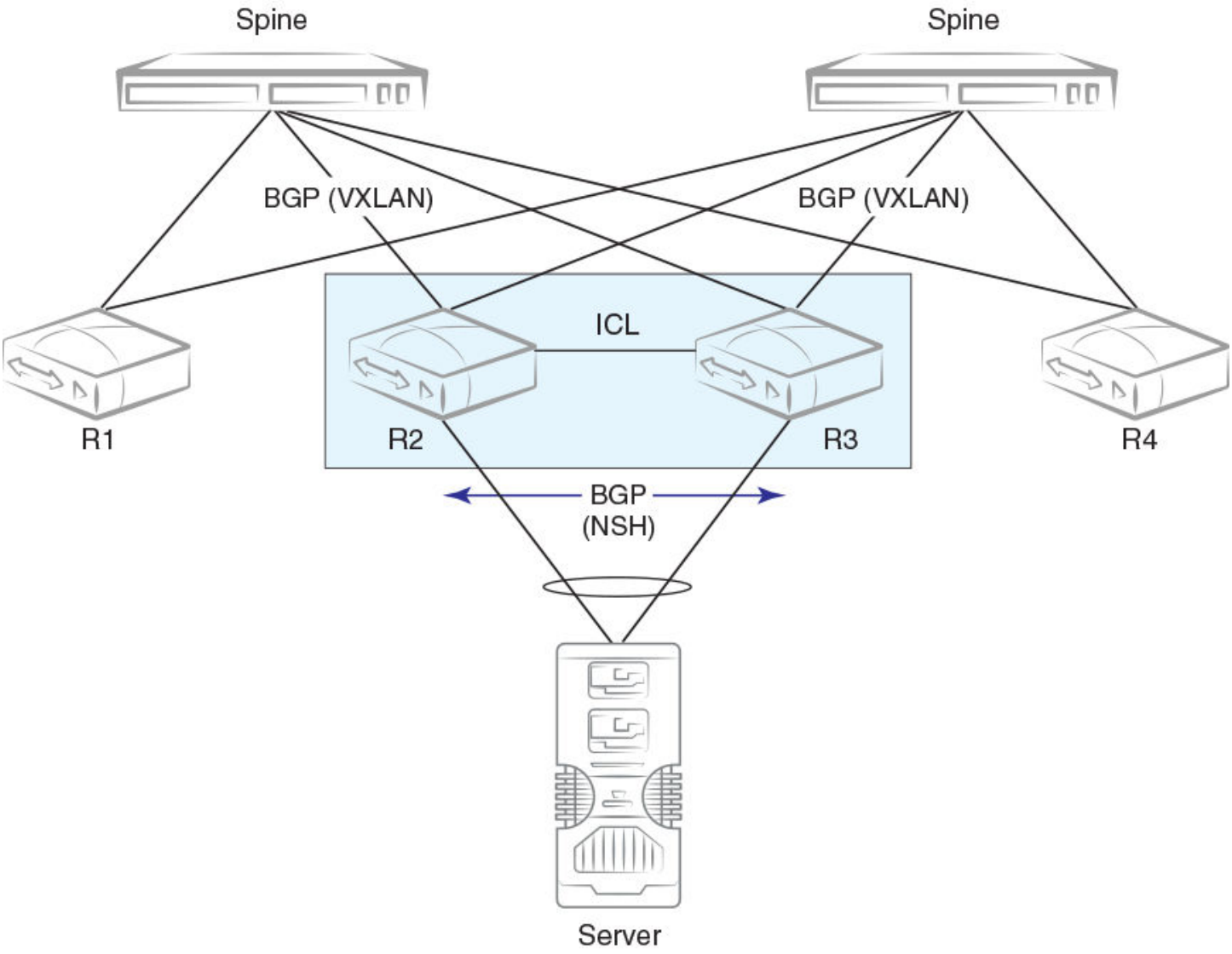
```
overlay-gateway gateway1
 type layer2-extension
!
!
site tunnel1
 ip address 192.168.32.20
 extend vlan add 10-100
 extend bridge-domain add 10-100
    !
activate
!
```

MCT and logical VTEP

MCT cluster configuration is required for logical VTEP (LVTEP) operation. Depending on the platform, MCT peers require BGP EVPN route reachability with the corresponding encapsulation (NSH).

The following figure shows a typical example of two-node MCT cluster in a VXLAN IP fabric. MCT cluster members have two BGP peerings: one with MCT cluster encapsulation, and another with a VXLAN to the spine.

FIGURE 12 MCT with logical VTEP and direct BGP peering



NOTE
MPLS is not supported.

In the presence of an LVTEP, BGP with VXLAN encapsulation cannot be used to exchange EVPN routes between MCT peers. This is because VXLAN routes originate with the source VTEP IP address, which is same across MCT peers. In addition, VXLAN as the transport between MCT peers is not supported.

The following are requirements and recommendations for overlay gateway configurations on MCT cluster nodes:

- Configuring MCT cluster nodes as VXLAN tunnel end points is not recommended. MCT cluster nodes should be configured to form an LVTEP if VXLAN overlay is required.
- The same source VTEP IP address should be configured on all nodes in the MCT cluster in order to form an LVTEP.
- The same set of VLANs/BDs should be configured in the system and added under BGP EVPN on all nodes in the MCT cluster.

- VLAN/BD-to-VNI mapping should be the same on all nodes in the MCT cluster.
- If routing is enabled on a VLAN/BD, static anycast gateway is recommended.

Multi-VRF for BGP EVPN

Multi-VRF must be configured in an IP Fabric to support asymmetric or symmetric routing. The link between the leaf and spine switches must be configured in the default VRF. For a nondefault VRF, traffic is routed at the leaf switch and is forwarded to a VXLAN tunnel.

For asymmetric routing, all VLANs are configured on every leaf switch and enabled for IP routing and forwarding. Traffic flows through different routes in different directions. This severely affects scaling with each switch carrying the MAC and MAC-IP routes for hosts that are not necessarily locally connected.

For symmetric routing, a VLAN is configured only at the leaf switch where a local host exists. A unique, common Layer 3 virtual network identifier (VNI) number is generated for all leaf nodes for the VRF instance. The ingress leaf switch performs a Layer 3 lookup and routes the packets towards the remote leaf switch over the common Layer 3 VNI. The inner MAC destination address (DA) of the packet is rewritten to that of the gateway MAC address advertised by the remote leaf switch and the packet is then encapsulated in a VXLAN tunnel and sent to the remote leaf switch. The remote leaf switch then terminates the VXLAN tunnel. Because the MAC DA of the inner packet is now that of the gateway MAC address advertised by the leaf switch, it performs Layer 3 lookup and the packet is routed to the locally attached host.

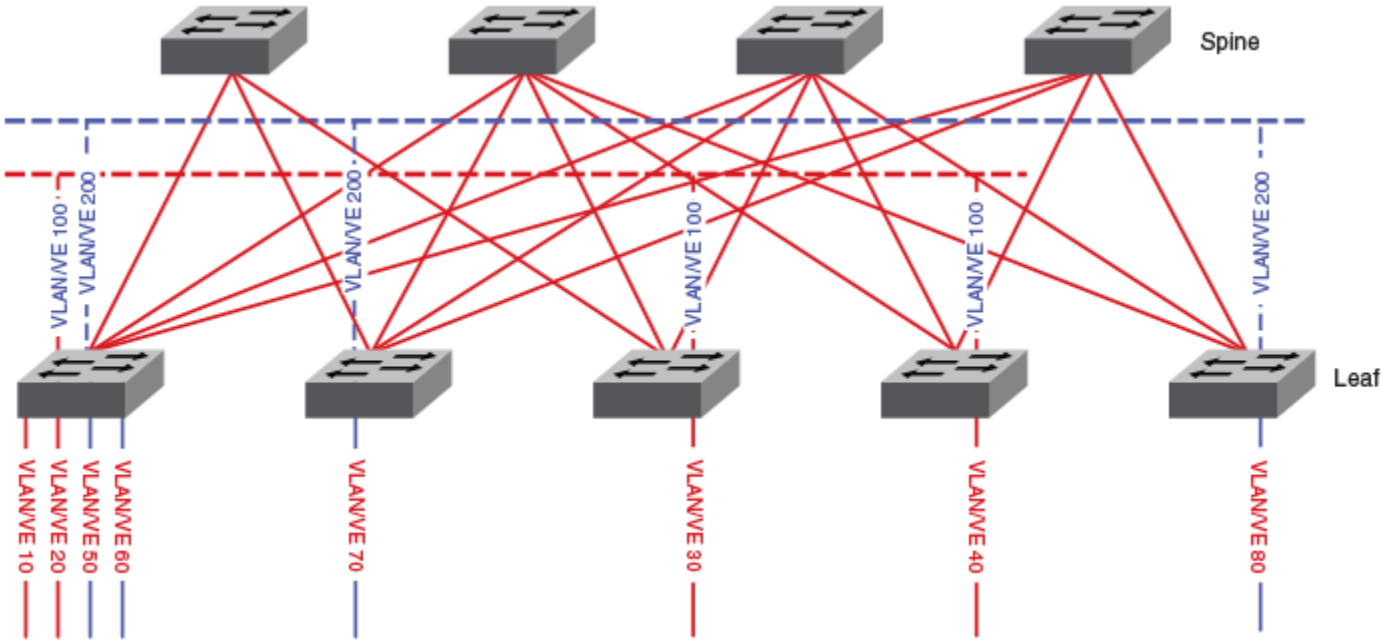
VRFs can be configured with the following features:

- Virtual network identifiers (VNIs): Layer 3 VNIs can be added and removed for a VRF instance. The VNI configuration under a VRF instance tells BGP to include the Layer 3 VNI and its associated MAC address in EVPN updates.
- Route distinguisher (RD): Unique route distinguishers are assigned for a VRF instance. RDs are manually configured for a VRF instance using the **rd (VRF)** command.
- Route targets: The route target (RT) extended community attribute, enabling logical grouping of EVPN routes that can be imported or exported, can be specified for a VRF instance. Route targets are manually configured for a specific VRF instance using the **route-target (VRF)** command. BGP EVPN uses these route targets to control the exchange of EVPN routes.
- Route filters: Filtering based on route-map policies can be added to the EVPN IPv4 and IPv6 prefix routes that are imported to or exported from VRFs. Refer to "BGP VRF route filters" in the "BGP4" and "BGP4+" chapters in this document.

For more information on the EVPN commands, refer to the *Command Reference*.

The following figure illustrates Multi-VRF topology using BGP EVPN. In this figure VRF red has VLAN 10, 20, 30, and 40. VLAN 100 is the Layer 3 VNI for VRF red. VRF blue has VLAN 50, 60, 70, and 80. VLAN 200 is the Layer 3 VNI for VRF blue.

FIGURE 13 Multi-VRF for BGP EVPN



The leaking of IPv4 and IPv6 prefix routes across VRFs is allowed using the BGP-EVPN control plane across leaf nodes. If you configure route-targets of the source VRF as import route-targets, routes originated at a leaf node in a given VRF can be imported into one or more VRFs by a node. Routes from a given VRF, including MAC IP (ARP) routes that were converted to /32 prefix routes and installed into the RIB, can be leaked into multiple VRFs. Refer to the **route-target (VRF)** command in the *Command Reference* for more information.

EVPN prefix route exchange and Multi-VRF support

In order to exchange prefixes in a VRF, end-to-end routing protocol adjacency in that VRF is required between the routers hosting a given tenant.

An EVPN prefix route eliminates such a limitation, and like an IP VPN it can accumulate routes across all VRFs in BGP and allow the per-VRF import of prefixes by routers based on the route targets. EVPN prefix route exchange provides the same functionality over any type of overlay encapsulation type supported by EVPN. (Refer to BGP EVPN control plane on page 24.)

Prefix route exchange is supported over all types of BGP EVPN encapsulation and peering—VXLAN and NSH. Each tenant VRF from which prefixes are required to be exported into EVPN or imported from EVPN, the following set of configurations is required:

1. Configure VRF import/export route targets for EVPN.
2. Select an Integrated Routing and Bridging (IRB) interface in the VRF for routing, by means of the **evpn irb ve** command, as in the following example.

```
device# configure terminal
device(config)# vrf myvrf
device(config-vrf-myvrf)# evpn irb ve 10
```

The following configuration example shows VRF RD and RT configurations in IPv4 and IPv6 address-families for both the import and export of prefixes from and into EVPN, respectively.

```
vrf red
 rd 100:1
    evpn irb ve 10
 address-family ipv4 unicast
     route-target export 100:100 evpn
     route-target import 100:100 evpn
!
 address-family ipv6 unicast
  route-target export 100:200 evpn
     route-target import 100:200 evpn
```

The IRB interface is the VE interface that is used for routing after tunnel termination. The IRB interface must belong to the tenant VRF and be administratively up. It is not necessary to configure an IP address on the IRB interface, as in the following example configuration.

```
interface ve 10
 vrf forwarding red
 no shutdown
```

Depending on the encapsulation, appropriate attributes for the overlay are used in the EVPN prefix route advertisement.

TABLE 8 Overlay attributes for EVPN prefix route advertisement

Attribute	Description
Route Distinguisher	Configured under IP VRF
ESI	Set to 0.
IP Prefix	Gateway or tenant prefix in VRF.
Gateway IP	Set to 0.

Export route targets are attached to prefixes exported into EVPN. Remote end routers compare configured IP VRF RTs against RTs in the route, and in case of a match, prefix routes are imported into EVPN and then into IP VRF tables. In the case of RTs in a prefix route match with multiple VRFs, routes from the EVPN table are imported into each matching VRF.

After routes are imported into IP VRF, the next-hop resolution of the prefix route is not performed in the VRF. Instead, an overlay next-hop is used and the prefix route is programmed with the exit interface as a tunnel.

In the case of VXLAN and NSH, the advertising router attaches the IRB interface's MAC address as part of the BGP router's MAC extended community in the prefix route. This MAC address is used as the next-hop MAC by the remote routers. In the forwarding plane, after tunnel decapsulation, routing on the inner frame is performed because the outer DA MAC is the IRB interface MAC.

Symmetric vs. asymmetric routing

Depending on how a VLAN/BD is extended on the leaf nodes, symmetric or asymmetric routing may be observed. In the case of symmetric routing, routing on a VLAN/BD takes place only at a single router/leaf in the network. On the other hand, when the same VLAN/BD is extended on multiple routers, in the presence of distributed anycast gateway, each leaf node performs routing for traffic entering from local edge ports, causing routing to take place asymmetrically depending on the direction of traffic.

Import/export route-map filtering

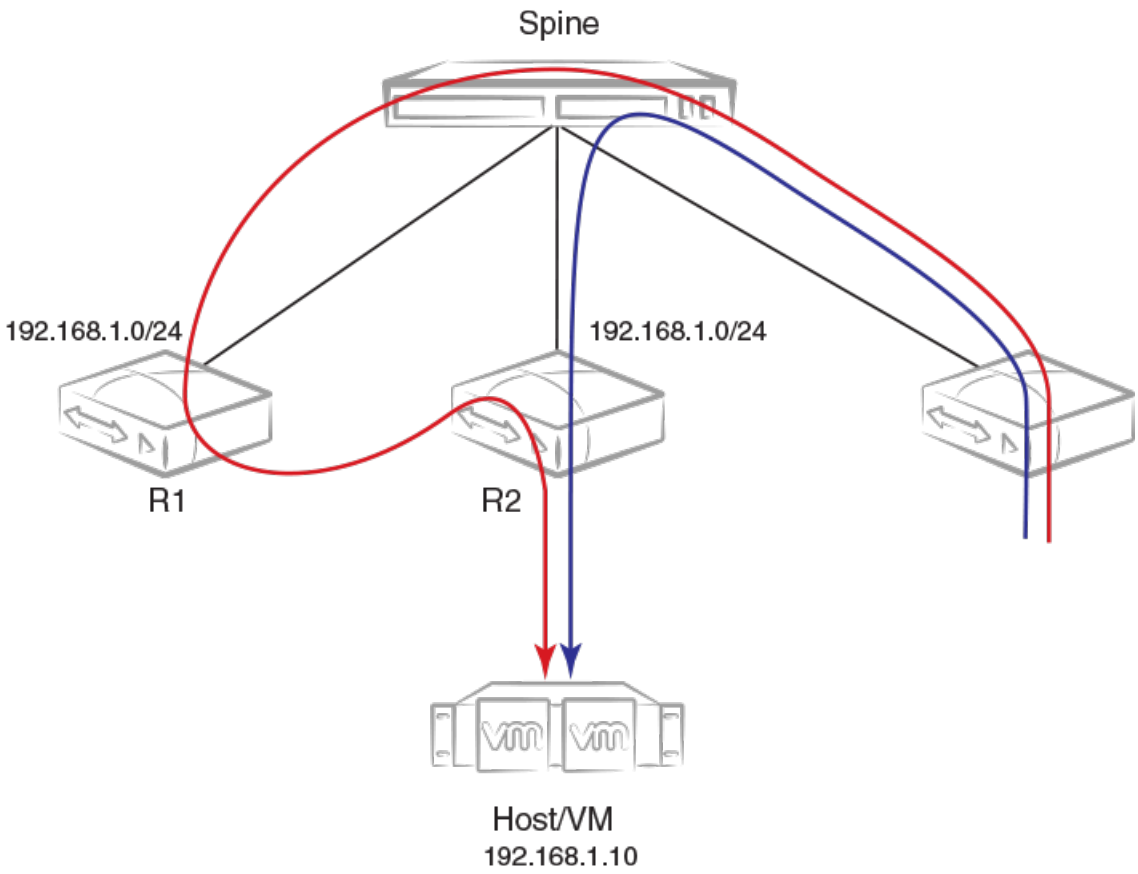
Route-map-based filtering of routes imported into IP VRF or exported into EVPN is supported. An import or export route map can be applied under IPv4 or IPv6 address family under VRF configuration, as shown in the following example.

```
device(vrf-red-ipv4-unicast)# import map my-import-filter evpn
device(vrf-red-ipv4-unicast)# export map my-export-filter evpn
```

Conversion of MACIP to host route for avoiding traffic trombone

In a typical IP Fabric network where distributed anycast gateway is configured on multiple leaf nodes, the same connected prefix is advertised by multiple nodes. A leaf node that does not extend the same VLAN/BD, but has extended the VRF, imports these prefixes, forms ECMP, and load-balances the traffic across the tunnels. Because of ECMP, traffic may take a suboptimal path as shown in the following figure.

FIGURE 14 Avoiding suboptimal paths by converting MACIP routes to host routes



In order to avoid the suboptimal forwarding, the router extending the tenant VRF accepts MACIP routes matching the IP VRF RT in the EVPN table and converts those MACIP routes to IPv4 or IPv6 /32 or /128 prefix routes and imports them into the IP VRF table. These prefixes undergo conversational ARP behavior and are pushed into hardware when an active flow is detected.

Anycast/VRRP IPv4 and IPv6 prefix routes are advertised with BGP default gateway extended community and are installed as a BGP TRAP on the remote leaf nodes for achieving conversation-based host-route programming.

EVPN next-hop resolution and tunnel EVI membership

The BGP EVPN next-hop is resolved on the basis of the encapsulation type for BGP encapsulation "extended community" that is attached to the route.

EVPN routes are filtered if the encapsulation type in the route differs from the encapsulation type configured for the neighbor.

The following table summarizes the next-hop that is set in the EVPN MP_REACH Network Layer Reachability Information (NLRI) in BGP updates originated by a router irrespective of the eBGP/iBGP peering type.

TABLE 9 Next-hop set in the MP_REACH NLRI for BGP

Field	Description
VXLAN	Source VTEP IP address
NSH	BGP peering IP address

When there are intermediate BGP routers between the service end points, and these routers do not participate in tunnel termination and re-origination without routing in the overlay, BGP neighbors on the intermediate routers should be configured to keep the next-hop unchanged. This applies to all types of tunnels, including VXLAN and NSH.

Support for IPv6 overlay next-hops is not available. The next-hop value in the EVPN routes is the IPv4 address, irrespective of whether IPv4 or IPv6 BGP peering is used.

The edge/leaf router performs next-hop resolution according to the encapsulation. For VXLAN, if VTEP discovery is enabled the VXLAN tunnels are automatically created and destroyed. NSH tunnels are not discovered. Instead, they are established according to the MCT cluster configuration.

The inclusive multicast route is originated for each VLAN/BD configured under EVPN, and this route triggers the extension of a corresponding VLAN/BD on the tunnel at the remote end. Import rules for inclusive multicast routes are discussed in "BGP routing tables". Inclusive-multicast routes are required to carry the P-Multicast Service Interface (PMSI) tunnel attribute (RFC 6514 Sec. 5) in the path-attribute to signal the tunnel attributes. PMSI tunnel attribute fields are populated as shown in the following table.

TABLE 10 PMSI tunnel attribute fields and descriptions

Tunnel Type	Tunnel Identifier
Tunnel type is always “Ingress Replication” (value 6) for all types of tunnels. No other type of tunnel is supported currently.	This field carries the source VTEP IP address for VXLAN (the BGP peering IP address for NSH).

In the case of LVTEP, both of the MCT cluster members advertise inclusive-multicast routes with different router-ids but same next-hop. The VLAN/BD membership of the tunnel is removed only when both of the advertising routers withdraw their inclusive-multicast routes. Therefore, consistent VLAN/BD configuration is required in an MCT cluster.

There is no use case for aliasing in the case of NSH.

Static anycast gateway overview

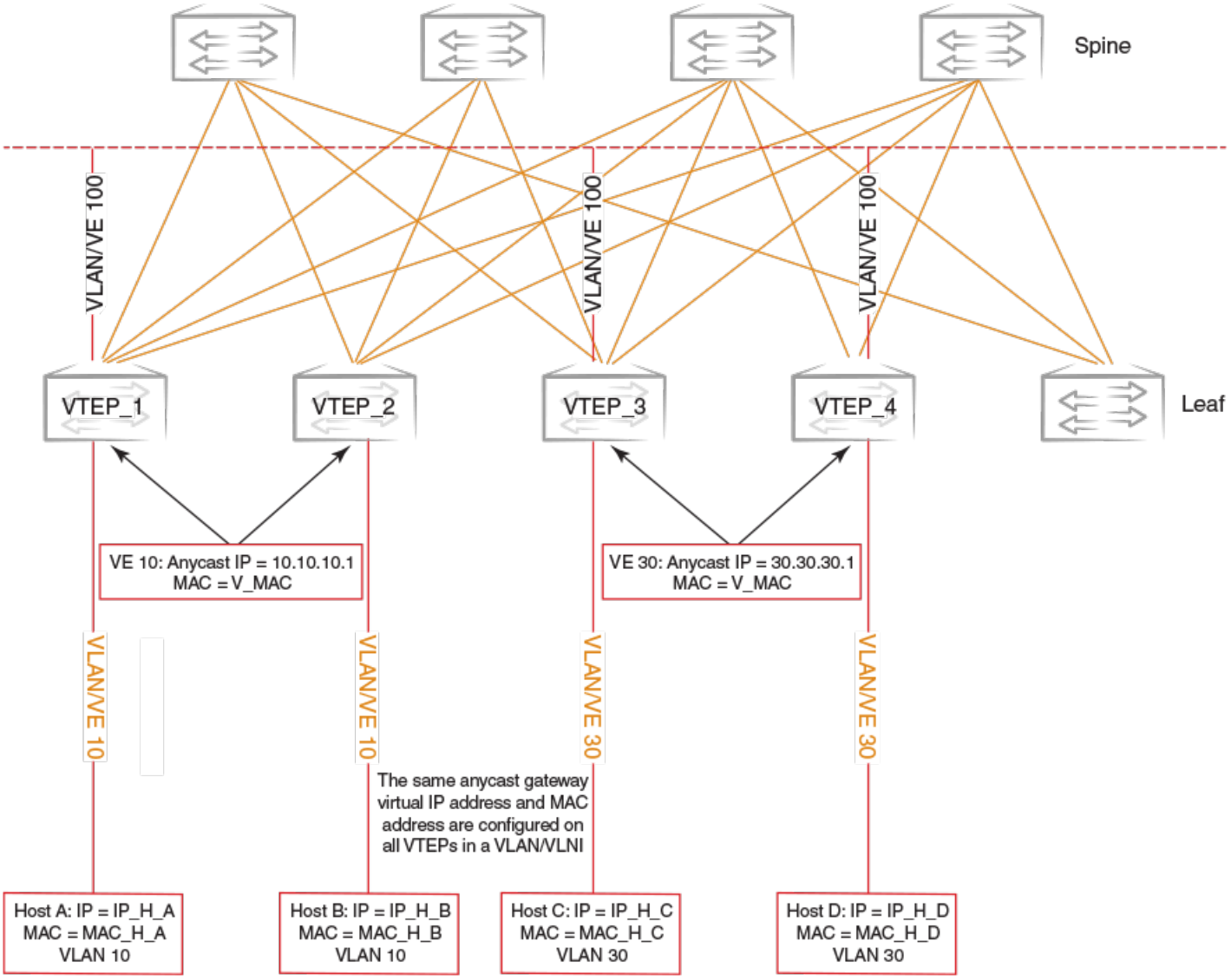
Static anycast gateway functionality provides support for seamless virtual machine (VM) mobility across the leaf switches in the IP Fabrics deployments.

Hosts attached to the leaf switches are configured with a default gateway that typically has the IP address of the leaf switch in the VLAN facing the host. If this host moves to another leaf switch, the host has to be reconfigured with the IP address of the new leaf switch to which it is now connected as its default gateway.

To make seamless this movement of host from one leaf switch to another, all the leaf switches are configured with the same anycast IP address and the associated virtual MAC (VMAC, or anycast MAC) address for a given VLAN/BD. This allows any leaf switch to behave as the default gateway for the host and allows for the most optimal forwarding behavior.

The ingress leaf switch recognizes the VMAC address as its own MAC address and does Layer 3 forwarding. Refer to the following diagram.

FIGURE 15 Static anycast gateway



For VLAN 10, all leaf switches are configured with the same MAC and IP address. When the host sends an ARP request for the gateway IP address on VLAN 10, the ingress leaf switch intercepts the ARP request and responds with the VMAC address that is associated with anycast IP address. This behavior is controlled on a per VE-interface basis.

ATTENTION

Static anycast gateway is recommended only in presence of a BGP EVPN control plane.

In order to configure static anycast gateway, configure the virtual MAC address first by specifying the default MAC address, by means of the **ip anycast-gateway-mac** or **ipv6 anycast-gateway-mac** commands, as appropriate, or an arbitrary unicast MAC address. The default MAC address values are 02e0.5200.0100 and 02e0.5200.0200 for IPv4 and IPv6, respectively. The following is an example global configuration.

```
device(config)# ip anycast-gateway-mac default-mac
device(config)# ipv6 anycast-gateway-mac default-mac
```

```
device(config)# ip anycast-gateway-mac 0000.0101.0101
device(config)# ipv6 anycast-gateway-mac 0000.0101.0102
```

The anycast gateway IP address can be configured under the VE interface for a VLAN/BD, as in the following example.

```
device(config)# vlan 100
device(config-vlan-100)# router-interface ve 100
device(config-vlan-100)# int ve 100
device(config-if-Ve-100)# ip anycast-address 100.0.0.1/24
device(config-if-Ve-100)# ipv6 anycast-address 1000::1/64
```

The **show ip anycast-gateway** and **show ipv6 anycast gateway** commands display the configuration.

```
device# show ip anycast-gateway
Gateway mac: 02e0.5200.0100
Ve10     1.1.1.0/24    Inactive (Interface Down)
Ve100    100.0.0.1/24  Active

device# show ipv6 anycast-gateway
Gateway mac: 02e0.5200.0200
Ve100    1000::1/64    Active
```

ATTENTION

It is recommended that ARP suppression be configured on the VLAN/BD if static anycast gateway is configured on a corresponding VE interface. Otherwise, duplicate ARP responses to the gateway IP address will occur.

ARP and ND scaling enhancements

This section discusses Address Resolution Protocol (ARP) and Neighbor Discovery (ND) features and illustrates example configurations.

ARP and Neighbor Discovery

When forwarding traffic, a device needs to know the destination's MAC address, because each IP packet is encapsulated in an ethernet frame. The MAC address is needed not only for the packet's final destination but also for a next hop towards the destination.

The technology by which a device gets the MAC address varies between IPv4 and IPv6, as follows:

- IPv4: Address Resolution Protocol (ARP)
- IPv6: Neighbor Discovery (ND)

IPv4 traffic

When the destination's IP address is known, to get the MAC address, a device first searches its ARP cache. A match for the IP address supplies the corresponding MAC address. Otherwise, the device broadcasts an ARP request. The network devices receive such ARP requests, and the host with a matching IP address sends an ARP reply that includes its MAC address.

IPv6 traffic

When the destination's IPv6 address is known, to get the MAC address, a device first searches its neighbor cache. A match for the IPv6 address supplies the corresponding MAC address. Otherwise, the device broadcasts a neighbor solicitation (NS) request. The network devices receive the NS request, and the host with a matching IPv6 address sends a neighbor advertisement (NA) reply that includes its MAC address.

ARP and ND scaling enhancements

In many network configurations, ARP and Neighbor Discovery (ND) traffic and caching consume significant resources, which can be optimized.

ARP and ND suppression

In a data center fabric, the ARP and ND suppression features can help reduce ARP and ND control traffic.

NOTE
This feature is supported on VLANs and bridge domains (BDs).

The default scenario (disablement of ARP and ND suppression) can lead to excess control traffic:

1. A device needs to forward traffic to an IP address, but the corresponding MAC address is not in its ARP or ND cache.
2. The device broadcasts an ARP or ND request throughout the IP Fabric.

When ARP and ND suppression are enabled, excess control traffic is reduced:

1. A device needs to forward traffic to an IP address, but the corresponding MAC address is not in its ARP or ND cache.
2. The device broadcasts an ARP or ND request.
3. The leaf BGP EVPN control plane intercepts the request and looks for a match in its local cache.
   - If there is a match, the control plane responds to the device (rather than broadcasting the original request to the entire IP Fabric).
   - Only if there is no match, does the leaf control plane broadcast the request to the entire IP Fabric.

   NOTE
   When the static anycast gateway feature is enabled in an IP Fabric, to prevent ARP/ND broadcast across the network, the user should enable ARP/ND suppression—disabled by default—on the respective VLANs or BDs.

Enabling ARP and ND suppression on a VLAN

Use this procedure to enable and disable ARP and ND suppression on a VLAN.

1. Enter **configure terminal** to access global configuration mode.

   ```
   device# configure terminal
   ```

2. Enter the **vlan** command to access VLAN configuration mode.

   ```
   device(config)# vlan 110
   ```

3. To enable ARP suppression, enter **suppress-arp**.

   ```
   device(config-Vlan-110)# suppress-arp
   ```

   To disable ARP suppression, enter **no suppress-arp**.

4. To enable ND suppression, enter **suppress-nd**.

   ```
   device(config-Vlan-110)# suppress-nd
   ```

   To disable ND suppression, enter **no suppress-nd**.

The following example enables ARP suppression on VLAN 110.

```
device# configure terminal
device(config)# vlan 110
device(config-Vlan-110)# suppress-arp
```

Enabling and disabling ARP learning

Use this procedure to enable ARP learning not only locally, but from all ARP requests. ARP learning decreases the time needed to populate the ARP cache.

1. Enter **configure terminal** to access global configuration mode.

   ```
   device# configure terminal
   ```

2. Enter the **interface ve** command to access VE configuration mode.

   ```
   device(config)# interface ve 110
   ```

3. To enable fabric learning, enter the **ip arp learn-any** command.

   ```
   device(config-ve-110)# ip arp learn-any
   ```

   To disable fabric learning, enter the **no ip arp learn-any** command.

ARP and ND suppression show and clear commands

There is a full range of ARP and ND suppression **show** and **clear** commands, listed here with descriptions.

TABLE 11 ARP and ND suppression show commands in the *Command Reference*

Command	Description
show ip arp suppression-cache	Displays IPv4 ARP suppression information.
show ip arp suppression-statistics	Displays IPv4 ARP suppression statistics.
show ip arp suppression-status	Displays the IPv4 ARP suppression status.
show ipv6 nd suppression-cache	Displays IPv6 ND suppression information.
show ipv6 nd suppression-statistics	Displays IPv6 ND suppression statistics.
show ipv6 nd suppression-status	Displays the IPv6 ND suppression status.

TABLE 12 ARP and ND suppression clear commands in the *Command Reference*

Command	Description
clear ip arp suppression-cache	Clears the IPv4 ARP suppression cache. You can also clear the cache for a specified VLAN or bridge domain.
clear ip arp suppression-statistics	Clears the IPv4 ARP suppression statistical information. You can also clear statistics for a specified VLAN or bridge domain.
clear ipv6 nd suppression-cache	Clears the IPv6 ND suppression cache. You can also clear the cache for a specified VLAN or bridge domain.
clear ipv6 nd suppression-statistics	Clears the IPv6 ND suppression statistical information. You can also clear statistics for a specified VLAN or bridge domain.

Conversational ARP and ND

Conversational ARP and ND reduce the number of cached ARP and ND entries by programming only active flows into the forwarding plane. This feature helps to optimize utilization of hardware resources.

In many use-case scenarios—especially in an IP Fabric—there are software requirements for ARP and ND entries beyond the hardware capacity. Conversational ARP and ND limit storage-in-hardware to active ARP and ND entries; aged-out entries are deleted automatically.

By default, the aging-out threshold is 300 seconds. (You can change the threshold to any integer value from 60 through 100,000 seconds, either before or during enablement.) Any entry that does not have at least one conversation before aging-out is deleted from the cache. Each conversation restarts the clock for that entry.

However, aging-out is also influenced by the enablement and disablement cycle:

1. Under conversational ARP and ND—enabled by default—a fast-aging policy of 30 seconds (not configurable) applies to all entries in the ARP and ND caches.
2. Upon disablement, the conversational ARP and ND timers no longer apply. All current entries become permanent as do all new entries.

Static ARPs and NDs are not subject to conversational behavior.

Enabling and disabling conversational ARP and ND

Conversational ARP and ND can reduce the number of cached ARP and ND entries, optimizing utilization of hardware resources.

1. Enter **configure terminal** to access global configuration mode.

```
device# configure terminal
```

2. To specify an aging-time value other than the default 300 seconds, enter the **host-table aging-time conversational** command.

```
device(config)# host-table aging-time conversational 600
```

   To restore the default aging-time value of 300 seconds, enter the **no host-table aging-time conversational** command.

3. To enable conversational ARP and ND, enter the **host-table aging-mode conversational** command.

```
device(config)# host-table aging-mode conversational
```

   To disable conversational ARP and ND, enter the **no host-table aging-mode conversational** command.

The following example implements conversational ARP and ND. The current aging-time value applies.

```
device# configure terminal
device(config)# host-table aging-mode conversational
```

Conversational ARP and ND show and clear commands

Conversational ARP and ND **show** and **clear** commands are listed here with descriptions.

TABLE 13 Conversational ARP and ND show commands in the *Command Reference*

Command	Description
show arp	Displays the ARP cache.
show ipv6 neighbor	Displays IPv6 neighbor information.

TABLE 14 Conversational ARP and ND clear commands in the *Command Reference*

Command	Description
clear arp	Clears the local ARP cache and then resends ARP requests to the local hosts.
clear arp no-refresh	Clears the ARP cache, without resending ARP requests to the local hosts.
clear ipv6 neighbor	Clears the IPv6 ND cache and then resends ND requests to the local hosts.
clear ipv6 neighbor no-refresh	Clears the ND cache, without resending ND requests to the local hosts.

ARP/ND (MACIP) route exchange

ARP/ND addresses learned on a VLAN/BD are added to EVPN and are automatically exported into BGP.

ARP/ND routes are referred as MACIP (type 2) in RFC 7432. The following types of ARP/ND routes are exported into BGP:

- Dynamically learned ARP/ND routes on VE interfaces with a local egress interface
- Statically configured ARP/ND routes on VE interfaces with a local egress interface
- IPv4/IPv6 addresses configured on VE interfaces
- Virtual IPv4/IPv6 addresses configured by means of VRRP/anycast gateway

If ARP/ND is learned with the egress interface as an MCT client interface, the corresponding ESI is attached to the MACIP route.

The resolution of ARP/ND routes received from BGP EVPN is based on the resolution of the corresponding MAC route. In addition, MAC route movement triggers the movement of MACIP routes. ARP/ND routes are withdrawn from BGP when MAC resolution changes from the local egress interface to a tunnel.

Statically configured ARP/ND entries carry a sticky flag in the MAC mobility extended community and are installed as static ARP/ND routes on the routers importing the routes. When an ARP/ND entry is sticky, the binding of the host address does not change on the basis of a local learn event.

The following table describes the MACIP route fields.

TABLE 15 MACIP route fields and descriptions

Field	Description
Route Distinguisher	Either an auto or a manual RD value is used, dependding on the VLAN/BD configuration under EVPN.
ESI	If the MAC address is learned on an MCT client interface, the ESI of the client interface is present. Otherwise it is 0.
Ethernet Tag	This is field is 0.
MAC address	MAC address associated with the ARP/ND.
IP address	IPv4/IPv6 host address.

If ARP/ND is learned on a nondefault VRF, and the VRF is configured to export routes into EVPN, both IP VRF route targets (RTs) and VLAN/BD RTs are attached.

VRRP and anycast gateway addresses are advertised with BGP default gateway extended community, and these are used for logging errors when a MACIP route is imported but te same anycast IP address is not configured. All EVPN MACIP routes are held in the ARP suppression cache, and this database is used for ARP suppression.

ARP suppression

When ARP/ND binding for a host address is known in the control plane, the ARP suppression feature allows a router to respond to the ARP/ND requests received on edge ports on the basis of the control plane information, instead of flooding the ARP/ND requests in the overlay network. This may help reduce a significant amount of flooded traffic in the overlay.

ARP/ND suppression can be enabled individually on a VLAN/BD, as shown in the following configuration example.

```
device(config)# vlan 10
device(config-vlan-10)# suppress-arp
device(config-vlan-10)# suppress-nd
device(config)# bridge-domain 10
device(config-bridge-domain-10)# suppress-arp
device(config-bridge-domain-10)# suppress-nd
```

By default, ARP/ND suppression is disabled.

Conversational ARP learning

In a typical data-center deployment, the number of tenants may easily exceed the ARP/ND scale numbers in forwarding plane. However, traffic for only a subset of pairs of tenants would flow through a given node. Conversation-based ARP aging provides a solution to this problem by removing ARP entries from the forwarding plane depending on their activity in the hardware.

Conversational ARP is enabled by default on SLX-S platforms. The default value of time after which an ARP entry, after being inactive, is purged from the hardware is 300 seconds.

Layer 2 (MAC) route exchange

All dynamic MAC addresses learned on VLANs/BDs and added to the EVPN configuration are exported to BGP EVPN automatically.

Routes are imported from the remote peers according to the route-target match. The fields in the MAC routes are populated as shown in the following table.

TABLE 16 MAC route fields and descriptions

Field	Description
Route Distinguisher	Either auto or manual RD value is used, depending on the VLAN/BD configuration under EVPN.
ESI	In case the MAC address is learned on an MCT client interface, the ESI of the client interface is indicated. Otherwise it is 0.
Ethernet Tag	This field is 0.
MAC address	The static or dynamic MAC address is learned locally.
IP address	For MAC routes the IP address is absent.

BGP MAC mobility extended community is attached to the route to carry a sticky flag and sequence number. Static MAC addresses configured on the system are advertised by BGP with a sticky flag. Routers importing a MAC route with a sticky flag install it as static, and no MAC movement is allowed. For BGP best path selection, MAC routes with a sticky flag are preferred over routes without a sticky flag, irrespective of the sequence number.

The output of the show mac-address-table command is enhanced to show the remote VTEP IP address and route type as "EVPN" for BGP-learned MAC addresses over VXLAN, as in the following example.

```
device# show mac-address-table
Type Code - CL:Cluster Local MAC   CCL:Cluster Client Local MAC
            CR:Cluster Remote MAC  CCR:Cluster Client Remote MAC
```

```
VlanId/BDId   Mac-address       Type        State      Ports/LIF/PW/Tunnel
100 (V)       0000.0164.0100    Static      Inactive   Eth 0/31
101 (V)       0000.0164.0101    Static      Inactive   Eth 0/31
10 (V)        0000.0164.0010    Static      Active     Eth 0/31
10 (V)        0000.0191.0010    EVPN        Active     Tu 61441 (192.168.32.10)
10 (V)        0001.0191.0010    EVPN        Active     Tu 61441 (192.168.32.10)
10 (V)        0001.0221.0010    EVPN        Active     Tu 61441 (192.168.32.10)
```

MAC move detection and dampening

A host moving from one port to another port on the same router is not considered a move. However, when the host moves from one router to another router, the MAC address undergoes a MAC move process. The router where MAC address is learned locally prefers the local MAC and advertises that MAC to other routers in the network. If the MAC address is already present in BGP, the sequence number in the MAC mobility extended community is incremented in the route being advertisement. A MAC route without MAC mobility extended community implicitly means the sequence number is 0. A router receiving a MAC route from a remote peer, with sequence number greater than that of the locally advertised route, prefers the remote route and withdraws the local one.

MCT and LVTEP are two special cases in which a MAC move is not triggered. If the same MAC address is present in the BGP table, and is learned from the same next-hop or with the same ESI, the MAC route is advertised with the same sequence number that is present in the existing route.

The frequent movement of a MAC address from one router to another router causes unnecessary churn in network and is an indication of a malicious host or loop. When the number of moves for a given MAC address in a specified period of time (the default is 3 sec) exceeds the specified value (the default 5), the MAC route is dampened. (The EVPN MAC route dampening behavior differs from BGP route flap dampening as specified in RFC 2439.) When a MAC route is dampened, the local route is marked as "best" and present in the forwarding tables. Best-route selection based on sequence number is stopped until corrective action is taken. Default parameters of MAC route dampening can be modified by means of the duplicate-mac-timer command under EVPN configuration mode, as in the following example.

```
device(config-evpn-default)# duplicate-mac-timer 5 max-count 3
```

Automatic restoration of dampened routes

Per BGP-EVPN RFC 7432, once a MAC/IP route is dampened because of frequent moves, manual intervention is required to restore the route. According to Section 15.1, "The PE MUST alert the operator and stop sending and processing any BGP MAC/IP Advertisement routes for that MAC address until a corrective action is taken by the operator. "

A major drawback of this approach is that the route remains dampened and no processing of the updates is performed until the dampening state of the route is cleared manually. It may happen that after the initial frequent flap of the route, either one of the VMs goes away or the duplicate address situation is resolved. However, the route remains dampened until the network administrator intervenes. The automatic restoration of the route is desired in this case.

The following approach is taken to restore the dampened route. When BGP detects that only one source of the route (that is, all NLRIs are received from same next-hop) has remained and the route is dampened, the route is added to timer list, which is processed after 5 minutes. While the 5-minute timer is expiring, if a second source of the route is seen again, the route is removed from the timer list and remains dampened. After the timer expires, the dampening state of the route is cleared and the route is restored after best-path selection.

Conversational MAC

Conversational MAC over VXLAN tunnels is supported. By default, conversational MAC learning is disabled. It can be enabled by means of the **mac-address-table learning-mode conversational** command.

High availability

BGP EVPN graceful restart (GR) helper is supported, by means of the **graceful-restart (BGP EVPN)** command. However, BGP EVPN GR router restart is not supported.

Standards conformance and RFC support for BGP EVPN

The following table lists the IETF RFCs and other draft documents that support the implementation of BGP EVPN in support of Extreme IP Fabrics.

TABLE 17 IETF BGP EVPN RFCs and other draft documents supporting Extreme IP Fabrics

Document	URL	Description
"A Network Virtualization Overlay Solution using EVPN"	https://tools.ietf.org/html/draft-ietf-bess-evpn-overlay-01	Describes how Ethernet VPN (EVPN) can be used as an Network Virtualization Overlay (NVO) solution and explores the various tunnel encapsulation options over IP and their impact on the EVPN control-plane and procedures.
"Integrated Routing and Bridging in EVPN"	https://tools.ietf.org/html/draft-ietf-bess-evpn-inter-subnet-forwarding-00	Describes an extensible and flexible multi-homing VPN solution for intra-subnet connectivity among hosts/VMs over an MPLS/IP network.
"IP Prefix Advertisement in EVPN"	https://tools.ietf.org/html/draft-ietf-bess-evpn-prefix-advertisement-02	Defines a new EVPN route type for the advertisement of IP prefixes and explains some use-case examples where this new route- type is used.

Configuring a Extreme IP Fabric with Optional BGP EVPN Overlay

Configuration overview

This chapter provides the steps to set up a basic Extreme IP Fabric topology at the spine and leaf tiers, as well as to configure optional multihomed servers.

NOTE

The following is not meant to represent a production deployment scenario, with a full variety of deployment options. It is meant to facilitate the understanding of the basic steps that are required.

NOTE

Extreme offers Extreme Workflow Composer, an open and programmable turnkey automation solution that enables organizations to:

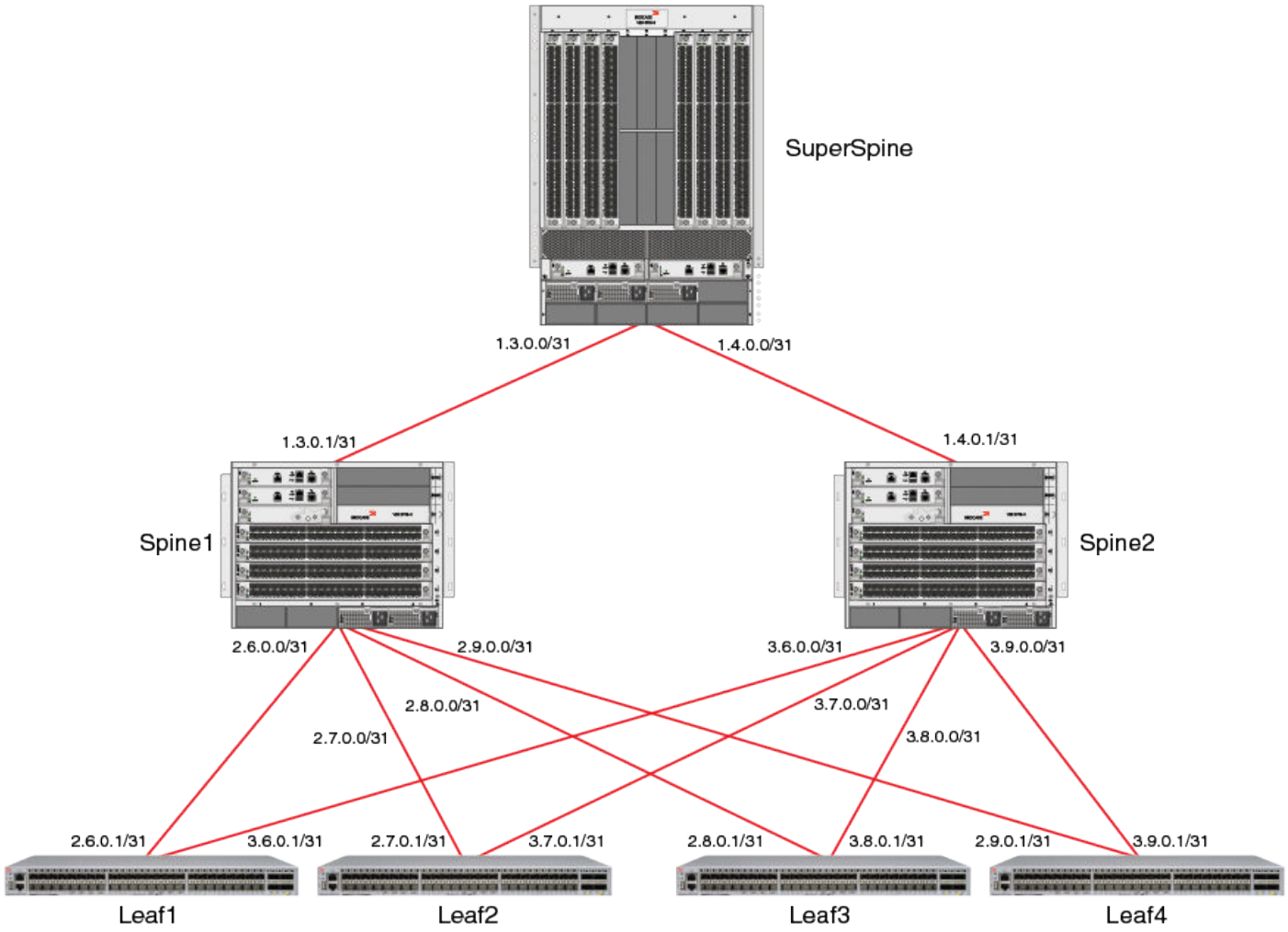
- Easily provision and validate IP Fabric infrastructure
- Customize automation to optimize IT operations

Extreme Workflow Composer's pre-packaged, Python-based automation modules can provision and validate Extreme IP Fabrics and BGP-EVPN with minimal effort, while support for open interfaces and commonly used infrastructure programming tools enables simple and straightforward customization when needed.

For more information, refer to the *Extreme Workflow Composer User Guide*.

The following illustrates the topology and subnets of the nodes used in this example configuration.

FIGURE 16 IP Fabrics topology



NOTE

Although this configuration example uses BGP EVPN to implement the overlay network, BGP EVPN is not essential to the configuration of an IP Fabric.

The example topology consists of a pair of switches at the spine tier, four switches at the leaf tier, and an optional superspine switch. The following table summarizes the basic configurations at each tier for an example switch.

TABLE 18 Basic configurations at leaf, spine, and optional superspine tiers

Leaf	Spine	Superspine
VLANs		
Static anycast gateway		
Ethernet interfaces	Ethernet interfaces	Ethernet interfaces
Loopback interfaces	Loopback interfaces	Loopback interfaces
Access/trunk ports		
Routed, IP port-channel, or optional unnumbered IP interfaces	Routed, IP port-channel, or optional unnumbered IP interfaces	Routed, IP port-channel, or optional unnumbered IP interfaces

TABLE 18 Basic configurations at leaf, spine, and optional superspine tiers (continued)

Leaf	Spine	Superspine
VRF instances (route distinguisher, VXLAN Network Identifier, route)		
Address family (IPv4, IPv6, EVPN): route target	Address family (IPv4, IPv6, EVPN)	Address family (IPv4, IPv6, EVPN)
Virtual Ethernet interfaces (VRF forwarding, anycast address)		
(Required for EVPN) IP-based management cluster (See following **Note**)		
(Required for EVPN) EVPN instance (add VLANs)		
Overlay gateway (Layer 2 extension, loopback address, map VLANs to VNIs)		
(Optional) Configure interoperability with other vendors where Extreme extended VLANs must be remapped manually to their respective VNIs		
		(Optional) OSPF routing
Configure BGP and IGP (for iBGP-based networks) on all switches, to distribute Layer 3 reachability information; if eBGP is used, configure the **neighbor allowas-in** command, to reduce load on hardware processing; optionally enable Bidirectional Forwarding Detection (BFD) for BGP neighbors, for fast recovery in case of link failure	Configure BGP and IGP (for iBGP-based networks) on all switches, to distribute Layer 3 reachability information; optionally enable Bidirectional Forwarding Detection (BFD) for BGP neighbors, for fast recovery in case of link failure	Configure BGP and IGP (for iBGP-based networks) on all switches, to distribute Layer 3 reachability information; optionally enable Bidirectional Forwarding Detection (BFD) for BGP neighbors, for fast recovery in case of link failure
	BGP neighbor (remote AS, prevent route rejection, load balancing, recursive next-hop lookups, redistribute directly connected routes)	BGP neighbor (local AS, load balancing, recursive next-hop lookups, redistribute directly connected routes)

NOTE

For the details of IP-based management cluster and a configuration example, refer to "IP-based management cluster" in the *Extreme SLX-OS Management Configuration Guide*.

Configuring the leaf switches

The following tasks configure the leaf switches.

Creating the VLANs

This task establishes the required VLANs, to support both VRFs and VXLAN Network Identifiers (VNIs).

1. In global configuration mode, create the VLANs required to support service, starting with leaf switch Leaf1.

```
Leaf1# configure terminal
Leaf1(config)# vlan 101-108,120,121-128,140,141-148,160,161-168,180
```

NOTE

VLANs 120, 140, 160, and 180 will support VRFs as VXLAN Network Identifiers (VNIs). VLANs must be created before they can be added to a switchport trunk.

2. Repeat Step 1 as appropriate for the remaining leaf switches.

3. In VLAN configuration mode, enter the **router-interface ve** command to attach (bind) a router interface to a VLAN, creating a Layer 3 interface.

```
device(config)# vlan 101
device(config-vlan-101)# router-interface ve 101
```

   Attaching a router interface to a Layer 2 VLAN ensures that the interface comes up. Repeat the above for additional VLANs as appropriate.

4. Use the **show running-config vlan** command to verify the configuration.

Configuring ARP and ND suppression

ARP suppression is required to support static anycast gateway. This task configures both ARP and Neighbor Discovery (ND) suppression, which is done in VLAN configuration mode.

1. Enter global configuration mode.

```
device# configure terminal
```

2. Specify an existing VLAN and enter VLAN configuration mode. (This example starts with VLAN 101.)

```
device(config)# interface vlan 101
```

3. To enable ARP suppression, enter the **suppress-arp** command.

```
device(config-Vlan-101)# suppress-arp
```

4. To enable ND suppression, enter the **suppress-nd** command.

```
device(config-Vlan-101)# suppress-nd
```

5. Repeat Step 2 through Step 4, as appropriate, for all remaining VLANs on which ARP or ND suppression is required.

Configuring static anycast gateway MAC addresses

This task configures static anycast gateway MAC addresses for IPv4 and IPv6 support.

It is recommend that you do not use this feature unless BGP EVPN is implemented. It is also recommended that ARP/ND suppression be enabled.

Within a dual-stack IP Fabric, the first three bytes (six digits) of a nondefault IPv4 static-anycast-gateway MAC address must be identical with those of the corresponding IPv6 MAC address. IPv4 and IPv6 addresses can also be used, as well as default MAC addresses.

1. In RBridge ID configuration mode, specify IPv4 and IPv6 static anycast gateway MAC addresses, by using the **ip anycast-gateway-mac** and **ipv6 anycast-gateway-mac** commands, respectively.

```
Leaf1(config)# ip anycast-gateway-mac 0000.abba.baba
Leaf1(config)# ipv6 anycast-gateway-mac 0000.abba.abba
```

2. Repeat Step 1 as appropriate for the remaining leaf switches. The same MAC address must be used in all cases.
3. To verify the configuration, use the **show running-config ip anycast-gateway-mac** and the **show running-config ipv6 anycast-gateway-mac** commands.

Configuring the interfaces

This task configures the physical interfaces, as well as the virtual (loopback) interfaces configured with a donor IP address in the optional IP Unnumbered address configuration. It also references an optional task where servers are dual-homed to a leaf node for high availability, as well as an optional task to configure the maximum transmission unit (MTU) to account for encapsulation overhead.

1. In interface subtype configuration mode, configure an Ethernet interface for point-to-point connectivity between leaf switch Leaf1 and the spine switches.

   This example uses both IPv4 and IPv6 addresses, with /31 and /127 networks, respectively.

   ```
   Leaf1(config)# interface ethernet 0/1
   Leaf1(conf-if-eth-0/1)# ip address 2.6.0.1/31
   Leaf1(conf-if-eth-0/1)# ipv6 address 1000:2:6::1/127
   Leaf1(conf-if-eth-0/1)# no shut
   ```

   Use the **description** keyword to facilitate management and maintenance of the network.

   Repeat the preceding configuration for the second Ethernet interface.

   ```
   interface ethernet 0/5
    ip address 3.6.0.1/31
    ipv6 address 1000:3:6::1/12
    no shut
   ```

2. In global configuration mode, configure a loopback interface.

   The loopback interface provides the source of the donor addresses used for the IP unnumbered feature. This example uses both IPv4 and IPv6 addresses, with /32 and /128 networks, respectively.

   ```
   Leaf1(config)# interface loopback 1
   Leaf1(config-Loopback-1)# ip address 6.0.0.6/32
   Leaf1(config-Loopback-1)# ipv6 address 1000:6::6/128
   Leaf1(config-Loopback-1)# no shut
   ```

3. (Optional) You can configure unnumbered IPv4 and IPv6 addresses, with an appropriate loopback address, as follows.

   ```
   Leaf1(config)# interface ethernet 0/1
   Leaf1(conf-if-eth-0/1)# ip unnumbered loopback 1
   Leaf1(conf-if-eth-0/1)# ipv6 unnumbered loopback 1
   Leaf1(conf-if-eth-0/1)# no shut
   ```

4. (Optional) To accommodate encapsulation overhead on a SLX 9140, you can use the **mtu** command to configure a nondefault MTU both globally and on an interface, with the interface configuration taking precedence.
   - The following example configures a nondefault MTU globally.

     ```
     device# configure terminal
     device(config)# mtu 2000
     ```

   - The following example configures a nondefault MTU on an Ethernet interface.

     ```
     Leaf1(conf-if-eth-0/1)# mtu 2000
     ```

5. (Optional) To ensure that sufficient Time To Live (TTL) values are available for hello messages to establish separate neighbor adjacencies on leaf switches when eBGP is used in conjunction with the IP unnumbered feature, it is important to set the number of available eBGP paths to 2.

6. In interface subtype configuration mode, configure an Ethernet interface for switchport trunk connectivity between leaf switch Leaf1 and the spine switches.

   These trunks will carry the ranged VLANs.

```
Leaf1(config)# interface ethernet 0/16
Leaf1(conf-if-eth-0/16)# switchport
Leaf1(conf-if-eth-0/16)# switchport mode trunk
Leaf1(conf-if-eth-0/16)# switchport trunk allowed vlan add 101-108
Leaf1(conf-if-eth-0/16)# switchport trunk allowed vlan add 121-128
Leaf1(conf-if-eth-0/16)# no shut
```

   Repeat the preceding configuration for the second Ethernet switchport trunk interface.

```
Leaf1(config)# interface ethernet 0/17
Leaf1(conf-if-eth-0/17)# switchport
Leaf1(conf-if-eth-0/17)# switchport mode trunk
Leaf1(conf-if-eth-0/17)# switchport trunk allowed vlan add 141-148
Leaf1(conf-if-eth-0/17)# switchport trunk allowed vlan add 161-168
Leaf1(conf-if-eth-0/17)# no shut
```

7. Repeat Step 1 through Step 7, as appropriate, for the remaining leaf nodes.
8. To verify the configuration, use the **show running-config interface** command.

Configuring the VRF instances

This task establishes the VRF instances and applies them to virtual Ethernet (VE) interfaces.

1. Create VRF instances, and configure a route distinguisher and specify the Layer 3 VNI to be used for intersubnet forwarding.

   This task illustrates the implementation of BGP EVPN Layer 3 multitenancy where BGP EVPN is used for the overlay. BGP EVPN must be enabled to support VXLAN Network Identifiers (VNIs).

   a) Create a VRF instance, in this example "red".

```
Leaf1(config)# vrf red
Leaf1(config-vrf-red)#
```

   b) Configure a route distinguisher (RD) and VNI.

```
Leaf1(config-vrf-red)# rd 6.0.0.6:1
Leaf1(config-vrf-red)# vni 120
```

2. Enter address-family IPv4 unicast configuration mode, configure import and export route targets, and specify EVPN routes as targets.

```
Leaf1(config-vrf-red)# address-family ipv4 unicast
Leaf1(config-vrf-red-ipv4-unicast)# route-target import 3:3 evpn
Leaf1(config-vrf-red-ipv4-unicast)# route-target export 3:3 evpn
Leaf1(config-vrf-red-ipv4-unicast)# exit
Leaf1(config-vrf-red)#
```

3. Repeat the preceding configuration for IPv6.

```
Leaf1(config-vrf-red)# address-family ipv6 unicast
Leaf1(config-vrf-red-ipv6-unicast)# route-target import 3:3 evpn
Leaf1(config-vrf-red-ipv6-unicast)# route-target export 3:3 evpn
```

4. Repeat Step 1 through Step 3 for all remaining VRF instances, and return to interface subtype configuration mode.

5. In interface subtype configuration mode, configure the VE interfaces corresponding to the ranged VLANs (101-108, 121-128, 141-148, 161-168) used to support the VRF instances with anycast addresses. The anycast gateway address is used to configure the static anycast gateway MAC addresses for the hosts. This example also shows the optional configuration to support IPv6.
   a) Configure VE 101 to forward VRF "red".

```
Leaf1(config)# interface ve 101
Leaf1(config-Ve-101)# vrf forwarding red
```

   b) Configure VE 101 to support IPv4 and IPv6 static anycast gateway addresses, and enable the interface.

```
Leaf1(config-Ve-101)# ip anycast-address 101.1.1.254/24
Leaf1(config-Ve-101)# ipv6 anycast-address 101:1:1::254/64
Leaf1(config-Ve-101)# no shut
```

6. Repeat Step 5 for VE interfaces 102-108, 121-128, 141-148, 161-168 and their respective VRF instances.
7. In interface subtype configuration mode, configure the VE interfaces 120, 140, 160, 180 to support their respective VRF instances without anycast addresses. These are Layer 3 VNIs and therefore do not need IP addresses.
   a) Configure VE 120 to forward VRF "red".

```
Leaf1(config)# interface ve 120
Leaf1(config-Ve-120)# vrf forwarding red
```

   b) Enable the interface.

```
Leaf1(config-Ve-120)# no shut
```

8. To support IPv6 traffic over the L3 VNI configured above, then configure support for an IPv6 address on the interface.
   a) Enter interface subtype configuration mode and specify the VE interface used above.

```
Leaf1(config)# interface ve 120
```

   b) Enter the **ipv6 address use-link-local-only** command.

```
Leaf1(config)# ipv6 address use-link-local-only
```

9. Repeat Step 7 for VE interfaces 140, 160, and 180 to support their respective VRF instances.
10. Repeat Step 1 through Step 9 for the remaining leaf nodes, as appropriate.
11. To verify the configuration, use the **show running-config interface ve** command.

Configuring BGP routing

This task enables BGP routing and configures a variety of parameters.

ATTENTION

IPv4 neighbors are activated by default under the IPv4 address family. To negotiate only the BGP L2VPN EVPN address family on an IPv4 neighbor, you must explicitly deactivate the IPv4 unicast address family by using the **no neighbor activate** command. The BGP configuration examples in this document assume that the neighbors configured for Layer 2 EVPN carry both IPv4 and IPv6 unicast routes as well.

This example illustrates the optional redistribution of OSPF routes into BGP, to support the example spine configuration. This is required only if OSPF is used for the underlay network.

1. Enable BGP routing and enter BGP global configuration mode.

```
Leaf1(config)# router bgp
```

2. Use the **neighbor** command to configure the following attributes.
   a) Configure a local AS number.

> NOTE
> A local AS number facilitates the merging of networks by allowing a switch in an acquired network to appear to belong to the former AS. In eBGP deployments, the local AS values must be different for each node.

```
Leaf1(config-bgp-router)# local-as 1
```

   b) Configure a remote AS number for both IPv4 and IPv6 addresses to support BGP sessions to the spine.

```
Leaf1(config-bgp-router)# neighbor 2.6.0.0 remote-as 2
Leaf1(config-bgp-router)# neighbor 1000:2:6:: remote-as 2
Leaf1(config-bgp-router)# neighbor 3.6.0.0 remote-as 2
Leaf1(config-bgp-router)# neighbor 1000:3:6:: remote-as 2
```

3. (Optional) To ensure that sufficient Time To Live (TTL) values are available for hello messages to establish separate neighbor adjacencies on leaf switches when eBGP is used in conjunction with the IP unnumbered feature, it is important to set the number of allowed hops to 2, as in the following example.

```
Leaf1(config-bgp-router)# neighbor 2.6.0.0 ebgp-multihop 2
Leaf1(config-bgp-router)# neighbor 3.6.0.0 ebgp-multihop 2
```

4. Enter address-family IPv4 unicast configuration mode and configure the following attributes.
   a) Use the **neighbor allowas-in** command to disable the AS_PATH check function for routes learned from a specified location.

```
Leaf1(config-bgp-router)# address-family ipv4 unicast
Leaf1(config-bgp-ipv4u)# neighbor 2.6.0.0 allowas-in 1
Leaf1(config-bgp-ipv4u)# neighbor 3.6.0.0 allowas-in 1
```

      NOTE
      This prevents BGP from rejecting routes that contain the recipient BGP speaker's AS number, where *number* here specifies the number of times that the AS path of a received route may contain the recipient BGP speaker's AS number and still be accepted.

   b) Configure the maximum number of paths for ECMP load balancing. (The default is 1.)

```
Leaf1(config-bgp-ipv4u)# maximum-paths 8
```

   c) Enable BGP recursive next-hop lookups.

```
Leaf1(config-bgp-ipv4u)# next-hop-recursion
```

   d) Enable directly connected routes to be redistributed into BGP.

```
Leaf1(config-bgp-ipv4u)# redistribute connected
```

   e) (Optional) Enable OSPF routes to be redistributed into BGP.

```
Leaf1(config-bgp-ipv4u)# redistribute ospf
```

   f) Exit address-family IPv4 unicast configuration mode.

```
Leaf1(config-bgp-ipv4u)# exit
```

5. (Optional) Add a route map to filter only VTEP addresses.
   a) In global configuration mode, enter the **route-map** command to specify and configure a route map.

```
Leaf1(config)# route-map rm-allow-vtep-addresses permit 5
Leaf1(config-route-map-rm-allow-vtep-addresse/permit/5)# match ip address prefix-list pl-vtep-
addresses
Leaf1(config-route-map-rm-allow-vtep-addresse/permit/5)# exit
```

   b) In global configuration mode, enter the **ip prefix-list** command to specify and configure a prefix list.

```
Leaf1(config)# ip prefix-list pl-vtep-addresses
```

   c) In global configuration mode, enter the **ip access-list** command to create a standard access list and enter ACL configuration mode, then use the **seq** command to specify a loopback address.

```
device(config)# ip access-list standard stdACL
device(conf-ipacl-std)# seq 5 permit 6.0.0.6/32
```

      The above address must be a loopback address.

6. Enter address-family IPv6 unicast configuration mode and configure the following attributes.
   a) Activate the interface, enabling the exchange of information with BGP neighbors and peer groups, and use the **neighbor allowas-in** command to disable the AS_PATH check function for routes learned from a specified location.

```
Leaf1(config-bgp-router)# address-family ipv6 unicast
Leaf1(config-bgp-ipv6u)# neighbor 1000:2:6:: activate
Leaf1(config-bgp-ipv6u)# neighbor 1000:2:6:: allowas-in 1
Leaf1(config-bgp-ipv6u)# neighbor 1000:3:6:: activate
Leaf1(config-bgp-ipv6u)# neighbor 1000:3:6:: allowas-in 1
```

   b) Configure the maximum number of paths for ECMP load balancing. (The default is 1.)

```
Leaf1(config-bgp-ipv6u)# maximum-paths 8
```

   c) Enable BGP recursive next-hop lookups.

```
Leaf1(config-bgp-ipv6u)# next-hop-recursion
```

   d) Enable directly connected routes to be redistributed into BGP.

```
Leaf1(config-bgp-ipv6u)# redistribute connected
```

   e) (Optional) Enable OSPF connected routes to be redistributed into BGP, and exit to BGP global configuration mode.

```
Leaf1(config-bgp-ipv6u)# redistribute ospf
Leaf1(config-bgp-ipv6u)# exit
Leaf1(config-bgp-router)#
```

   NOTE
   This required only if OSPF is used for the underlay network.

7. (Optional) Configure neighbor Bidirectional Forwarding Detection (BFD), by means of the **neighbor bfd** command, to facilitate fast recovery in case of a link failure.

```
Leaf1(config-bgp-router)# neighbor 2.6.0.0 bfd
Leaf1(config-bgp-router)# neighbor 3.6.0.0 bfd
```

   NOTE
   For more information, refer to the "BFD" chapter in the *Extreme SLX-OS Layer 3 Routing Configuration Guide*.

8. (Optional) Configure Bidirectional Forwarding Detection (BFD) session parameters for BGP globally, by means of the **bfd interval** command.

```
device(config-bgp-router)# bfd interval 140 min-rx 125 multiplier 44
```

9. (Optional) To provide support for BFD for IP unnumbered ECMP interfaces, enable BFD and specify multipath BFD as in the following example.

```
Leaf1(config-bgp-router)# neighbor 2.6.0.0 bfd multipath
```

   Repeat as appropriate for all ECMP interfaces. Refer to BFD for IP Unnumbered ECMP Interfaces.

10. (Optional) Enter address-family Layer 2 Virtual Private Network (VPN) Ethernet VPN (EVPN) configuration mode and configure the following attributes for IPv4 addresses.
    a) Activate the interface, enabling the exchange of information with BGP neighbors and peer groups. Use the **neighbor allowas-in** command to disable the AS_PATH check function for routes learned from a specified location, and use the **neighbor encapsulation vxlan** command to enable VXLAN encapsulation.

```
Leaf1(config-bgp-router)# address-family l2vpn evpn
Leaf1(config-evpn)# neighbor 2.6.0.0 activate
Leaf1(config-evpn)# neighbor 2.6.0.0 encapsulation vxlan
Leaf1(config-evpn)# neighbor 2.6.0.0 allowas-in 1
Leaf1(config-evpn)# neighbor 3.6.0.0 activate
Leaf1(config-evpn)# neighbor 3.6.0.0 encapsulation vxlan
Leaf1(config-evpn)# neighbor 3.6.0.0 allowas-in 1
```

    NOTE
    This step is applicable only when BGP EVPN is used for the overlay.

    b) Enable BGP to send an update to an eBGP peer with the next-hop attribute unchanged.

```
Leaf1(config-evpn)# neighbor 2.6.0.0 next-hop-unchanged
Leaf1(config-evpn)# neighbor 3.6.0.0 next-hop-unchanged
```

11. In BGP router configuration mode, configure a BGP instance for IPv4 unicast address-family for VRF red and use the **redistribute connected** command to redistribute connected routes, and do the same for IPv6.

```
Leaf1(config-bgp-router)# address-family ipv4 unicast vrf red
Leaf1(config-bgp-ipv4u-vrf)# redistribute connected

Leaf1(config-bgp-router)# address-family ipv6 unicast vrf red
Leaf1(config-bgp-ipv6u-vrf)# redistribute connected
```

12. Repeat the above steps on the remaining leaf nodes as appropriate.
13. To verify the configuration, use the **show running-config router bgp** , the **show ip bgp summary** , the **show bgp evpn summary**, and the **show bfd neighbors** commands.

Configuring the overlay gateway

This task creates an overlay gateway instance and configures a variety of parameters.

1. In global configuration mode, create an overlay gateway instance and enter overlay-gateway configuration mode.

```
Leaf1(config)# overlay-gateway Leaf1
```

2. In overlay-gateway configuration mode, specify the type as **layer2-extension**.

```
Leaf1(config-overlay-gw-Leaf1)# type layer2-extension
```

3. Specify the IP interface as Loopback 1.

```
Leaf1(config-overlay-gw-Leaf1)# ip interface loopback 1
```

4. Specify the automatic mapping of VLANs to VNIs.

```
Leaf1(config-overlay-gw-Leaf1)# map vni auto
```

NOTE
This is the preferred method for ease of configuration. If the user does not want to use automatic VLAN VNI mapping, then manual VLAN-to-VNI mapping can be used. This manual mapping is required for Cisco interoperability. Refer to Configuring interoperability with other vendors on page 76.

5. Activate the overlay gateway

```
Leaf1(config-overlay-gw-Leaf1)# activate
```

6. Repeat Step 1 through Step 6, as appropriate, for the remaining leaf nodes.

Configuring the spine switches

This task configures underlay physical connectivity between the spine, leaf, and optional superspine switches, and also sets BGP neighbor and AS attributes.

1. In interface subtype configuration mode, configure an Ethernet interface for point-to-point connectivity between spine switch Spine1 and supported leaf switches.

NOTE
This example uses both IPv4 and IPv6 addresses, with /31 and /127 networks, respectively.

```
Spine1# configure terminal
Spine1(config)# interface ethernet 0/33
Spine1(conf-if-eth-0/33)# ip address 1.2.1.1/31
Spine1(conf-if-eth-0/33)# ipv6 adddress 1000:1:2:2::1/127
Spine1(conf-if-eth-0/33)# no shut
Spine1(conf-if-eth-0/33)# exit
Spine1(config)#
```

2. Configure loopback interfaces, to be the donor interfaces used for optional IP unnumbered.

NOTE
This example uses both IPv4 and IPv6 addresses, with /32 and /128 networks, respectively.

```
Spine1(config)# interface loopback 1
Spine1(config-Loopback-1)# ip address 2.0.0.2/32
Spine1(config-Loopback-1)# ipv6 address 1000:2::2/128
Spine1(config-Loopback-1)# no shut
Spine1(config-Loopback-1)# exit
Spine1(config)#
```

3. (Optional) You can configure unnumbered IPv4 and IPv6 addresses, with an appropriate loopback address, as follows.

```
Leaf1(config)# interface ethernet 0/33
Leaf1(conf-if-eth-0/33)# ip unnumbered loopback 1
Leaf1(conf-if-eth-0/33)# no shut
```

4. Repeat Step 1 or Step 2 for all Ethernet interfaces supporting the leaf switches.
5. Enable BGP routing and enter BGP global configuration mode.

```
Spine1(config)# router bgp
Spine1(config-bgp-router)#
```

6. Use the **neighbor** command to configure the following attributes.
   a) Configure a local AS number for the spine switch.

      NOTE
      A local AS number facilitates the merging of networks by allowing a switch in an acquired network to appear to belong to the former AS. In eBGP deployments, the local AS values must be different for each node.

      ```
      Spine1(config-bgp-router)# local-as 2
      ```

   b) Configure a remote AS number for both IPv4 and IPv6 addresses to support BGP sessions to the superspine.

      ```
      Spine1(config-bgp-router)# neighbor 1.2.1.0 remote-as 1
      Spine1(config-bgp-router)# neighbor 1000:1:2:1:: remote-as 1
      ```

   c) Configure remote AS numbers for both IPv4 and IPv6 addresses to support BGP sessions to the leaf switches.

      ```
      Spine1(config-bgp-router)# neighbor 2.6.0.1 remote-as 1
      Spine1(config-bgp-router)# neighbor 1000:2:6:1:: remote-as 1
      ```

   d) Repeat Step 6a through Step 6c, as appropriate, for the remaining leaf interfaces. The following shows the results of the configuration.

      ```
      neighbor 2.7.0.1 remote-as 3
      neighbor 1000:2:7::1 remote-as 3
      neighbor 2.8.0.1 remote-as 3
      neighbor 1000:2:8::1 remote-as 3
      neighbor 2.9.0.1 remote-as 3
      neighbor 1000:2:9::1 remote-as 3
      ```

   e) (Optional) To provide support for BFD for IP unnumbered ECMP interfaces, enable BFD and specify multipath BFD as in the following example.

      ```
      Leaf1(config-bgp-router)# neighbor 2.6.0.0 bfd multipath
      ```

      Repeat as appropriate for all ECMP interfaces. Refer to BFD for IP Unnumbered ECMP Interfaces.

7. Enter address-family IPv4 unicast configuration mode and configure the following attributes.
   a) (Optional) Use the **neighbor allowas-in** command to disable the AS_PATH check function for routes learned from a superspine.

      NOTE
      This prevents BGP from rejecting routes that contain the recipient BGP speaker's AS number, where *number* here specifies the number of times that the AS path of a received route may contain the recipient BGP speaker's AS number and still be accepted.

      ```
      Spine1(config-bgp-router)# address-family ipv4 unicast
      Spine1(config-bgp-ipv4u)# neighbor 1.2.1.0 allowas-in 1
      ```

   b) Configure the maximum number of paths for ECMP load balancing. (The default is 1.)

      ```
      Spine1(config-bgp-ipv4u)# maximum-paths 8
      ```

   c) Enable BGP recursive next-hop lookups.

      ```
      Spine1(config-bgp-ipv4u)# next-hop-recursion
      ```

   d) (Optional) Enable directly connected routes to be redistributed into BGP.

      ```
      Spine1(config-bgp-ipv4u)# redistribute connected
      ```

   e) (Optional) Enable OSPF routes to be redistributed into BGP.

      ```
      Spine1(config-bgp-ipv4u)# redistribute ospf
      ```

   f) Exit address-family IPv4 unicast configuration mode.

      ```
      Spine1(config-bgp-ipv4u)# exit
      Spine1(config-bgp-router)#
      ```

8. Enter address-family IPv6 unicast configuration mode and configure the following attributes.
   a) Activate the interface, enabling the exchange of information with BGP neighbors and peer groups, and optionally use the **neighbor allowas-in** command to disable the AS_PATH check function for routes learned from a superspine.

```
Spine1(config-bgp-router)# address-family ipv6 unicast
Spine1(config-bgp-ipv6u)# neighbor 1000:1:2:1:: activate
Spine1(config-bgp-ipv6u)# neighbor 1000:1:2:1:: allowas-in 1
```

   b) Configure the maximum number of paths for ECMP load balancing. (The default is 1.)

```
Spine1(config-bgp-ipv6u)# maximum-paths 8
```

   c) Enable BGP recursive next-hop lookups.

```
Spine1(config-bgp-ipv6u)# next-hop-recursion
```

   d) Enable directly connected routes to be redistributed into BGP.

```
Spine1(config-bgp-ipv6u)# redistribute connected
```

   e) Enable OSPF routes to be redistributed into BGP.

```
Spine1(config-bgp-ipv6u)# redistribute ospf
```

   f) Exit address-family IPv6 unicast configuration mode, and enter BGP configuration mode.

```
Spine1(config-bgp-ipv6u)# exit
Spine1(config-bgp-router)#
```

9. (Optional) Enable Bidirectional Forwarding Detection (BFD) for BGP neighbors, to facilitate fast recovery in case of a link failure.

```
Spine1(config-bgp-router)# neighbor 1.2.1.0 bfd
Spine1(config-bgp-router)# neighbor 1000:1:2:1:: bfd
```

Repeat Step 10 for all IPv4 and IPv6 neighbors.

10. (Optional) Enter address-family Layer 2 Virtual Private Network (VPN) Ethernet VPN (EVPN) configuration mode and configure the following attributes for IPv4 addresses.
    a) Activate the interface, enabling the exchange of information with BGP neighbors and peer groups, and use the **neighbor allowas-in** command to disable the AS_PATH check function for routes learned from a specified location.

```
Spine1(config-bgp-router)# address-family l2vpn evpn
Spine1(config-evpn)# neighbor 1.2.1.0 activate
Spine1(config-evpn)# neighbor 1.2.1.0 allowas-in 1
```

    b) Enable BGP to send an update to an eBGP peer with the next-hop attribute unchanged.

```
Spine1(config-evpn)# neighbor 1.2.1.0 next-hop-unchanged
```

    c) Repeat Step 10a through Step 10b, as appropriate, for the remaining IPv4 leaf interfaces. The following shows the results of the configuration.

```
neighbor 2.6.0.1 activate
neighbor 2.6.0.1 next-hop-unchanged
neighbor 2.7.0.1 activate
neighbor 2.7.0.1 next-hop-unchanged
neighbor 2.8.0.1 activate
neighbor 2.8.0.1 next-hop-unchanged
neighbor 2.9.0.1 activate
neighbor 2.9.0.1 next-hop-unchanged
```

11. Enter the **retain-route-target-all** under BGP EVPN address-family configuration mode to configure the route reflector (RR) to accept all route targets (RTs), preventing filtering on routes that do not match the local configuration on the spine.

```
Spine1(config-bgp-router)# address-family l2vpn evpn
Spine1(config-bgp-evpn)# retain route-target all
```

12. (Optional) Where iBGP is used, use the **neighbor update-source** command to default to the loopback address as a backup interface in case of a link failure.

```
Spine1(config-bgp-router)# neighbor 2.0.0.2 update-source loopback 1
Spine1(config-bgp-router)# neighbor 1000:2::2 update-source loopback 1
```

13. Repeat Step 1 through Step 12, as appropriate, for the remaining interfaces.
14. Repeat Step 1 through Step 13, as appropriate, for Spine2.
15. To verify the configuration, use the **show ip bgp summary** , the **show bgp evpn summary** , and the **show bfd neighbors** commands.

(Optional) Configuring a BGP EVPN instance

A BGP EVPN instance (EVI) can be configured and various commands can be executed in EVPN instance configuration mode, as shown in the following configuration example.

BGP EVPN is not essential to the configuration of an IP Fabric, and is one among a variety of overlay solutions.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **evpn** command and specify a name to configure an EVPN instance and enter EVPN instance configuration mode.

```
device(config)# evpn default
```

3. Enter the **rd auto** command to enable auto-generation of the RD value.

```
device(config-evpn-default)# rd auto
```

4. Enter the **route-target** command using the **both** and **auto** parameters to configure auto-generation of the import and export route-target community attributes globally.

```
device(config-evpn-default)# route-target both auto
```

5. Enter the **vni** command with the **add** *value* parameter to add VLANs to the EVI.

```
device(config-evpn-default)# vni add 1-100
```

   Ensure that only local VNIs are added, supporting local VLANs. With symmetric integrated routing and bridging (IRB), if a remote VNI is added that is not on the local switch, that VNI is treated as being used for Layer 2 extension and ARP entries are not programmed in hardware, resulting in potential performance problems. For example, if switch Switch-1 has only VNI 100 and another switch, Switch-2, has VNI 200, you would execute the**vlan add 100** command on Switch-1 and the **no vlan add 200** command on Switch-2.

6. (Optional) Enter the **duplicate-mac-timer** *count* command with the **max-count** *interval* parameter to set the duplicate MAC detection timer interval and the maximum count.

```
device(config-evpn-default)# duplicate-mac-timer 22 max-count 5
```

The following example configures the EVPN instance “myinstance” so that the RD value is generated automatically. The import and export route-target community attributes are also generated automatically. VLANs are added to the EVI. The duplicate MAC detection timer is set to 22 seconds and the number of times a MAC move can be detected in the configured 22 second interval before the MAC is suppressed is set to 5.

```
device# configure terminal
device(config)# evpn-instance myinstance
device(config-evpn-default)# rd auto
device(config-evpn-default)# route-target both auto
device(config-evpn-default)# vni add 1-100
device(config-evpn-default)# duplicate-mac-timer 22 max-count 5
```

Configuring a superspine switch

This task configures a superspine switch that can be used to interconnect one datacenter pod to another. This example uses OSPF as the routing protocol for the interconnect.

1. In interface subtype configuration mode, configure an Ethernet interface for point-to-point connectivity between superspine switch SuperSpine and supported spine switches.

   NOTE

   This example uses both IPv4 and IPv6 addresses, with /31 and /127 networks, respectively.

```
SuperSpine# configure terminal
SuperSpine(config)# interface ethernet 0/11
SuperSpine(conf-if-eth-0/11)# ip address 1.4.0.0/31
SuperSpine(conf-if-eth-0/11)# ipv6 adddress 1000:1::/127
SuperSpine(conf-if-eth-0/11)# no shut
SuperSpine(conf-if-eth-0/11)# exit
SuperSpine(config)#
```

2. Configure a loopback interface.

   NOTE
   This example uses both IPv4 and IPv6 addresses, with /32 and /128 networks, respectively.

```
SuperSpine(config)# interface loopback 1
SuperSpine(config-Loopback-1)# ip address 1.0.0.1/32
SuperSpine(config-Loopback-1)# ipv6 address 1000:1::1/128
SuperSpine(config-Loopback-1)# no shut
SuperSpine(config-Loopback-1)# exit
SuperSpine(config)#
```

3. (Optional) You can configure unnumbered IPv4 and IPv6 addresses, with an appropriate loopback address, as follows.

```
SuperSpine(config)# interface ethernet 0/11
SuperSpine(conf-if-eth-0/11)# ip unnumbered loopback 1
SuperSpine(conf-if-eth-0/11)# ipv6 unnumbered loopback 1
SuperSpine(conf-if-eth-0/11)# no shut
```

4. Repeat Step 1 through Step 3, as appropriate, for all Ethernet interfaces supporting the spine switches.
5. Enable BGP routing and enter BGP global configuration mode.

```
SuperSpine(config)# router bgp
SuperSpine(config-bgp-router)#
```

6. Configure a local AS for the superspine switch.

```
SuperSpine(config-bgp-router)# local-as 1
```

   NOTE
   A local AS number facilitates the merging of networks by allowing a switch in an acquired network to appear to belong to the former AS. In eBGP deployments, the local AS values must be different for each node.

7. Use the **neighbor** command to configure a remote AS number for both IPv4 and IPv6 addresses to support BGP sessions to the spine switches.

```
SuperSpine(config-bgp-router)# neighbor 1.2.1.1 remote-as 2
SuperSpine(config-bgp-router)# neighbor 1000:1:2:1:: remote-as 2
SuperSpine(config-bgp-router)# neighbor 1.3.1.1 remote-as 2
SuperSpine(config-bgp-router)# neighbor 1000:1:5:1:: remote-as 2
```

8. Enter address-family IPv4 unicast configuration mode and configure the following attributes.
   a) Configure the maximum number of paths for ECMP load balancing. (The default is 1.)

```
SuperSpine(config-bgp-ipv4u)# maximum-paths 8
```

   b) Enable BGP recursive next-hop lookups.

```
SuperSpine(config-bgp-ipv4u)# next-hop-recursion
```

   c) Enable directly connected routes to be redistributed into BGP.

```
SuperSpine(config-bgp-ipv4u)# redistribute connected
```

   d) Enable OSPF connected routes to be redistributed into BGP, and exit to BGP configuration mode.

```
Leaf1(config-bgp-ipv4u)# redistribute ospf
Leaf1(config-bgp-ipv4u)# exit
Leaf1(config-bgp-router)#
```

9. Enter address-family IPv6 unicast configuration mode and configure the following attributes.
   a) Activate the interface, enabling the exchange of information with BGP neighbors and peer groups.

```
SuperSpine(config-bgp-router)# address-family ipv6 unicast
SuperSpine(config-bgp-ipv6u)# neighbor 1000:1:2:1::1 activate
SuperSpine(config-bgp-ipv6u)# neighbor 1000:1:3:1::1 activate
```

   b) Configure the maximum number of paths for ECMP load balancing. (The default is 1.)

```
SuperSpine(config-bgp-ipv6u)# maximum-paths 8
```

   c) Enable BGP recursive next-hop lookups.

```
SuperSpine(config-bgp-ipv6u)# next-hop-recursion
```

   d) Enable directly connected routes to be redistributed into BGP.

```
SuperSpine(config-bgp-ipv6u)# redistribute connected
```

   e) Enable OSPF connected routes to be redistributed into BGP, and exit to BGP configuration mode.

```
SuperSpine(config-bgp-ipv6u)# redistribute ospf
SuperSpine(config-bgp-ipv6u)# exit
SuperSpine(config-bgp-router)#
```

10. (Optional) Enter address-family Layer 2 Virtual Private Network (VPN) Ethernet VPN (EVPN) configuration mode and configure the following attributes for IPv4 addresses.
   a) Activate the interface, enabling the exchange of information with BGP neighbors and peer groups.

```
SuperSpine(config-bgp-router)# address-family l2vpn evpn
SuperSpine(config-evpn)# neighbor 1.2.1.0 activate
```

   b) Enable BGP to send an update to an eBGP multihop peer with the next-hop attribute unchanged.

```
SuperSpine(config-evpn)# neighbor 1.2.1.0 next-hop-unchanged
```

   c) Repeat Step 10a and Step 10b, as appropriate, for the remaining IPv4 leaf interfaces.
11. Enter the **retain-route-target-all** under BGP EVPN address-family configuration mode to configure the route reflector (RR) to accept all route targets (RTs), preventing filtering on routes that do not match the local configuration on the spine.

```
SuperSpine(config-bgp-router)# address-family l2vpn evpn
SuperSpine(config-bgp-evpn)# retain route-target all
```

12. (Optional) Configure neighbor Bidirectional Forwarding Detection (BFD), by means of the **neighbor bfd** command, to facilitate fast recovery in case of a link failure.

```
SuperSpine(config-bgp-router)# neighbor 1.2.1.1 bfd
SuperSpine(config-bgp-router)# neighbor 1000:1:2:1:: bfd
SuperSpine(config-bgp-router)# neighbor 1.3.1.1 bfd
SuperSpine(config-bgp-router)# neighbor 1000:1:3:1:: bfd
```

13. Repeat Step 1 through Step 9, as appropriate, for a peer superspine switch.

Configuring interoperability with other vendors

When BGP EVPN is used, the automatic mapping of VLANs to VNIs may not work with all vendors.

Some vendors do not use the range of available VNIs (1 through 8191) that Extreme does. As a result of the automatic mapping process that is initiated by the **map vni auto** command in overlay-gateway configuration mode (entered by means of the **overlay-gateway** command), a vendor's VLANs that do not conform to the Extreme extended VLAN range must be remapped manually to their respective VNIs.

In addition, the route target (RT) and route distinguisher (RD) values are set automatically. Because such route targets use AS numbers in the administrator field, routes cannot be exchanged between leaf nodes belonging to different AS numbers. In such cases manual adjustments are required, as illustrated in this task.

1. From global configuration mode, enter overlay-gateway configuration mode and specify an overlay gateway.

```
device(config)# overlay-gateway gateway1
device(config-overlay-gw-gateway1)#
```

2. In overlay-gateway configuration mode, do the following.
   a) Use the **map vlan** command to map a VLAN to a VNI manually, as in the following example.

```
device(config-overlay-gw-gatweway1)# map vlan 100 vni 10100
```

   b) Repeat Step 2a as appropriate for additional manual mappings.

3. Enter EVPN configuration mode.

```
device# configure terminal
device(config)# evpn-instance myinstance
device(config-evpn-myinstance)#
```

4. In EVPN instance configuration mode, use the **route-target (EVPN)** command to specify auto-generation of the import and export route-target community attributes and ignore the ASN, and the **rd auto (EVPN)** command to enable auto-generation of a route distinguisher (RD) for an EVPN instance.

```
device(config-evpn-myinstance)# route-target both auto ignore-as
device(config-evpn-instance-myinstance)# rd auto
```

5. In EVPN instance configuration mode, configure the VPN route distinguisher (RD) manually.
   a) Enter the **vlan** command to specify the VLAN and enter EVPN VLAN configuration mode, where you use the **rd (VNI)** command and the **route-target (VNI)** command to specify manually the RD and RT, respectively.

```
device(config-evpn-myinstance)# vlan 10
device(evpn-vlan-10)# rd 100:100
device(evpn-vlan-10)# route-target import 1:1
device(evpn-vlan-10)# route-target export 1:1
```

   b) Repeat Step 6a for all RDs and RTs that need to be mapped manually.

Verifying the configuration

A variety of **show** commands can be used to verify configurations on a leaf or spine.

Verifying configurations on a leaf

To verify the Ethernet configuration, use the **show running-config interface** command, as in the following example.

```
Leaf1# show running-config interface ethernet 0/16
interface Ethernet 0/16
 switchport
 switchport mode trunk
 switchport trunk allowed vlan add 22-25
 switchport trunk tag native-vlan
 spanning-tree shutdown
 fabric isl enable
 fabric trunk enable
 no shutdown
```

To verify a VLAN configuration, use the **show running-config vlan** command, as in the following example.

```
Leaf1# show running-config interface vlan 25
interface Vlan 25
 suppress-nd
 suppress-arp
```

To verify a virtual Ethernet (VE) configuration, use the **show running-config interface interface ve** command, as in the following example.

```
Leaf1# show running-config ve 25
interface Ve 25
 vrf forwarding vrf2
 ipv6 anycast-address 2:25:1::254/64
 ip anycast-address 2.25.1.254/24
 no shutdown
```

To verify IPv4 or IPv6 static-anycast-gateway configurations, use the **show running-config ip anycast-gateway-mac** command or the **show running-config ipv6 anycast-gateway-mac** command, respectively, as in the following examples.

```
Leaf1# show running-config ip anycast-gateway-mac
ip anycast-gateway-mac 0000.abba.abba

Leaf1# show running-config ipv6 anycast-gateway-mac
ipv6 anycast-gateway-mac 0000.abba.abba
```

To verify the entire BGP configuration, use the **show running-config router bgp** command, as in the following examples.

```
Leaf1# show running-config router bgp
router bgp
  local-as 65006
  neighbor 1000:2:6:: remote-as 65002
  neighbor 1000:2:6:: password 2 $TDk1UHNkRC1afDg=
  neighbor 1000:3:6:: remote-as 65002
  neighbor 1000:3:6:: password 2 $TDk1UHNkRC1afDg=
  neighbor 2.6.0.0 remote-as 65002
  neighbor 2.6.0.0 password 2 $TDk1UHNkRC1afDg=
  neighbor 2.6.0.0 bfd
  neighbor 3.6.0.0 remote-as 65002
  neighbor 3.6.0.0 password 2 $TDk1UHNkRC1afDg=
  neighbor 3.6.0.0 bfd
  address-family ipv4 unicast

   redistribute connected
   neighbor 3.6.0.0 allowas-in 1
   neighbor 2.6.0.0 allowas-in 1
   maximum-paths 8
   next-hop-recursion
```

```
!
address-family ipv4 unicast vrf vrf12
 redistribute connected
 maximum-paths 8
!
address-family ipv4 unicast vrf vrf13
 redistribute connected
 maximum-paths 8
!
<---output omitted--->
!
address-family ipv4 unicast vrf vrf8
 redistribute connected
 maximum-paths 8
!
address-family ipv4 unicast vrf vrf9
 redistribute connected
 maximum-paths 8
!
address-family ipv6 unicast

 redistribute connected
 neighbor 1000:3:6:: allowas-in 1
 neighbor 1000:3:6:: activate
 neighbor 1000:2:6:: allowas-in 1
 neighbor 1000:2:6:: activate
 maximum-paths 8
 next-hop-recursion
!
address-family ipv6 unicast vrf vrf12
 redistribute connected
 maximum-paths 8
!
address-family ipv6 unicast vrf vrf13
 redistribute connected
 maximum-paths 8
!
 <---output omitted--->
!
address-family ipv6 unicast vrf vrf8
 redistribute connected
 maximum-paths 8
!
address-family ipv6 unicast vrf vrf9
 redistribute connected
 maximum-paths 8
!
address-family l2vpn evpn
 neighbor 3.6.0.0 activate
 neighbor 3.6.0.0 allowas-in 1
 neighbor 3.6.0.0 next-hop-unchanged
 neighbor 2.6.0.0 activate
 neighbor 2.6.0.0 allowas-in 1
 neighbor 2.6.0.0 next-hop-unchanged
```

To view summary BGP information, use the **show ip bgp summary** command, as in the following example.

```
Leaf1# show ip bgp summary
  BGP4 Summary
  Router ID: 6.0.0.6   Local AS Number: 65006
  Confederation Identifier: not configured
  Confederation Peers:
  Maximum Number of IP ECMP Paths Supported for Load Sharing: 8
  Number of Neighbors Configured: 2, UP: 1
  Number of Routes Installed: 17, Uses 2040 bytes
  Number of Routes Advertising to All Neighbors: 19 (16 entries), Uses 960 bytes
  Number of Attribute Entries Installed: 6, Uses 690 bytes
  Neighbor Address  AS#         State     Time     Rt:Accepted Filtered Sent     ToSend
  2.6.0.0           65002       CONN      1h23m33s    0        0        0        16
  3.6.0.0           65002       ESTAB    23h28m32s    14       0        3        0
```

To view summary BGP EVPN information, use the **show bgp evpn summary** command, as in the following example.

```
Leaf1# show bgp evpn summary
  BGP4 Summary
  Router ID: 6.0.0.6   Local AS Number: 65006
  Confederation Identifier: not configured
  Confederation Peers:
  Maximum Number of IP ECMP Paths Supported for Load Sharing: 1
  Number of Neighbors Configured: 2, UP: 1
  Number of Routes Installed: 4536, Uses 544320 bytes
  Number of Routes Advertising to All Neighbors: 5652 (4536 entries), Uses 272160 bytes
  Number of Attribute Entries Installed: 2327, Uses 267605 bytes
  Neighbor Address  AS#         State     Time     Rt:Accepted Filtered Sent     ToSend
  2.6.0.0           65002       CONN      1h23m39s    0        0        0        4536
  3.6.0.0           65002       ESTAB    23h28m37s    3420     24       1116     0
```

To verify the BFD neighbors configuration, use the **show bfd neighbors** command, as in the following example.

```
Leaf1# show bfd neighbors
OurAddr                                 NeighAddr                       State     Int
=======                                 =========                       =====     ===
3.6.0.1                                 3.6.0.0                         UP        Eth 0/5
```

Verifying configurations on a spine

To verify the BGP configuration, use the **show ip bgp summary** command, as in the following example.

```
Spine2# show ip bgp summary
  BGP4 Summary
  Router ID: 3.0.0.3   Local AS Number: 65002
  Confederation Identifier: not configured
  Confederation Peers:
  Maximum Number of IP ECMP Paths Supported for Load Sharing: 8
  Number of Neighbors Configured: 5, UP: 5
  Number of Routes Installed: 22, Uses 2640 bytes
  Number of Routes Advertising to All Neighbors: 70 (16 entries), Uses 960 bytes
  Number of Attribute Entries Installed: 7, Uses 805 bytes
  Neighbor Address  AS#         State     Time     Rt:Accepted Filtered Sent     ToSend
  1.3.0.0           65001       ESTAB    23h14m16s    4        0        13       0
  3.6.0.1           65006       ESTAB    23h30m19s    3        0        14       0
  3.7.0.1           65078       ESTAB    23h21m35s    3        0        15       0
  3.8.0.1           65078       ESTAB    23h21m47s    3        0        14       0
  3.9.0.1           65009       ESTAB    23h24m15s    3        0        14       0
```

To verify the BGP EVPN configuration, use the **show bgp evpn summary** command, as in the following example.

```
Spine2# show bgp evpn summary
  BGP4 Summary
  Router ID: 3.0.0.3   Local AS Number: 65002
  Confederation Identifier: not configured
  Confederation Peers:
  Maximum Number of IP ECMP Paths Supported for Load Sharing: 1
  Number of Neighbors Configured: 5, UP: 5
  Number of Routes Installed: 4536, Uses 544320 bytes
  Number of Routes Advertising to All Neighbors: 18144 (4536 entries), Uses 272160 bytes
  Number of Attribute Entries Installed: 2200, Uses 253000 bytes
  Neighbor Address  AS#         State     Time     Rt:Accepted Filtered Sent     ToSend
  1.3.0.0           65001       ESTAB    23h14m37s    0        0        4536     0
  3.6.0.1           65006       ESTAB    23h30m40s    1116     0        3420     0
  3.7.0.1           65078       ESTAB    23h21m57s    1152     0        3384     0
  3.8.0.1           65078       ESTAB    23h22m 9s    1152     0        3384     0
  3.9.0.1           65009       ESTAB    23h24m36s    1116     0        3420     0
```

To verify the BFD neighbors configuration, use the **show bfd neighbors** command, as in the following example.

```
Spine2# show bfd neighbors
OurAddr                                 NeighAddr                           State     Int
```

```
=======                         =========                       =====     ===
3.6.0.0                         3.6.0.1                         UP        Eth 0/5
```

Configuring additional features for IP Fabrics

The following optional tasks enhance the implementation and management of IP Fabrics. The configuration of BGP EVPN, although demonstrated as the overlay implementation shown in this guide, is not essential to the deployment of an IP Fabric.

Configuring MAC learning of Layer 2 extension site through BGP

The user can choose the way MAC learning is achieved for a Layer 2 extension tunnel.

BGP routing must be configured.

Data plane (Layer 2) MAC address learning is enabled by default at the remote site. However, this leads to scalability issues. Where BGP routing is used, do the following to enable MAC learning by means of BGP instead. This delegates the responsibility for MAC learning on a tunnel to the Layer 3 control-plane protocol, such as BGP EVPN.

1. Enter VXLAN overlay-gateway site configuration mode.

```
device# configure terminal
device(config)# overlay-gateway gateway1
device(config-overlay-gw-gateway1)# site mysite
device(config-overlay-gw-gateway1-site-mysite)#
```

2. Enter the **mac-learning protocol bgp** command.

```
device(config-overlay-gw-gateway1-site-mysite)# mac-learning protocol bgp
```

3. To disable BGP MAC learning and return to the default of Layer 2 learning, use the **no mac-learning protocol bgp** command.

```
device(config-overlay-gw-gateway1-site-mysite)# no mac-learning protocol bgp
```

Disabling automatic VXLAN tunnel endpoint discovery by BGP

Automatic VXLAN tunnel endpoint (VTEP) discovery by BGP is enabled by default and can be disabled by means of the following procedure.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **router bgp** command to enable BGP routing.

```
device(config)# router bgp
```

3. Enter the **address-family l2vpn evpn** command to enter L2VPN EVPN address family configuration mode.

```
device(config-bgp-router)# address-family l2vpn evpn
```

4. Enter the **no vtep-discovery** command to disable automatic VTEP discovery by BGP.

```
device(config-bgp-evpn)# no vtep-discovery
```

The following example disables automatic VTEP discovery by BGP.

```
device# configure terminal
device(config)# router bgp
device(config-bgp-router)# address-family l2vpn evpn
device(config-bgp-evpn)# no vtep-discovery
```

Configuring BGP next hop unchanged

BGP next hop unchanged can be configured for the L2VPN EVPN address family, allowing BGP to send updates to eBGP peers with the next hop attribute unchanged. The **neighbor next-hop-unchanged** command should be configured for the Spine switch for each EVPN neighbor if the leaf switch is an eBGP peer. The **neighbor next-hop-unchanged** command should be configured for the leaf switch for each EVPN neighbor if the spine switch is an eBGP peer.

1. Enter the **configure terminal** command to access global configuration mode.

   ```
   device# configure terminal
   ```

2. Enter the **router bgp** command to enable BGP routing.

   ```
   device(config)# router bgp
   ```

3. Enter the **local-as** command to configure the autonomous system number (ASN) in which your device resides.

   ```
   device(config-bgp-router)# local-as 1000
   ```

4. Enter the **neighbor** *ipv4* **remote-as** command to specify the ASN in which the remote neighbor resides.

   ```
   device(config-bgp-router)# neighbor 10.1.1.1 remote-as 2
   ```

5. Enter the **address-family l2vpn evpn** command to enter L2VPN EVPN address family configuration mode.

   ```
   device(config-bgp-router)# address-family l2vpn evpn
   ```

6. Enter the **neighbor** *ip address* **activate** command to establish a BGP EVPN session with the eBGP neighbor.

   ```
   device(config-bgp-evpn)# neighbor 10.1.1.1 activate
   ```

7. Enter the **neighbor** *ip address* **neighbor next-hop-unchanged** command to configure BGP next hop unchanged.

   ```
   device(config-bgp-evpn)# neighbor 10.1.1.1 next-hop-unchanged
   ```

The following example establishes a BGP EVPN session with an eBGP neighbor and configures BGP next hop unchanged so that updates can be sent to the neighbor with the next hop attribute unchanged.

```
device# configure terminal
device(config)# router bgp
device(config-bgp-router)# local-as 1000
device(config-bgp-router)# neighbor 10.1.1.1 remote-as 2
device(config-bgp-router)# address-family l2vpn evpn
device(config-bgp-evpn)# neighbor 10.1.1.1 activate
device(config-bgp-evpn)# neighbor 10.1.1.1 next-hop-unchanged
```

Configuring BGP retain route target

BGP retain route target can be configured for the L2VPN EVPN address family so that a route reflector (RR) accepts all route targets (RTs). For iBGP, enable this feature on a spine switch so that route-target filtering is not performed and all route targets are retained and route-target attributes in the EVPN routes are not modified or removed.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **router bgp** command to enable BGP routing.

```
device(config)# router bgp
```

3. Enter the **local-as** command to configure the autonomous system number (ASN) in which your device resides.

```
device(config-bgp-router)# local-as 1000
```

4. Enter the **neighbor** *ipv4* **remote-as** command to specify the ASN in which the remote neighbor resides.

```
device(config-bgp-router)# neighbor 10.1.1.1 remote-as 2
```

5. Enter the **address-family l2vpn evpn** command to enter L2VPN EVPN address family configuration mode.

```
device(config-bgp-router)# address-family l2vpn evpn
```

6. Enter the **retain route-target all** command to configure the RR to accept all route targets RTs.

```
device(config-bgp-evpn)# retain route-target all
```

The following example configures a RR to accept all RTs.

```
device# configure terminal
device(config)# router bgp
device(config-bgp-router)# local-as 1000
device(config-bgp-router)# neighbor 10.1.1.1 remote-as 2
device(config-bgp-router)# address-family l2vpn evpn
device(config-bgp-evpn)# retain route-target all
```

Applying a BGP extended community filter for the L2VPN EVPN address family

A BGP extended community filter can be applied for the L2VPN EVPN address family.

BGP communities must already be defined. For more information on defining BGP communities, refer to the “BGP4” and “BGP4+” chapters in the *Extreme SLX-OS Layer 3 Routing Configuration Guide*

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **ip extcommunity-list** command and specify a number to set a BGP extended community filter.

```
device(config)# ip extcommunity-list 1 permit rt 123:2
```

3. Enter the **router bgp** command to enable BGP routing.

```
device(config)# router bgp
```

4. Enter the **local-as** command to configure the autonomous system number (ASN) in which your device resides.

```
device(config-bgp-router)# local-as 1000
```

5. Enter the **neighbor** *ip-address* **remote-as** command to specify the ASN in which the remote neighbor resides.

```
device(config-bgp-router)# neighbor 10.1.2.3 remote-as 1001
```

6. Enter the **address-family l2vpn evpn** command to enter L2VPN EVPN address family configuration mode.

```
device(config-bgp-router)# address-family l2vpn evpn
```

7. Enter the **neighbor** *ip-address* **activate** command to enable the exchange of information with the neighbor.

```
device(config-bgp-evpn)# neighbor 10.1.2.3 activate
```

8. Enter the **neighbor** *ip-address* **route-map** command and specify the **in** keyword to apply a route map to incoming routes.

```
device(config-bgp-evpn)# neighbor 10.1.2.3 route-map in extComRmapt
```

9. Enter the **neighbor** *ip-address* **route-map** command and specify the **out** keyword to apply a route map to outgoing routes.

```
device(config-bgp-evpn)# neighbor 10.1.2.3 route-map out sendExtComRmap
```

10. Enter the **neighbor** *ip-address* **send-community** command and specify the **both** keyword to enable the sending of standard and extended attributes in updates to the specified BGP neighbor.

```
device(config-bgp-evpn)# neighbor 10.1.2.3 send-community both
```

The following example applies a BGP extended community filter for the L2VPN EVPN address family. The steps for configuring a route map are not included in this example.

```
device# configure terminal
device(config)# ip extcommunity-list 1 permit rt 123:2
device(config)# router bgp
device(config-bgp-router)# local-as 1000
device(config-bgp-router)# neighbor 10.1.2.3 remote-as 1001
device(config-bgp-router)# address-family l2vpn evpn
device(config-bgp-evpn)# neighbor 10.1.2.3 activate
device(config-bgp-evpn)# neighbor 10.1.2.3 route-map in extComRmapt
device(config-bgp-evpn)# neighbor 10.1.2.3 route-map out sendExtComRmap
device(config-bgp-evpn)# neighbor 10.1.2.3 send-community both
```

Configuring BGP graceful restart for the L2VPN EVPN address family

Graceful restart can be configured for BGP EVPN in the L2VPN EVPN address family configuration mode, helping to retain peer routes and ensure that no route and topology changes occur in the network for the duration of a restart.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **router bgp** command to enable BGP routing.

```
device(config)# router bgp
```

3. Enter the **local-as** command to configure the autonomous system number (ASN) in which your device resides.

```
device(config-bgp-router)# local-as 1000
```

4. Enter the **neighbor** *ip address* **remote-as** command to specify the ASN in which the neighbor resides.

```
device(config-bgp-router)# neighbor 2001:db8:93e8:cc00::1 remote-as 2
```

5. Enter the **address-family l2vpn evpn** command to enter L2VPN EVPN address family configuration mode.

```
device(config-bgp-router)# address-family l2vpn evpn
```

6. Enter the **graceful-restart** command to enable the graceful restart feature.

```
device(config-bgp-evpn)# graceful-restart
```

The following example enables the graceful restart feature.

```
device# configure terminal
device(config)# router bgp
device(config-bgp-router)# local-as 1
device(config-bgp-router)# neighbor 2001:db8:93e8:cc00::1 remote-as 2
device(config-bgp-router)# address-family l2vpn evpn
device(config-bgp-evpn)# graceful-restart
```

Configuring BGP peer groups for the L2VPN EVPN address family

A peer group can be created and activated in the L2VPN EVPN address family configuration mode.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **router bgp** command to enable BGP routing.

```
device(config)# router bgp
```

3. Enter the **local-as** command to configure the autonomous system number (ASN) in which your device resides.

```
device(config-bgp-router)# local-as 1000
```

4. Enter the **neighbor** *peer-group-name* **peer-group** command to create a peer group.

```
device(config-bgp-router)# neighbor mypeergroup1 peer-group
```

5. Enter the **neighbor** *peer-group-name* **remote-as** command to specify the ASN of the peer group.

```
device(config-bgp-router)# neighbor mypeergroup1 remote-as 11
```

6. Enter the **neighbor** *ipv6-address* **peer-group** command to associate a neighbor with the peer group.

```
device(config-bgp-router)# neighbor 2001:2018:8192::125 peer-group mypeergroup1
```

7. Enter the **neighbor** *ipv6-address* **peer-group** command to associate a second neighbor with the peer group.

```
device(config-bgp-router)# neighbor 2001:2018:8192::124 peer-group mypeergroup1
```

8. Enter the **address-family l2vpn evpn** command to enter L2VPN EVPN address family configuration mode.

```
device(config-bgp-router)# address-family l2vpn evpn
```

9. Enter the **neighbor** *peer-group-name* **activate** command to establish a BGP EVPN session with the peer group.

```
device(config-bgp-evpn)# neighbor mypeergroup1 activate
```

The following example creates a peer group, specifying two neighbors to belong to the peer group, and activates the peer group in L2VPN EVPN.

```
device# configure terminal
device(config)# router bgp
device(config-bgp-router)# local-as 1000
device(config-bgp-router)# neighbor mypeergroup1 peer-group
device(config-bgp-router)# neighbor mypeergroup1 remote-as 11
device(config-bgp-router)# neighbor 2001:2018:8192::125 peer-group mypeergroup1
device(config-bgp-router)# neighbor 2001:2018:8192::124 peer-group mypeergroup1
device(config-bgp-router)# address-family l2vpn evpn
device(config-bgp-evpn)# neighbor mypeergroup1 activate
```

Configuring a route reflector client for the L2VPN EVPN address family

A BGP peer can be configured as a route reflector (RR) client for the L2VPN EVPN address family. When iBGP is used for BGP EVPN in an IP Fabric, spine switches are configured as RRs and leaf switches are configured as RR clients. The following task configures an RR client.

1. Enter the **configure terminal** command to access global configuration mode.

   ```
   device# configure terminal
   ```

2. Enter the **router bgp** command to enable BGP routing.

   ```
   device(config)# router bgp
   ```

3. Enter the **local-as** command to configure the autonomous system number (ASN) in which your device resides.

   ```
   device(config-bgp-router)# local-as 1000
   ```

4. Enter the **neighbor** *ipv4* **remote-as** command to specify the ASN in which the remote neighbor resides.

   ```
   device(config-bgp-router)# neighbor 10.1.1.1 remote-as 3
   ```

5. Enter the **address-family l2vpn evpn** command to enter L2VPN EVPN address family configuration mode.

   ```
   device(config-bgp-router)# address-family l2vpn evpn
   ```

6. Enter the **neighbor** *ip address* **route-reflector-client** command to configure a specified neighbor to be a route reflector client.

   ```
   device(config-bgp-evpn)# neighbor 10.1.1.1 route-reflector-client
   ```

The following example configures a neighbor with the IP address 10.1.1.1 to be a route reflector client in L2VPN EVPN configuration mode.

```
device# configure terminal
device(config)# router bgp
device(config-bgp-router)# local-as 1000
device(config-bgp-router)# neighbor 10.1.1.1 remote-as 3
device(config-bgp-router)# address-family l2vpn evpn
device(config-bgp-evpn)# neighbor 10.1.1.1 route-reflector-client
```

Disabling client-to-client reflection for the L2VPN EVPN address family

For iBGP, if the leaf switches are configured as route reflector clients and are connected in a full iBGP mesh, you can disable client-to-client reflection on the spine switch that is configured as a route reflector (RR).

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **router bgp** command to enable BGP routing.

```
device(config)# router bgp
```

3. Enter the **local-as** command to configure the autonomous system number (ASN) in which your device resides.

```
device(config-bgp-router)# local-as 1000
```

4. Enter the **neighbor** *ipv4* **remote-as** command to specify the ASN in which the remote neighbor resides.

```
device(config-bgp-router)# neighbor 10.1.1.1 remote-as 3
```

5. Enter the **address-family l2vpn evpn** command to enter L2VPN EVPN address family configuration mode.

```
device(config-bgp-router)# address-family l2vpn evpn
```

6. Enter the **no client-to-client-reflection** command to disable client-to-client reflection.

```
device(config-bgp-evpn)# no client-to-client-reflection
```

The following example disables client-to-client reflection in L2VPN EVPN configuration mode.

```
device# configure terminal
device(config)# router bgp
device(config-bgp-router)# local-as 1000
device(config-bgp-router)# neighbor 10.1.1.1 remote-as 3
device(config-bgp-router)# address-family l2vpn evpn
device(config-bgp-evpn)# no client-to-client-reflection
```

Disabling the BGP AS_PATH check function for the L2VPN EVPN address family

A device can be configured so that the AS_PATH check function for routes learned from a specific peer is disabled, and routes that contain the recipient BGP speaker's AS number are not rejected. For eBGP, with leaves in one AS and spines in another AS, the AS_PATH check function can be disabled for a leaf switch, so that route updates from the Spine layer, containing routes from the other leaves, are not discarded by the receiving leaf.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **router bgp** command to enable BGP routing.

```
device(config)# router bgp
```

3. Enter the **local-as** command to configure the autonomous system number (ASN) in which your device resides.

```
device(config-bgp-router)# local-as 1000
```

4. Enter the **neighbor** *ipv6-address* **remote-as** command to specify the ASN in which the remote neighbor resides.

```
device(config-bgp-router)# neighbor 2001:db8:e0ff:783a::4 remote-as 2
```

5. Enter the **address-family l2vpn evpn** command to enter L2VPN EVPN address family configuration mode.

```
device(config-bgp-router)# address-family l2vpn evpn
```

6. Enter the **neighbor** *ipv6 address* **activate** command to establish a BGP EVPN session with the eBGP neighbor.

```
device(config-bgp-evpn)# neighbor 2001:db8:e0ff:783a::4 activate
```

7. Enter the **neighbor** *ipv6-address* **allowas-in** command and specify a **number** to disable the BGP AS_PATH check function, and specify the number of times that the AS path of a received route may contain the recipient BGP speaker's AS number and still be accepted.

```
device(config-bgp-evpn)# neighbor 2001:db8:e0ff:783a::4 allowas-in 1
```

This example specifies that the AS path of a received route may contain the recipient BGP speaker's AS number once and still be accepted in L2VPN EVPN.

```
device# configure terminal
device(config)# router bgp
device(config-bgp-router)# local-as 1000
device(config-bgp-router)# neighbor 2001:db8:e0ff:783a::4 remote-as 2
device(config-bgp-router)# address-family l2vpn evpn
device(config-bgp-evpn)# neighbor 2001:db8:e0ff:783a::4 activate
device(config-bgp-evpn)# neighbor 2001:db8:e0ff:783a::4 allowas-in 1
```

Enabling the BGP AS_PATH check function for the sender BGP speaker

A device can be configured so that a BGP sender speaker does not send routes with an AS path that contains the ASN of the receiving speaker.

NOTE
This task enforces the outbound BGP AS_PATH check function for the BGP IPv4 unicast. This feature is also supported for the BGP address-family IPv6 unicast and BGP address-family L2VPN EVPN address families.

1. Enter the **configure terminal** command to access global configuration mode.

```
device# configure terminal
```

2. Enter the **router bgp** command to enable BGP routing.

```
device(config)# router bgp
```

3. Enter the **local-as** command to configure the autonomous system number (ASN) in which your device resides.

```
device(config-bgp-router)# local-as 1000
```

4. Enter the **neighbor** *ip-address* **remote-as** command to specify the ASN in which the remote neighbor resides.

```
device(config-bgp-router)# neighbor 10.1.1.1 remote-as 2
```

5. Enter the **address-family ipv4 unicast** command to enter BGP address-family IPv4 unicast configuration mode.

```
device(config-bgp-router)# address-family ipv4 unicast
```

6. Enter the **neighbor** *ip address* **enable-peer-as-check** command to enforce the outbound AS_PATH check function for the IPv4 address family.

```
device(config-bgp-ipv4u)# neighbor 10.1.1.1 enable-peer-as-check
```

The following example enables the outbound AS_PATH check function for the BGP IPv4 unicast address family for a particular peer.

```
device# configure terminal
device(config)# router bgp
device(config-bgp-router)# local-as 1000
device(config-bgp-router)# neighbor 10.1.1.1 remote-as 2
device(config-bgp-router)# address-family ipv4 unicast
device(config-bgp-ipv4u)# neighbor 10.1.1.1 enable-peer-as-check
```

The following example enables the outbound AS_PATH check function for the BGP IPv6 unicast address family for a particular peer.

```
device# configure terminal
device(config)# router bgp
device(config-bgp-router)# local-as 1000
device(config-bgp-router)# neighbor 2001:2018:8192::125 remote-as 2
device(config-bgp-router)# address-family ipv6 unicast
device(config-bgp-ipv6u)# neighbor 2001:2018:8192::125 enable-peer-as-check
```

The following example enables the outbound AS_PATH check function for the L2VPN EVPN address family for a particular peer.

```
device# configure terminal
device(config)# router bgp
device(config-bgp-router)# local-as 1000
device(config-bgp-router)# neighbor 10.1.1.1 remote-as 2
device(config-bgp-router)# address-family l2vpn evpn
device(config-bgp-evpn)# neighbor 10.1.1.1 enable-peer-as-check
```

A neighbor reset is required once this task is complete.

Embedded Fabric Automation

Overview

Embedded Fabric Automation (EFA) supports the automated configuration of an IP Fabric, greatly simplifying the time and effort that would otherwise be required.

EFA provides the following features:

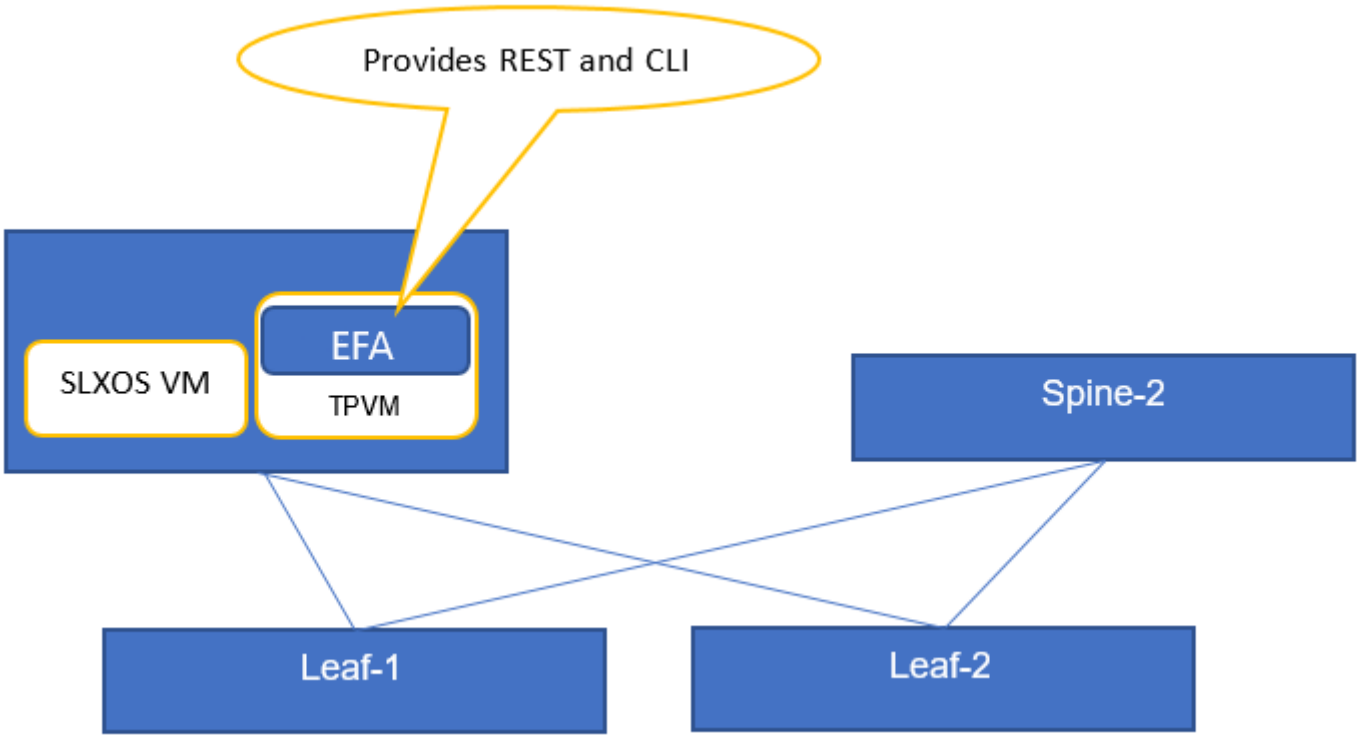
- Provides a quick and easy way to provision a new IP Fabric, or to add or remove new nodes to or from a fabric.
- Automates configuration on al nodes that are registered as part of a fabric.
- Runs as a service on Guest OS for TPVM.
- Provides ultra-high performance and speed achieved through concurrency.
- Uses an industry-standard Open API (https://www.openapis.org/)-based programmable interface.
- Uses CLI commands to manage devices in an IP Fabric.

This chapter presents the architecture, features, and use cases for support on the following platforms: SLX 9140, SLX 9240

High-level architecture

The following figure illustrates the high-level architecture.

FIGURE 17 High-level architecture



The SLX-OS virtual machine resides on an SLX platform within the fabric.

Configuration features

This section summarizes the configuration features of EFA.

Underlay

The following are configured automatically.

- Interfaces (IP numbered or unnumbered)
- BGP underlay for spine and leaf nodes, with default ASN numbers
- BGP auto neighbor discovery
- MCT configurations

Overlay

The following are enabled by default and can be disabled by changing the fabric setting configuration:

- Overlay gateway configuration
- Overlay gateway activation

Fabric Templates

EFA leverages fabric templates to configure switch interfaces, BGP peering, and EVPN settings. Templates make the IP Fabric provisioning very simple. Once a template is defined, EFA uses the information from the template to configure current and future switches, ensuring consistent configuration across the fabric with minimal effort. In addition, EFA includes a default template and also enables users to define custom templates to change any of the fabric parameters, for example, leaf or spine ASN ranges, IP address ranges, and so on.

A default fabric is configured with the following settings:

```
P2P_LINK_IP_RANGE=10.10.10.0/23
MCT_LINK_IP_RANGE=20.20.20.0/24
LOOPBACK_IP_RANGE=172.32.254.0/24
LEAF_ASN_BLOCK=65000-65534
SPINE_ASN_BLOCK=64512
```

The user can override the default settings by means of a CLI.

Persistence

EFA supports persistence for the following, so that information is not lost as a result of VM reboots:

- Fabric templates
- Fabric topologies

All network details discovered from devices are persistent. This information is retained following the reboot of TPVM on the server.

Zero-Touch Provisioning (ZTP)

ZTP can be used to upgrade nodes. However, to deploy EFA, the user must execute the **efa deploy** command from the SLX-OS CLI.

Troubleshooting

The following table lists support for troubleshooting.

TABLE 19 Troubleshooting features

Feature	Description
Node configuration failure reporting	Reporting to the console, with detailed information in log files. No information is written to Syslog.
Incorrect correct cabling reporting	Reporting to the console, with log files capturing (a) spines connected to other spines and (b) more than two leaf nodes connected. Cabling issues reported when devices are added.
Reporting on and overwriting conflicting configurations	The following configuration problems can occur: • IP address conflicts • IP interfaces configured as unnumbered, but user wants numbered links • Overlay gateways configured with different names from desired names • EVPN instances configured with different name from desired names • Port-channel speeds incorrect during cluster formation • Cluster formation issues resulting from timing problems or invalid configurations

Deploying and using EFA

Deploying EFA

This section shows how to deploy Embedded Fabric Automation, and provides the system response.

The deployment of the application on both client and server in the TPVM is enabled.

> NOTE
> Refer to the "Guest OS for TPVM" chapter in the *SLX-OS Management Configuration Guide*. For additional details on using TPVM for EFA deployment, refer to Using TPVM on page 92.

Deploy the application by executing the **efa deploy** command on one of the spine devices.

```
device# efa deploy user admin password password
```

The username and password are optional. If not specifiec, the defaults "admin" and "password" are used.

A successful response is as follows. (Output from this process is recorded in /var/log/efa_deploy.log.)

```
Step 1 Checking if TPVM is already setup
Done
Step 2 Checking if TPVM is already running
Done
Step 3 Setting auto-boot for TPVM
Done
Step 4 Stopping TPVM for EFA deployment
Done
Step 5 Creating required directories
Done
Step 6 Deploying EFA client
Done
Step 7 Deploying EFA server
Done
Step 8 Adding EFA binaries to path
Done
Step 9 Starting TPVM after EFA deployment
Done
Step 10 Get IP Address assigned to TPVM to deploy the app
Done
Step 11 Verifying Client and Server deployment on the TPVM
Done
Application is Installed and ready for use on the TPVM with IP:  10.25.225.160
```

1.

If the user chooses not to use ZTP, the same script can be executed through the command line.

Using TPVM

This section provides additional details on using TPVM for this application.

The following conditions are assumed for EFA deployment to work.

- No error messages or warning logs should be present. Use the **tpvm show status** command to see additional information.
- It is assumed that TPVM obtains an IP address for the host device by means of DHCP or through static address allocation. The IP address is not needed to deploy the application.
- For deployment verification to work, the host IP address must be reachable through TPVM.

The following steps summarize TPVM deployment details.

1. Use the **tpvm show status** command to confirm that TPVM is set up and running.

```
$ show tpvm status

TPVM is running, and AutoStart is enabled on this host.
```

2. Use the **tpvm auto-boot enable** command to enable auto-boot on the host.

```
$ tpvm auto-boot enable

auto-boot enable succeeds
```

3. Use the **tpvm show ip-address** command to confirm the IP address.

```
$ show tpvm ip-address

IP Address of the TPVM:  10.25.225.160
```

   As the application is deployed, by means of the **efa deploy** command, the following actions are taken:

   1. **ATTENTION**: TPVM is rebooted.
   2. The following directories are created:
      - /var/log/efa
      - /opt/efa
      - /var/efa
   3. The EFA binary and the EFA server (efa-server) binary are copied to the /opt/efa directory. The system makes sure that efa-server can start, stop, restart, and check the status of the server.
   4. A configuration file is placed in /opt/efa to ensure that efa-server starts up when TPVM boots up.

4. Use the **tpvm start** command to start TPVM after EFA deployment.

```
$ tpvm start

start succeeds
```

5. To verify the deployment, use SSH to connect to the TPVM and use the **efa fabric setting show** command.

```
$ efa fabric setting show
+---------------------------------------+------------------------+
|           NAME                        |      VALUE             |
+---------------------------------------+------------------------+
| Fabric Name                           | default                |
| Link IP Range                         | 10.10.10.0/23          |
| Loopback IP Range                     | 172.32.254.0/24        |
| Loopback Port Number                  | 1                      |
| VTEP Loopback Port Number             | 2                      |
| Spine ASN Block                       | 64512                  |
| LEAF ASN Block                        | 65000-65534            |
| P2P IP Type                           | numbered               |
+---------------------------------------+------------------------+
```

Configuring an IP Fabric with EFA

EFA achieves the following:

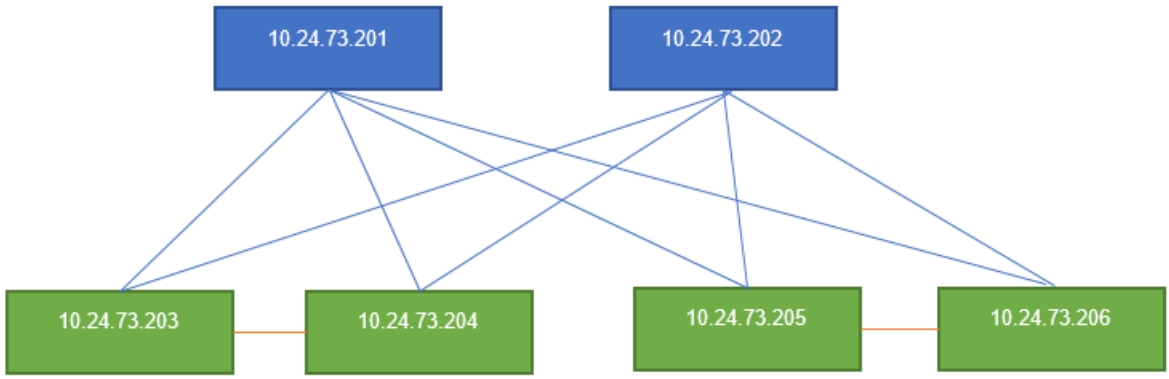
- Registers multiple switches to the fabric with the roles spine or leaf.
- Validates the physical topology.

- Configures the underlay and optional overlay networks on registered switches.

Default configurations on the switches must be configured by means of the **copy default-config startup-config** command.

The following figure illustrates a simplified two-tier IP Clos topology.

FIGURE 18 IP Fabric topology



This example uses the **efa fabric configure** command to register switches, along with their roles, in a default fabric.

```
$ efa fabric configure  --leaf <list-of-leaf-ips> --spine <list-of-spine-ips> --persist  --username <user-
name> --password <password> --force
```

NOTE
For CLI details, see the **efa fabric configure** command in Appendix D: EFA Commands on page 119.

Configuring additional switches

This use case addresses the configuration of additional switches following the initial default configuration.

This example uses the **efa fabric configure** command to configure additional comma-delimited addresses of leaf and spine nodes, respectively.

```
$ efa fabric configure --leaf 10.24.73.207,10.24.73.208
```

```
$ efa fabric configure --spine 10.24.73.209,10.24.73.210
```

NOTE
For CLI details, see the **efa fabric configure** command in Appendix D: EFA Commands on page 119.

Deleting switches from the fabric

This use case addresses the deletion of switches from the fabric following the initial default configuration.

This example uses the **efa fabric deconfigure** command to delete the comma-delimited management IP addresses of switches.

```
$ efa fabric deconfigure --device 10.24.73.207,10.24.73.208
```

NOTE
For CLI details, see the **efa fabric deconfigure** command in Appendix D: EFA Commands on page 119.

Displaying fabric settings

This use case addresses the displaying of parameters following the initial default configuration.

This is example output for basic parameters.

```
$ efa fabric setting show
+-----------------------------------------------+-------------------------+
|           NAME                                |      VALUE              |
+-----------------------------------------------+-------------------------+
| Fabric Name                                   | default                 |
| Link IP Range                                 | 10.10.10.0/23           |
| Loopback IP Range                             | 172.32.254.0/24         |
| Loopback Port Number                          | 1                       |
| VTEP Loopback Port Number                     | 2                       |
| Spine ASN Block                               | 64512                   |
| LEAF ASN Block                                | 65000-65534             |
| P2P IP Type                                   | numbered                |
+-----------------------------------------------+-------------------------+
--- Time Elapsed: 65.551062ms ---
```

This is example output for advanced parameters.

```
admin@TPVM:~$ efa fabric setting show --advanced
+--------------------------------+------------------+
|              NAME              |      VALUE       |
+--------------------------------+------------------+
| Fabric Name                    | default          |
+--------------------------------+------------------+
| Link IP Range                  | 10.10.10.0/23    |
+--------------------------------+------------------+
| Loopback IP Range              | 172.32.254.0/24  |
+--------------------------------+------------------+
| Loopback Port Number           | 1                |
+--------------------------------+------------------+
| VTEP Loopback Port Number      | 2                |
+--------------------------------+------------------+
| Spine ASN Block                | 64512            |
+--------------------------------+------------------+
| Leaf ASN Block                 | 65000-65534      |
+--------------------------------+------------------+
| P2P IP Type                    | numbered         |
+--------------------------------+------------------+
| Any cast MAC                   | 0201.0101.0101   |
+--------------------------------+------------------+
| IPV6 Any cast MAC              | 0201.0101.0102   |
+--------------------------------+------------------+
| ARP Aging Timeout              | 300              |
+--------------------------------+------------------+
| MAC Aging Timeout              | 1800             |
+--------------------------------+------------------+
| MAC Aging Conversational       | 300              |
| Timeout                        |                  |
+--------------------------------+------------------+
| MAC Move Limit                 | 20               |
+--------------------------------+------------------+
| Duplicate MAC Timer            | 5                |
+--------------------------------+------------------+
| Duplicate MAC Timer MAX Count  | 3                |
+--------------------------------+------------------+
| Configure Overlay Gateway      | Yes              |
+--------------------------------+------------------+
| BFD Tx                         | 300              |
+--------------------------------+------------------+
| BFD Rx                         | 300              |
+--------------------------------+------------------+
| BFD Multiplier                 | 3                |
+--------------------------------+------------------+
| BGP MultiHop                   | 2                |
+--------------------------------+------------------+
| MaxPaths                       | 8                |
+--------------------------------+------------------+
| AllowAsIn                      | 0                |
+--------------------------------+------------------+
| MTU                            | 9216             |
+--------------------------------+------------------+
| IPMTU                          | 9100             |
+--------------------------------+------------------+
| Leaf PeerGroup                 | spine-group      |
+--------------------------------+------------------+
| Spine PeerGroup                | leaf-group       |
+--------------------------------+------------------+
| MCT Link IP Range              | 20.20.20.0/24    |
+--------------------------------+------------------+
| MCT PortChannel                | 1024             |
+--------------------------------+------------------+
| Routing MCT PortChannel        | 64               |
+--------------------------------+------------------+
| Control Vlan                   | 4090             |
+--------------------------------+------------------+
| Control VE                     | 4090             |
+--------------------------------+------------------+
```

```
| VNI Auto Map                    | Yes             |
+---------------------------------+-----------------+
--- Time Elapsed: 69.926589ms ---
```

NOTE

For CLI details, see the **efa fabric setting show** command in Appendix D: EFA Commands on page 119.

Modifying fabric settings

This use case addresses the modification of parameters following the initial default configuration.

The following example uses the **efa fabric setting update** command to modify default parameters.

```
$ efa fabric setting update --link_ip_range 10.10.110.0/23 --loopback_ip_range 172.32.244.0/24 --
loopback_port_number=3 --vtep_loopback_port_number=4 --spine_asn_block=61000 --leaf_asn_block 64000-66534
```

NOTE

For CLI details, see the **efa fabric setting update** command in Appendix D: EFA Commands on page 119.

Disabling optional features

This use case addresses the disabling of optional features.

The following example uses the **esa fabric setting update --configure_overlay_gateway yes** command to modify default parameters.

```
$ esa fabric setting update --configure_overlay_gateway no
```

NOTE

For CLI details, see the **efa fabric setting update** command in Appendix D: EFA Commands on page 119.

Fetching execution status

This use case addresses the fetching of the execution status of the previous CLI execution of the application.

The following example uses the **efa execution show** command to fetch current status.

```
$ efa execution show
+--------------------------------------+---------------+----------------------+---------------------------
+---------------------------+
|                  ID                  |    COMMAND    |        STATUS        |        START TIME
|         END TIME          |
+--------------------------------------+---------------+----------------------+---------------------------
+---------------------------+
| 173c0fa4-8cc5-4beb-bf15-ccf900071c5b | fabric show   | Completed(54.2014ms) | 2018-01-25T10:36:46+05:30 |
2018-01-25T10:36:46+05:30 |

| 68fec530-b7da-4606-a7a9-b4648019dbfe | configure add | Failed(15.1241363s)  | 2018-01-22T14:29:09+05:30 |
2018-01-22T14:29:25+05:30 |

+--------------------------------------+---------------+----------------------+---------------------------
+---------------------------+
```

NOTE

For CLI details, see the **efa execution show** command in Appendix D: EFA Commands on page 119.

Fetching execution details

This use case addresses the fetching of the execution details for a specified CLI execution of the application.

The following example uses the **efa execution show** command to fetch current status.

```
$ efa execution show –execution-id 68fec530-b7da-4606-a7a9-b4648019dbfe
"id": "68fec530-b7da-4606-a7a9-b4648019dbfe"
command: configure add
parameters: {spines: leaves:[“10.24.73.204”,”1.24.73.205”]}
duration: 768ms
errors: “No Spine devices”
```

NOTE

For CLI details, see the **efa execution show** command in Appendix D: EFA Commands on page 119.

Updating device credentials

You can update the credentials of existing devices in an IP Fabric.

This example updates all devices in the fabric.

```
$ efa device credentials update
```

NOTE

For CLI details, see the **efa device credentials update** command in Appendix D: EFA Commands on page 119.

Using supportSave

You can create a supportSave .zip package that contains the EFA database and execution logs.

The database and execution logs are available in `/var/efa/efa_SS.zip`.

supportSave requires root privileges. The following illustrates the execution of this feature.

```
admin@TPVM:~$ sudo su

[sudo] password for admin:

root@TPVM:/home/admin#

root@TPVM:/home/admin# efa supportsave

Version : 1.1.0

Time Stamp: 18-10-03:10:31:42

--- Time Elapsed: 233.458121ms ---
```

admin@TPVM:~$ sudo su

[sudo] password for admin:

root@TPVM:/home/admin#

root@TPVM:/home/admin# efa supportsave

Version : 1.1.1

Time Stamp: 18-10-03:10:31:42

--- Time Elapsed: 233.458121ms ---

Fabric events

This use case describes the handling of various fabric events by means of the ZTP script and the application.

The application does not maintain the fabric status of individual switches in the fabric. Instead, it relies on the ZTP script to call the appropriate REST APIs. Specific behaviors are presented below.

Addition of a new switch

When a new node is added to the fabric, the ZTP script invokes a **configure add** REST API call

Fabric link flapping between switches following configuration

The configurations pushed by the application are persistent on the switches. Link flapping subsequent to this is not consequential.

Node rejoin following configuration

Any node leaving the fabric and rejoining has no effect, as the configuration is pushed to the respective switches.

Missing or incorrect fabric links during switch additions

If there are missing links during a node addition, they are flagged by the application. The user must correct the topology and rerun the **efa fabric configure** command by means of either the CLI or REST.

Removal of a switch from the fabric

The user must use the CLI or REST **efa fabric deconfigure** command to remove a switch from the fabric

Upgrading to EFA 1.1.0 with TPVM Ubuntu version 16.04_SLXS

Do the following to upgrade the EFA application.

In the previous version, SLX-OS 17s.1.02b used TPVM Ubuntu version 14.04 on an SLX-9240. This task upgrades the version to 16.04.

1. Log in to TPVM with the TPVM IP address.

```
$ ssh -l root <TPVM_IP>
```

2. Copy the EFA database and logs backup to an external server.

```
$ service efa-server stop
$ scp /var/efa/efa.db <server_EFA_DB_Location>
$ scp /var/log/efa/efa.log <server_EFA_Log_Location>
```

3. Stop and uninstall TPVM.

```
$ tpvm stop
$ tpvm uninstall
```

4. With the device upgraded to 18s.1.01, execute the **efa deploy** command.

```
$ efa deploy
```

5. Log in to TPVM and verify the TPVM version.

```
$ ssh -l admin <TPVM_IP>
```

6. Restore the EFA database and logs.

```
$ sudo su <-- Provide TPVM root password
$ systemctl stop efa-server
$ mv /var/efa/efa.db /tmp/
$ mv /var/log/efa/efa.log /tmp/
$ scp <server_EFA_DB_Location> /var/efa/
$ scp <server_EFA_Log_Location> /var/log/efa/
$ systemctl start efa-server
```

7. Verify that EFA is running and verify the version.

```
$ ps -ef | grep -i efa
```

   The EFA version should be 1.1.0.

8. Before executing the **efa deconfigure** command, execute the **efa configure** command at least once following the upgrade.

Appendix A: Supported BGP EVPN Commands

Configuration commands supporting BGP EVPN

The following configuration commands support BGP EVPN.

For more details, refer to the *Command Reference*.

Global configuration mode

- evpn

BGP configuration mode

- address-family l2vpn evpn

BGP address-family L2VPN EVPN configuration mode

- clear bgp evpn neighbor
- clear bgp evpn routes
- client-to-client-reflection
- graceful-restart (BGP)
- neighbor activate
- neighbor allowas-in
- neighbor enable-peer-as-check
- neighbor encapsulation
- neighbor maximum-prefix
- neighbor next-hop-unchanged
- neighbor route-map
- neighbor route-reflector-client
- neighbor send-community
- retain route-target all
- vtep-discovery

EVPN instance configuration mode

- duplicate-mac-timer

- evpn
- **rd auto (EVPN)**
- **route-target (EVPN)**
- **vlan (EVPN)**
-
-

VNI configuration mode

- **rd (VNI)**
- **route-target (VNI)**

VRF configuration mode

- **rd (VRF)**
- **route-target (VRF)**
- **vlan (VRF)**
-

Port-channel configuration mode

- **esi**

VXLAN overlay-gateway site configuration mode

- **mac-learning protocol bgp**

Show commands supporting BGP EVPN

The following show commands support BGP EVPN.

For more details, refer to the *Command Reference*.

- **show bgp evpn dampened-routes**
- **show bgp evpn interface port-channel**
- **show bgp evpn interface tunnel**
- **show bgp evpn l2route detail**
- **show bgp evpn l2route next-hop**
- **show bgp evpn l2route summary**
- **show bgp evpn l2route type arp**
- **show bgp evpn l2route type auto-discovery**
- **show bgp evpn l2route type ethernet-segment**
- **show bgp evpn l2route type inclusive-multicast**
- **show bgp evpn l2route type mac**

- show bgp evpn l2route type nd
- show bgp evpn l2route unreachable
- show bgp evpn l3vni
- show bgp evpn neighbors
- show bgp evpn neighbors advertised-routes
- show bgp evpn neighbors advertised-routes detail
- show bgp evpn neighbors advertised-routes type
- show bgp evpn neighbors routes best
- show bgp evpn neighbors routes detail
- show bgp evpn neighbors routes not-installed-best
- show bgp evpn neighbors routes type
- show bgp evpn neighbors routes unreachable
- show bgp evpn neighbors routes-summary
- show bgp evpn routes
- show bgp evpn routes best
- show bgp evpn routes detail
- show bgp evpn routes local
- show bgp evpn routes next-hop
- show bgp evpn routes no-best
- show bgp evpn routes not-installed-best
- show bgp evpn routes rd
- show bgp evpn routes rd type
- show bgp evpn routes summary
- show bgp evpn routes type arp
- show bgp evpn routes type auto-discovery
- show bgp evpn routes type ethernet-segment
- show bgp evpn routes type inclusive-multicast
- show bgp evpn routes type ipv4-prefix
- show bgp evpn routes type ipv6-prefix
- show bgp evpn routes type mac
- show bgp evpn routes type nd
- show bgp evpn routes unreachable
- show bgp evpn summary
- show mac-address-table
- show mac-address-table count evpn
- show mac-address-table evpn

Appendix B: Sample BGP EVPN configuration files

Sample BGP EVPN superspine configuration using eBGP

The following example is a sample BGP EVPN superspine configuration using eBGP in an IP Fabric. The **neighbor remote-as** command is used so that the switches are configured to be in one autonomous system (AS). The switches are configured to advertise routes to other eBGP peers leaving the next-hop attribute unchanged using the **neighbor next-hop-unchanged** command.

> NOTE
> This configuration sample demonstrates BGP EVPN configuration using eBGP on the superspine switch in an IP Fabric. For full configuration details for configuring an IP Fabric involving all the steps required, refer to Configuring a Extreme IP Fabric with Optional BGP EVPN Overlay on page 57. For more details on the commands used in this sample, refer to the *Command Reference*. For more information on eBGP-based BGP EVPN, refer to eBGP-based BGP EVPN on page 26.

```
router bgp
 local-as 1
! IPv4/IPv6 BGP Sessions to Spine switches
 neighbor 10.2.1.1 remote-as 2
 neighbor 10.3.1.1 remote-as 2
address-family ipv4 unicast
 maximum-paths 8
 next-hop-recursion
 redistribute connected
address-family l2vpn evpn
 retain route-target all
! Activate EVPN Session to Spine switches
 neighbor 10.2.1.1 activate
 neighbor 10.3.1.1 activate
 neighbor 10.2.1.1 next-hop-unchanged
 neighbor 10.3.1.1 next-hop-unchanged
```

Sample BGP EVPN spine configuration using eBGP

The following example is a sample BGP EVPN spine configuration using eBGP in an IP Fabric. There are two spine switches, Spine1 and Spine2. The spine switches are configured to be in one autonomous system (AS) using the **neighbor remote-as** command and are configured to advertise routes to other eBGP peers leaving the next-hop attribute unchanged using the **neighbor next-hop-unchanged** command.

NOTE

This configuration sample demonstrates BGP EVPN configuration using eBGP on two spine switches in an IP Fabric. For full configuration details for configuring an IP fabric involving all the steps required, refer to Configuring a Extreme IP Fabric with Optional BGP EVPN Overlay on page 57. For more details on the commands used in this sample, refer to the *Command Reference*.

Spine1

```
router bgp
 local-as 2
!BGP Sessions to the SuperSpine
 neighbor 10.2.1.0 remote-as 1
!IPv4/IPv6 BGP Sessions to the leaf switches
 neighbor 10.6.0.1 remote-as 3
 neighbor 10.7.0.1 remote-as 3
 neighbor 10.8.0.1 remote-as 3
 neighbor 10.9.0.1 remote-as 3
address-family ipv4 unicast
 maximum-paths 8
 next-hop-recursion
 redistribute connected
 neighbor 10.2.1.0 allowas-in 1
address-family l2vpn evpn
 retain route-target all
 neighbor 10.2.1.0 activate
 neighbor 10.2.1.0 allowas-in 1
 neighbor 10.2.1.0 next-hop-unchanged
 neighbor 10.6.0.1 activate
 neighbor 10.6.0.1 next-hop-unchanged
 neighbor 10.7.0.1 activate
 neighbor 10.7.0.1 next-hop-unchanged
 neighbor 10.8.0.1 activate
 neighbor 10.8.0.1 next-hop-unchanged
 neighbor 10.9.0.1 activate
 neighbor 10.9.0.1 next-hop-unchanged
```

Spine2

```
router bgp
 local-as 2
!BGP Sessions to the SuperSpine
 neighbor 10.3.1.0 remote-as 1
!IPv4/IPv6 BGP Sessions to the leaf switches
 neighbor 10.6.1.1 remote-as 3
 neighbor 10.7.1.1 remote-as 3
 neighbor 10.8.1.1 remote-as 3
 neighbor 10.9.1.1 remote-as 3
address-family ipv4 unicast
 maximum-paths 8
 next-hop-recursion
 redistribute connected
 neighbor 10.3.1.0 allowas-in 1
address-family l2vpn evpn
 retain route-target all
 neighbor 10.3.1.0 activate
 neighbor 10.3.1.0 allowas-in 1
 neighbor 10.3.1.0 next-hop-unchanged
 neighbor 10.6.1.1 activate
 neighbor 10.6.1.1 next-hop-unchanged
 neighbor 10.7.1.1 activate
 neighbor 10.7.1.1 next-hop-unchanged
 neighbor 10.8.1.1 activate
 neighbor 10.8.1.1 next-hop-unchanged
 neighbor 10.9.1.1 activate
 neighbor 10.9.1.1 next-hop-unchanged
```

Sample BGP EVPN leaf configuration using eBGP

The following example is a sample BGP EVPN leaf configuration using eBGP in an IP Fabric. The AS_PATH check function is disabled for the leaf switch using the **neighbor allowas-in** command so that route updates from the spine layer are not discarded by the leaf.

> NOTE
> This configuration sample demonstrates BGP EVPN configuration using eBGP on a leaf switch in an IP Fabric. For full configuration details for configuring an IP fabric involving all the steps required, refer to Configuring a Extreme IP Fabric with Optional BGP EVPN Overlay on page 57. For more details on the commands used in this sample, refer to the *Command Reference*.

```
evpn myinstance
 rd auto
 route-target both auto
 bridge-domain add 1-500
 vlan add 101-108
 vlan add 121-128
 vlan add 141-148
 vlan add 161-168
 vlan add 120
 vlan add 140
 vlan add 160
 vlan add 180
!
router bgp
 local-as 3
neighbor 10.6.0.0 remote-as 2
neighbor 10.7.0.0 remote-as 2
 !
 address-family ipv4 unicast
  maximum-paths 8
  next-hop-recursion
  redistribute connected
  neighbor 10.6.0.0 allowas-in 1
  neighbor 10.7.0.0 allowas-in 1
 !
 address-family l2vpn evpn
  neighbor 10.6.0.0 activate
  neighbor 10.7.0.0 activate
  neighbor 10.6.0.0 allowas-in 1
  neighbor 10.7.0.0 allowas-in 1
  neighbor 10.6.0.0 next-hop-unchanged
  neighbor 10.7.0.0 next-hop-unchanged
 !
overlay-gateway Leaf1
 type layer2-extension
 ip interface Loopback 1
 map vni auto
 activate
```

Sample BGP EVPN superspine configuration using iBGP

The following example is a sample BGP EVPN superspine configuration using iBGP in an IP Fabric.

> **NOTE**
> This configuration sample demonstrates BGP EVPN configuration using iBGP on the superspine in an IP Fabric. For full configuration details for configuring an IP Fabric involving all the steps required, refer to Configuring a Extreme IP Fabric with Optional BGP EVPN Overlay on page 57. For more details on the commands used in this sample, refer to the *Command Reference*. For more information on eBGP-based BGP EVPN, refer to the section iBGP-based BGP EVPN on page 27.

```
router ospf
 area 0
 redistribute connected
!
router bgp
 local-as 3
! IPv4/IPv6 BGP Sessions to Spine switches
 neighbor 10.4.1.1 remote-as 3
 neighbor 10.5.1.1 remote-as 3
address-family ipv4 unicast
 maximum-paths 8
 next-hop-recursion
 redistribute connected
address-family l2vpn evpn
 retain route-target all
! Activate EVPN Session to Spine switches
 neighbor 10.4.1.1 activate
 neighbor 10.5.1.1 activate
 neighbor 10.4.1.1 route-reflector-client
 neighbor 10.5.1.1 route-reflector-client
int eth 4/11
 ip address 10.4.0.0/31
 ipv6 address 1000:1:4::/127
 ip ospf area 0
 no shutdown
 !
int eth 4/12
 ip address 10.5.0.0/31
 ipv6 address 1000:1:5::/127
 ip ospf area 0
 no shutdown
```

Sample BGP EVPN spine configuration using iBGP

The following example is a sample BGP EVPN spine configuration using iBGP in an IP Fabric. The spine switch is configured as a route reflector (RR) and is configured to accept received updates containing at least one route target. Neighboring leaf switches are configured as route reflector (RR) clients using the **route-reflector-client** command.

> **NOTE**
> This configuration sample demonstrates BGP EVPN configuration using iBGP on a spine switch in an IP Fabric. For full configuration details for configuring an IP Fabric involving all the steps required, refer to Configuring a Extreme IP Fabric with Optional BGP EVPN Overlay on page 57.

> **NOTE**
> For more details on the commands used in this sample, refer to the *Command Reference*.

```
router ospf
 area 0
```

```
 redistribute connected
!
router bgp
 local-as 3
 cluster-id ipv4-address 10.4.4.4
!BGP Sessions to the SuperSpine
!IPv4/IPv6 BGP Sessions to the leaf switches
 neighbor 10.0.0.10 remote-as 3
 neighbor 10.0.0.10 update-source loopback 1
 neighbor 10.0.0.11 remote-as 3
 neighbor 10.0.0.11 update-source loopback 1
 neighbor 10.0.0.12 remote-as 3
 neighbor 10.0.0.12 update-source loopback 1
 neighbor 10.0.0.13 remote-as 3
 neighbor 10.0.0.13 update-source loopback 1
address-family ipv4 unicast
 maximum-paths 8
 next-hop-recursion
 redistribute connected
 client-to-client-reflection
 neighbor 10.0.0.10 route-reflector-client
 neighbor 10.0.0.11 route-reflector-client
 neighbor 10.0.0.12 route-reflector-client
 neighbor 10.0.0.13 route-reflector-client
address-family l2vpn evpn
 retain route-target all
 neighbor 10.4.1.0 activate
 neighbor 10.4.1.0 route-reflector-client
 neighbor 10.0.0.10 activate
 neighbor 10.0.0.11 activate
 neighbor 10.0.0.12 activate
 neighbor 10.0.0.13 activate
!
```

Sample BGP EVPN leaf configuration using iBGP

The following example is a sample BGP EVPN leaf configuration using iBGP in an IP Fabric. The leaf switch is configured to communicate with a neighbor through a specified loopback interface.

> NOTE
> This configuration sample demonstrates BGP EVPN configuration using iBGP on a leaf in an IP Fabric. For full configuration details for configuring an IP Fabric involving all the steps required, refer to Configuring a Extreme IP Fabric with Optional BGP EVPN Overlay on page 57. For more details on the commands used in this sample, refer to the *Command Reference*.

```
evpn myinstance
 rd auto
 route-target both auto ignore-as
 bridge-domain add 1-500
 vlan add 101-108
 vlan add 121-128
 vlan add 141-148
 vlan add 161-168
 vlan add 120
 vlan add 140
 vlan add 160
 vlan add 180
!
router bgp
 local-as 3
neighbor 10.0.0.4 remote-as 3
neighbor 10.0.0.4 update-source loopback 1
neighbor 10.0.0.5 remote-as 3
neighbor 10.0.0.5 update-source loopback 1
 !
 address-family ipv4 unicast
  maximum-paths 8
```

```
  next-hop-recursion
  redistribute connected
 !
 address-family l2vpn evpn
  neighbor 10.0.0.4 activate
  neighbor 10.0.0.5 activate
  neighbor 10.0.0.4 next-hop-unchanged
  neighbor 10.0.0.5 next-hop-unchanged
 !
router ospf
 area 0
 redistribute connected
!
overlay-gateway Leaf10
 type layer2-extension
 ip interface Loopback 1
 map vni auto
 activate
int eth 0/1
 ip address 10.10.0.1/31
 ipv6 address 1000:4:10::1/127
 ip ospf area 0
 no shut
!
int eth 0/2
 ip address 10.10.0.2/31
 ipv6 address 1000:5:10::1/127
 ip ospf area 0
 no shut
!
int eth 0/16
 switchport
 switchport mode trunk
 switchport trunk allowed vlan add 101-108
 switchport trunk allowed vlan add 121-128
 no shut
!
int eth 0/17
 switchport trunk allowed vlan add 141-148
 switchport trunk allowed vlan add 161-168
 no shut
!
```

Appendix C: Sample Topology Configuration Files

Leaf

The following configuration file is for the leaf Leaf1.

```
vlan 101-108
vlan 120
vlan 121-128
vlan 140
vlan 141-148
vlan 160
vlan 161-168
vlan 180
host-name Leaf1
ip anycast-gateway-mac 0000.ABBA.BABA
ipv6 anycast-gateway-mac 0000.ABBA.ABBA
vrf red
 rd 6.0.0.6:1
 evpn irb ve 120
 address-family ipv4 unicast
  route-target import 2:2 evpn
  route-target export 2:2 evpn
 address-family ipv6 unicast
  route-target import 2:2 evpn
  route-target export 2:2 evpn
!
vrf blue
 rd 6.0.0.6:2
 evpn irb ve 140
 address-family ipv4 unicast
  route-target import 2:2 evpn
  route-target export 2:2 evpn
 address-family ipv6 unicast
  route-target import 2:2 evpn
  route-target export 2:2 evpn
!
vrf green
 rd 6.0.0.6:3
 evpn irb ve 160
 address-family ipv4 unicast
  route-target import 2:2 evpn
  route-target export 2:2 evpn
 address-family ipv6 unicast
  route-target import 2:2 evpn
  route-target export 2:2 evpn
!
vrf yellow
 rd 6.0.0.6:4
 evpn irb ve 180
 address-family ipv4 unicast
  route-target import 2:2 evpn
  route-target export 2:2 evpn
 address-family ipv6 unicast
  route-target import 2:2 evpn
  route-target export 2:2 evpn
```

```
!
!!!!!Loopback interface
int loopback 1
 ip address 6.0.0.6/32
 ipv6 address 1000:6::6/128
 no shut
!
int ve 101
 vrf forwarding red
 ip anycast-address 101.1.1.254/24
 ipv6 anycast-address 101:1:1::254/64
 no shutdown
 !
int ve 102
 vrf forwarding red
 ip anycast-address 102.1.1.254/24
 ipv6 anycast-address 102:1:1::254/64
 no shutdown
 !
int ve 103
 vrf forwarding red
 ip anycast-address 103.1.1.254/24
 ipv6 anycast-address 103:1:1::254/64
 no shutdown
 !
int ve 104
 vrf forwarding red
 ip anycast-address 104.1.1.254/24
 ipv6 anycast-address 104:1:1::254/64
 no shutdown
 !
int ve 105
 vrf forwarding red
 ip anycast-address 105.1.1.254/24
 ipv6 anycast-address 105:1:1::254/64
 no shutdown
 !
int ve 106
 vrf forwarding red
 ip anycast-address 106.1.1.254/24
 ipv6 anycast-address 106:1:1::254/64
 no shutdown
 !
int ve 107
 vrf forwarding red
 ip anycast-address 107.1.1.254/24
 ipv6 anycast-address 107:1:1::254/64
 no shutdown
 !
int ve 108
 vrf forwarding red
 ip anycast-address 108.1.1.254/24
 ipv6 anycast-address 108:1:1::254/64
 no shutdown
 !
int ve 121
 vrf forwarding blue
 ip anycast-address 121.1.1.254/24
 ipv6 anycast-address 121:1:1::254/64
 no shutdown
 !
int ve 122
 vrf forwarding blue
 ip anycast-address 122.1.1.254/24
 ipv6 anycast-address 122:1:1::254/64
 no shutdown
 !
int ve 123
 vrf forwarding blue
 ip anycast-address 123.1.1.254/24
 ipv6 anycast-address 123:1:1::254/64
 no shutdown
```

```
 !
int ve 124
 vrf forwarding blue
 ip anycast-address 124.1.1.254/24
 ipv6 anycast-address 124:1:1::254/64
 no shutdown
 !
int ve 125
 vrf forwarding blue
 ip anycast-address 125.1.1.254/24
 ipv6 anycast-address 125:1:1::254/64
 no shutdown
 !
int ve 126
 vrf forwarding blue
 ip anycast-address 126.1.1.254/24
 ipv6 anycast-address 126:1:1::254/64
 no shutdown
 !
int ve 127
 vrf forwarding blue
 ip anycast-address 127.1.1.254/24
 ipv6 anycast-address 127:1:1::254/64
 no shutdown
 !
int ve 128
 vrf forwarding blue
 ip anycast-address 128.1.1.254/24
 ipv6 anycast-address 128:1:1::254/64
 no shutdown
 !
int ve 141
 vrf forwarding green
 ip anycast-address 141.1.1.254/24
 ipv6 anycast-address 141:1:1::254/64
 no shutdown
 !
int ve 142
 vrf forwarding green
 ip anycast-address 142.1.1.254/24
 ipv6 anycast-address 142:1:1::254/64
 no shutdown
 !
int ve 143
 vrf forwarding green
 ip anycast-address 143.1.1.254/24
 ipv6 anycast-address 143:1:1::254/64
 no shutdown
 !
int ve 144
 vrf forwarding green
 ip anycast-address 144.1.1.254/24
 ipv6 anycast-address 144:1:1::254/64
 no shutdown
 !
int ve 145
 vrf forwarding green
 ip anycast-address 145.1.1.254/24
 ipv6 anycast-address 145:1:1::254/64
 no shutdown
 !
int ve 146
 vrf forwarding green
 ip anycast-address 146.1.1.254/24
 ipv6 anycast-address 146:1:1::254/64
 no shutdown
 !
int ve 147
 vrf forwarding green
 ip anycast-address 147.1.1.254/24
 ipv6 anycast-address 147:1:1::254/64
 no shutdown
```

```
 !
int ve 148
 vrf forwarding green
 ip anycast-address 148.1.1.254/24
 ipv6 anycast-address 148:1:1::254/64
 no shutdown
 !
int ve 161
 vrf forwarding yellow
 ip anycast-address 161.1.1.254/24
 ipv6 anycast-address 161:1:1::254/64
 no shutdown
 !
int ve 162
 vrf forwarding yellow
 ip anycast-address 162.1.1.254/24
 ipv6 anycast-address 162:1:1::254/64
 no shutdown
 !
int ve 163
 vrf forwarding yellow
 ip anycast-address 163.1.1.254/24
 ipv6 anycast-address 163:1:1::254/64
 no shutdown
 !
int ve 164
 vrf forwarding yellow
 ip anycast-address 164.1.1.254/24
 ipv6 anycast-address 164:1:1::254/64
 no shutdown
 !
int ve 165
 vrf forwarding yellow
 ip anycast-address 165.1.1.254/24
 ipv6 anycast-address 165:1:1::254/64
 no shutdown
 !
int ve 166
 vrf forwarding yellow
 ip anycast-address 166.1.1.254/24
 ipv6 anycast-address 166:1:1::254/64
 no shutdown
 !
int ve 167
 vrf forwarding yellow
 ip anycast-address 167.1.1.254/24
 ipv6 anycast-address 167:1:1::254/64
 no shutdown
 !
int ve 168
 vrf forwarding yellow
 ip anycast-address 168.1.1.254/24
 ipv6 anycast-address 168:1:1::254/64
 no shutdown
 !
int ve 120
 vrf forwarding red
    ipv6 address use-link-local-only
 no shut
!
int ve 140
 vrf forwarding blue
    ipv6 address use-link-local-only
 no shut
!
int ve 160
 vrf forwarding green
    ipv6 address use-link-local-only
 no shut
!
int ve 180
 vrf forwarding yellow
```

```
    ipv6 address use-link-local-only
 no shut
!
evpn myinstance
 rd auto
 route-target both auto ignore-as
 bridge-domain add 1-500
 vlan add 101-108
 vlan add 121-128
 vlan add 141-148
 vlan add 161-168
 vlan add 120
 vlan add 140
 vlan add 160
 vlan add 180
!
router bgp
 local-as 3
 neighbor 2.6.0.0 remote-as 2
 neighbor 1000:2:6:: remote-as 2
 neighbor 3.6.0.0 remote-as 2
 neighbor 1000:3:6:: remote-as 2
 !
 address-family ipv4 unicast
  maximum-paths 8
  next-hop-recursion
  redistribute connected
  neighbor 2.6.0.0 allowas-in 1
  neighbor 3.6.0.0 allowas-in 1
 !
 address-family ipv6 unicast
  maximum-paths 8
  next-hop-recursion
  redistribute connected
  neighbor 1000:2:6:: activate
  neighbor 1000:3:6:: activate
  neighbor 1000:2:6:: allowas-in 1
  neighbor 1000:3:6:: allowas-in 1
 !
 address-family l2vpn evpn
  neighbor 2.6.0.0 activate
  neighbor 3.6.0.0 activate
  neighbor 2.6.0.0 allowas-in 1
  neighbor 3.6.0.0 allowas-in 1
  neighbor 2.6.0.0 next-hop-unchanged
  neighbor 3.6.0.0 next-hop-unchanged
 !
address-family ipv4 unicast vrf red
 redistribute connected
address-family ipv4 unicast vrf blue
 redistribute connected
address-family ipv4 unicast vrf green
 redistribute connected
address-family ipv4 unicast vrf yellow
 redistribute connected
address-family ipv6 unicast vrf red
 redistribute connected
address-family ipv6 unicast vrf blue
 redistribute connected
address-family ipv6 unicast vrf green
 redistribute connected
address-family ipv6 unicast vrf yellow
 redistribute connected
overlay-gateway Leaf6
 type layer2-extension
 ip interface Loopback 1
 map vni auto
 activate
int eth 1/1
 ip address 2.6.0.1/31
 ipv6 address 1000:2:6::1/127
 no shut
```

```
!
int eth 1/5
 ip address 3.6.0.1/31
 ipv6 address 1000:3:6::1/127
 no shut
!
int eth 1/16
 switchport
 switchport mode trunk
 switchport trunk allowed vlan add 101-108
 switchport trunk allowed vlan add 121-128
 no shut
!
int eth 1/17
 switchport
 switchport mode trunk
 switchport trunk allowed vlan add 141-148
 switchport trunk allowed vlan add 161-168
 no shut
!
```

Spine

The following configuration file is for the spine Spine1.

```
host-name Spine1
!!!!!Loopback interface
int loopback 1
 ip address 2.0.0.2/32
 ipv6 address 1000:2::2/128
 no shut
!
router bgp
 local-as 2
!BGP Sessions to the SuperSpine
 neighbor 1.2.1.0 remote-as 1
 neighbor 1000:1:2:1:: remote-as 1
!IPv4/IPv6 BGP Sessions to the leaf switches
 neighbor 2.6.0.1 remote-as 3
 neighbor 1000:2:6::1 remote-as 3
 neighbor 2.7.0.1 remote-as 3
 neighbor 1000:2:7::1 remote-as 3
 neighbor 2.8.0.1 remote-as 3
 neighbor 1000:2:8::1 remote-as 3
 neighbor 2.9.0.1 remote-as 3
 neighbor 1000:2:9::1 remote-as 3
address-family ipv4 unicast
 maximum-paths 8
 next-hop-recursion
 redistribute connected
 neighbor 1.2.1.0 allowas-in 1
address-family ipv6 unicast
 maximum-paths 8
 next-hop-recursion
 redistribute connected
 neighbor 1000:1:2:1:: activate
 neighbor 1000:1:2:1:: allowas-in 1
 neighbor 1000:2:6::1 activate
 neighbor 1000:2:7::1 activate
 neighbor 1000:2:8::1 activate
 neighbor 1000:2:9::1 activate
address-family l2vpn evpn
 retain route-target all
 neighbor 1.2.1.0 activate
 neighbor 1.2.1.0 allowas-in 1
 neighbor 1.2.1.0 next-hop-unchanged
 neighbor 2.6.0.1 activate
 neighbor 2.6.0.1 next-hop-unchanged
```

```
 neighbor 2.7.0.1 activate
 neighbor 2.7.0.1 next-hop-unchanged
 neighbor 2.8.0.1 activate
 neighbor 2.8.0.1 next-hop-unchanged
 neighbor 2.9.0.1 activate
 neighbor 2.9.0.1 next-hop-unchanged
!
int eth 1/33
 ip address 1.2.1.1/31
 ipv6 address 1000:1:2:1::1/127
 no shutdown
 !
int eth 1/34
 ip address 1.2.2.1/31
 ipv6 address 1000:1:2:2::1/127
 no shutdown
 !
int eth 1/35
 ip address 1.2.3.1/31
 ipv6 address 1000:1:2:3::1/127
 no shutdown
 !
int eth 1/36
 ip address 1.2.4.1/31
 ipv6 address 1000:1:2:4::1/127
 no shutdown
 !
int eth 1/37
 ip address 1.2.5.1/31
 ipv6 address 1000:1:2:5::1/127
 no shutdown
 !
int eth 1/1
 ip address 2.6.0.0/31
 ipv6 address 1000:2:6::0/127
 no shutdown
 !
int eth 1/4
 ip address 2.7.0.0/31
 ipv6 address 1000:2:7::0/127
 no shutdown
 !
int eth 1/3
 ip address 2.8.0.0/31
 ipv6 address 1000:2:8::0/127
 no shutdown
 !
int eth 1/2
 ip address 2.9.0.0/31
 ipv6 address 1000:2:9::0/127
 no shutdown
 !
```

SuperSpine

The following configuration file is for the spine SuperSpine.

```
 host-name SuperSpine
!!!!!Loopback interface
int loopback 1
 ip address 1.0.0.1/32
 ipv6 address 1000:1::1/128
 no shut
!
router bgp
 local-as 1
! IPv4/IPv6 BGP Sessions to Spine switches
 neighbor 1.2.1.1 remote-as 2
```

```
 neighbor 1.3.1.1 remote-as 2
 neighbor 1000:1:2:1::1 remote-as 2
 neighbor 1000:1:3:1::1 remote-as 2
address-family ipv4 unicast
 maximum-paths 8
 next-hop-recursion
 redistribute connected
address-family ipv6 unicast
 maximum-paths 8
 next-hop-recursion
 redistribute connected
 neighbor 1000:1:2:1::1 activate
 neighbor 1000:1:3:1::1 activate
address-family l2vpn evpn
 retain route-target all
! Activate EVPN Session to Spine switches
 neighbor 1.2.1.1 activate
 neighbor 1.3.1.1 activate
 neighbor 1.2.1.1 next-hop-unchanged
 neighbor 1.3.1.1 next-hop-unchanged
int eth 1/1
 ip address 1.2.1.0/31
 ipv6 address 1000:1:2:1::/127
 no shutdown
 !
int eth 1/5
 ip address 1.3.1.0/31
 ipv6 address 1000:1:3:1::/127
 no shutdown
 !
```

Appendix D: EFA Commands

efa deploy

Deploys the EFA application.

Syntax

efa deploy [**--help** | **--password** *string* | **--username** *string*]

Command Default

This command has no defaults.

Parameters

--help
Provides help for command options.

--password *string*
Specifies a password.

--username *string*
Specifies a username.

Modes

TPVM configuration mode

Usage Guidelines

This command is executed in TPVM configuration mode on one of the SLX switches within the fabric.

ATTENTION
This command causes TPVM to reboot.

The username and password are optional. The defaults "root" and "fibranne" are used if not specified.

Examples

This example deploys EFA.

```
$ efa deploy username root password fibranne
```

History

Release version	Command history
17s.1.02b	This command was introduced.

efa debug clear-config

Clears the EFA configuration from the device.

Syntax

efa debug clear-config [**--device** *string* | **--help** | **--password** *string* | **--username** *string*]

Command Default

This command has no defaults.

Parameters

--device *string*
Specifies a comma-separated list of device IP addresses or hostnames.

--help
Provides help for command options.

--password *string*
Clears the password for a comma-separated list of device IP addresses or hostnames.

--username *string*
Clears the username for a comma-separated list of device IP addresses or hostnames.

Modes

TPVM configuration mode

Examples

This example clears all EFA configured parameters for three devices.

```
$ efa debug clear-config --device 10.25.225.160,10.25.225.161,10.25.225.162
```

History

Release version	Command history
17s.1.02b	This command was introduced.

efa device credentials update

Updates the credentials of existing devices in an IP Fabric. This should be used before deconfiguring or refreshing a device if the device credentials have been changed.

Syntax

efa device credentials update [**--device** *string* | **--help** | **--password** *string*| **--username** *string*]

Command Default

This command has no defaults.

Parameters

--device *string*
Specifies a comma-separated list of device IP addresses or hostnames.

--help
Provides help for command options.

--password *string*
Specifies the password for a comma-separated list of device IP addresses or hostnames.

--username *string*
Specifies the username for a comma-separated list of device IP addresses or hostnames.

Modes

TPVM configuration mode

Usage Guidelines

The keywords are required. Otherwise an error is thrown, as shown below.

```
Error: Required flag(s) "device", "password", "username" have/has not been set
```

Examples

This example updates credentials for all devices in the fabric.

```
$ efa device credentials updatedevice 10.2.3.4 password mypassword username myusername
```

History

Release version	Command history
17s.1.02b	This command was introduced.

efa execution show

Displays the status and execution details of previous CLI or REST executions of the application.

Syntax

efa execution show [**--id** *string*] | **limit** *int32* [**--status** [**all** | **failed** | **succeeded**]

Parameters

--id *string*

Filters execution according to the execution ID.

--limit *int32*

Limits the number of executions to be listed, where *int32* is an unsigned integer (default is 10).

--status

Specifies whether the status of execution shows failed or succeeded.

all

Specifies all executions. This is the default, returning both succeeded and failed executions.

failed

Specifies failed executions.

succeeded

Specifies succeded executions.

Modes

TPVM configuration mode

Usage Guidelines

Examples

This is example output without any arguments.

```
$ efa execution show
+--------------------------------------+---------------+----------------------+---------------------------
+--------------------------+
|                  ID                  |    COMMAND    |        STATUS        |        START TIME
|         END TIME          |
+--------------------------------------+---------------+----------------------+---------------------------
+--------------------------+
| 173c0fa4-8cc5-4beb-bf15-ccf900071c5b | fabric show   | Completed(54.2014ms) | 2018-01-25T10:36:46+05:30 |
2018-01-25T10:36:46+05:30 |

| 68fec530-b7da-4606-a7a9-b4648019dbfe | configure add | Failed(15.1241363s)  | 2018-01-22T14:29:09+05:30 |
2018-01-22T14:29:25+05:30 |

+--------------------------------------+---------------+----------------------+---------------------------
+--------------------------+
```

Output field	Description
ID	Execution ID of the CLI request
Command	CLI command that is executed
STATUS	Completed or Failed status of the command
START TIME	Time when the command is received
END TIME	Time when the command execution is completed

This is example output with an execution ID specified.

```
efa execution show --id 68fec530-b7da-4606-a7a9-b4648019dbfe

id: 68fec530-b7da-4606-a7a9-b4648019dbfe
command: configure add
parameters: {spines: leaves:["10.24.73.204","1.24.73.205"]}
status: Failed
duration: 768ms
errors: "No Spine devices"
```

Output field	Description
ID	Execution ID of the CLI request
Logs	Logs for the given execution ID (error logs are displayed by default)

History

The **efa execution show** command displays the following information.

Release version	Command history
17s.1.02b	This command was introduced.

efa fabric configure

Registers switches and their roles with a default IP Fabric.

Syntax

efa fabric configure { **--leaf** *list-of-leaf-ips* **--spine** *list-of-spine-ips* } [**--persist** | **--force**] [**--username** *string* **--password** *string*]

Command Default

No devices are added, deleted, or configured by default.

Parameters

--leaf *list-of-leaf-ips*

Specifies a comma-separated list of leaf management IP addresses.

--spine *list-of-spine-ips*

Specifies a comma-separated list of spine management IP addresses.

--force

Replaces any conflicting configurations needed for fabric formation on the switch with the one inferred by the application. This involves overwriting the following: ASNs based on a fabric template; IP addresses of physical, loopback, and VE interfaces; overlay gateway configuration; EVPN instance; and MCT cluster configuration.

--persist

Executes **copy running-config startup-config** on all leaf and spine devices.

username *string*

Specifies a username. The default credentials used by the application are "admin" and "password". This option can be used to override the default credentials. The credentials apply for all devices that are provided as part of the command. In case the devices have different credentials, they can be added by means of a separate command.

password *string*

Specifies a password.

Modes

TPVM configuration mode

Usage Guidelines

This command is executed in TPVM configuration mode on one of the SLX switches within the fabric.

Examples

This example adds leaf and spine nodes to the fabric and resolves conflicting configurations on the switch.

```
$ efa fabric configure --leaf 10.24.73.203,10.24.73.204,10.24.73.205,10.24.73.206 --spine
10.24.73.201,10.24.73.202  --username admin --password password --force
```

This example adds more leaf nodes to the fabric.

```
$ efa fabric configure --leaf 10.24.73.207,10.24.73.208
```

This example adds more spine nodes to the fabric.

```
$ efa fabric configure --spine 10.24.73.209,10.24.73.210
```

History

Release version	Command history
17s.1.02b	This command was introduced.

efa fabric deconfigure

Deletes switches from the IP Fabric.

Syntax

efa fabric deconfigure { **--device** *list-of-device-ips* } [**--persist**] [**--username** *string* **--password** *string*]

Command Default

No devices are deleted by default.

Parameters

--device *list-of-device-ips*
Specifies a comma-separated ist of device IP addresses.

--persist
Executes **copy running-config startup-config** on all leaf and spine devices.

username *string*
Specifies a username. The default credentials used by the application are "admin" and "password" . This option can be used to override the default credentials. The credentials apply for all devices that are provided as part of the command. In case the devices have different credentials, they can be added by means of a separate command.

password *string*
Specifies a password.

Modes

TPVM configuration mode

Usage Guidelines

This command is executed in TPVM configuration mode on one of the SLX switches within the fabric.

Examples

This example deletes devices the fabric and resolves conflicting configurations on the switch.

```
$ efa fabric deconfigure --device 10.24.73.207,10.24.73.208
```

History

Release version	Command history
17s.1.02b	This command was introduced.

efa fabric setting show

Displays the parameters of an IP Fabric.

Syntax

efa fabric setting show [**--advanced**]

Parameters

--advanced

Displays advanced fabric parameters.

Modes

TPVM configuration mode

Command Output

This command is executed in TPVM configuration mode on one of the SLX switches within the fabric.

Examples

This is example output for basic parameters.

```
$ efa fabric setting show
+-----------------------------------------------+-------------------------+
|           NAME                                |      VALUE              |
+-----------------------------------------------+-------------------------+
| Fabric Name                                   | default                 |
| Link IP Range                                 | 10.10.10.0/23           |
| Loopback IP Range                             | 172.32.254.0/24         |
| Loopback Port Number                          | 1                       |
| VTEP Loopback Port Number                     | 2                       |
| Spine ASN Block                               | 64512                   |
| LEAF ASN Block                                | 65000-65534             |
| P2P IP Type                                   | numbered                |
+-----------------------------------------------+-------------------------+
```

This is example output for advanced parameters.

```
$ efa fabric setting show --advanced
+--------------------------------+-----------------+
|              NAME              |      VALUE      |
+--------------------------------+-----------------+
| Fabric Name                    | default         |
| Link IP Range                  | 10.10.10.0/23   |
| Loopback IP Range              | 172.32.254.0/24 |
| Loopback Port Number           | 1               |
| VTEP Loopback Port Number      | 2               |
| Spine ASN Block                | 64512           |
| Leaf ASN Block                 | 65000-65534     |
| P2P IP Type                    | numbered        |
| Any cast MAC                   | 0201.0101.0101  |
| IPV6 Any cast MAC              | 0201.0101.0102  |
| ARP Aging Timeout              | 300             |
| MAC Aging Timeout              | 1800            |
| MAC Aging Conversational       | 300             |
| Timeout                        |                 |
| MAC Move Limit                 | 20              |
| Duplicate MAC Timer            | 5               |
| Duplicate MAC Timer MAX Count  | 3               |
| Configure Overlay Gateway      | Yes             |
| BFD Tx                         | 300             |
| BFD Rx                         | 300             |
| BFD Multiplier                 | 3               |
| BGP MultiHop                   | 2               |
| MaxPaths                       | 8               |
| AllowAsIn                      | 1               |
| MTU                            | 9216            |
| IPMTU                          | 9100            |
| Leaf PeerGroup                 | spine-group     |
| Spine PeerGroup                | leaf-group      |
| MCT Link IP Range              | 20.20.20.0/24   |
| MCT PortChannel                | 1024            |
| Control Vlan                   | 4090            |
| Control VE                     | 4090            |
| VNI Auto Map                   | Yes             |
+--------------------------------+-----------------+
```

History

Release version	Command history
17s.1.02b	This command was introduced.

efa fabric setting update

Allows the user to modify the settings of an IP Fabric.

Syntax

efa fabric setting update {**--allow-as-in** *string* | **--anycast-mac-address** *string* | **--arp-aging-timeout** *string* | **--bfd-multiplier** *string* | **--bfd-rx** *string* | **--bfd-tx** *string* | **--bgp-multihop** *string* | **--configure-overlay-gateway** *string* } | **control-ve** *string*| **control-vlan** *string* |**--duplicate-mac-timer** *string* | **--duplicate-mac-timer-max-count** *string* | **--help** | **--ip-mtu** *string* | **--ipv6-anycast-mac-address** *value* | **--leaf-asn-block** *string* | **--leaf-peer-group** *string* | **--loopback-ip-range** *string* | --loopback-port-number *string* | **--mac-aging-conversation-timeout** *string* | **--mac-aging-timeout** *string* | **--mac-move-limit** *string* | **--max-paths** *string* | **--mct-port-channel** *string* | **--mctlink-ip-range** *string* | **--mtu** *string* | **--p2p-ip-type** *string* | **--p2p-link-range** *string* | **--spine-asn-block** *string* | **--spine-peer-group** *string* | **--vni-auto-map** *string* | **--vtep-loopback-port-number** *string* }

Command Default

No command defaults apply.

Parameters

--allow-as-in *string*
Disables the AS_PATH check of the routes learned from the AS. Range is from 1 through 10.

--anycast-mac-address *string*
Specifies an IPv4 anycast MAC address in HHHH.HHHH.HHHH format.

--arp-aging-timeout *string*
Specifies how long an ARP entry stays in the cache. Range is from 60 through 100000 seconds.

--bfd-multiplier *string*
Specifies a multiplier for BFD detection time. Range is from 3 through 50.

--bfd-rx *string*
Specifies the BFD desired minimum receive interval in milliseconds. Range is from 50 through 30000.

--bfd-tx *string*
Specifies the BFD desired minimum transmit interval in milliseconds. Range is from 50 through 30000.

--bgp-multihop *string*
Specifies EBGP neighbors to allow when they are not on directly connected networks. Range is from 1 through 255.

--**configure-overlay-gateway** *string*
Specifies whether an overlay gateway is enabled or not. Options are "yes" or "no".

--control-ve *string*
Specifies a VLAN. Range is from 1 through 4090.

--control-vlan *string*
Specifies a VLAN. Range is from 1 through 4090.

--duplicate-mac-timer *string*
Specifies duplicate MAC timer.

--duplicate-mac-timer-max-count *string*
Specifies duplicate MAC timer max count.

--help
Specifies help for options for this command.

--ip-mtu *string*
Specifies an IPv4 or IPv6 MTU size in bytes for an SLX device. Range is from 1300 through 9194.

--ipv6-anycast-mac-address *string*
Specifies an IPv6 anycast MAC address in HHHH.HHHH.HHHH format.

--leaf-asn-block *string*
Specifies a leaf ASN range, in comma-separated format.

--leaf-peer-group *string*
Specifies the name of a leaf peer group. Range is from 1 through 63 ASCII characters.

--loopback-ip-range *string*
Specifies a range of loopback IP addresses.

--loopback-port-number *string*
Specifies a loopback port number. Range is from 1 through 255.

--mac-aging-conversation-timeout *string*
Specifies the MAC conversational aging time in seconds. Range is from 60 through 100000. The default is 0.

--mac-aging-timeout *string*
Specifies the MAC aging time in seconds. Range is from 60 through 100000. The default is 0.

--mac-move-limit *string*
Specifies the MAC move detection limit. Range is from 5 through 500.

--max-paths *string*
Specifies the maximum number of paths for packet forwarding. Range is from 1 through 64.

--mct-port-channel *string*
Specifies a port-channel interface number. Range is from 1 through 1024.

--mctlink-ip-range *string*
Specifies a range of IP addresses.

--mtu *string*
Specifies the MTU size in bytes. Range is from 1548 through 9216.

--p2p-ip-type *string*
Specifies the IP type as numbered or unnumbered.

--p2p-link-range *string*
Specifies a range of IP addresses.

--spine-asn-block *string*
Specifies a spine ASN range, as a single AS or as comma-separated AS values.

--spine-peer-group *string*
Specifies the name of a spine peer group. Range is from 1 through 63 ASCII characters.

--vni-auto-map *string*
Specifies whether VNI automapping used or not. Options are "yes" or "no".

--vtep-loopback-port-number *string*
Specifies a VTEP loopback port number. Range is from 1 through 255.

Modes

TPVM configuration mode

Usage Guidelines

This command is executed in TPVM configuration mode on one of the SLX switches within the fabric.

Examples

This example specifies a range of leaf switches.

```
$ efa fabric setting update --leaf_asn_block 65000-65534
```

This example specifies a range of IP addresses to be used for fabric links.

```
$ efa fabric setting update --link_ip_range 10.10.10.0/23
```

History

Release version	Command history
17s.1.02b	This command was introduced.

efa supportsave

Creates a supportSave .zip package that contains the EFA database and execution logs.

Syntax

efa supportsave

Command Default

supportSave is not configured by default.

Modes

TPVM configuration mode

Usage Guidelines

This command is executed in TPVM configuration mode on one of the SLX switches within the fabric.

The database and execution logs are available in `/var/efa/efa_SS.zip`.

Examples

This example creates a supportSave .zip package that contains the EFA database and execution logs on the switch.

```
$ efa supportsave
```

History

Release version	Command history
17s.1.02b	This command was introduced.