Cisco Nexus 9000 Series NX-OS Label Switching Configuration Guide, Release 9.3(x)

First Published: 2019-07-20

Last Modified: 2020-07-30

Americas Headquarters
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
USA
http://www.cisco.com
Tel: 408 526-4000
800 553-NETS (6387)
Fax: 408 527-0883

CONTENTS

Preface

This preface includes the following sections:

Audience

This publication is for network administrators who install, configure, and maintain Cisco Nexus switches.

Document Conventions

Command descriptions use the following conventions:

Convention	Description
bold	Bold text indicates the commands and keywords that you enter literally as shown.
Italic	Italic text indicates arguments for which you supply the values.
[x]	Square brackets enclose an optional element (keyword or argument).
[x | y]	Square brackets enclosing keywords or arguments that are separated by a vertical bar indicate an optional choice.
{x | y}	Braces enclosing keywords or arguments that are separated by a vertical bar indicate a required choice.
[x {y | z}]	Nested set of square brackets or braces indicate optional or required choices within optional or required elements. Braces and a vertical bar within square brackets indicate a required choice within an optional element.

Convention	Description
`variable`	Indicates a variable for which you supply values, in context where italics cannot be used.
string	A nonquoted set of characters. Do not use quotation marks around the string or the string includes the quotation marks.

Examples use the following conventions:

Convention	Description
`screen font`	Terminal sessions and information the switch displays are in screen font.
`boldface screen font`	Information that you must enter is in boldface screen font.
italic screen font	Arguments for which you supply values are in italic screen font.
< >	Nonprinting characters, such as passwords, are in angle brackets.
[ ]	Default responses to system prompts are in square brackets.
!, #	An exclamation point (!) or a pound sign (#) at the beginning of a line of code indicates a comment line.

Related Documentation for Cisco Nexus 9000 Series Switches

The entire Cisco Nexus 9000 Series switch documentation set is available at the following URL:

http://www.cisco.com/en/US/products/ps13386/tsd_products_support_series_home.html

Documentation Feedback

To provide technical feedback on this document, or to report an error or omission, please send your comments to nexus9k-docfeedback@cisco.com. We appreciate your feedback.

Communications, Services, and Additional Information

- To receive timely, relevant information from Cisco, sign up at Cisco Profile Manager.
- To get the business impact you're looking for with the technologies that matter, visit Cisco Services.
- To submit a service request, visit Cisco Support.
- To discover and browse secure, validated enterprise-class apps, products, solutions and services, visit Cisco Marketplace.
- To obtain general networking, training, and certification titles, visit Cisco Press.
- To find warranty information for a specific product or product family, access Cisco Warranty Finder.

Cisco Bug Search Tool

Cisco Bug Search Tool (BST) is a web-based tool that acts as a gateway to the Cisco bug tracking system that maintains a comprehensive list of defects and vulnerabilities in Cisco products and software. BST provides you with detailed defect information about your products and software.

CHAPTER 1

New and Changed Information

This chapter provides release-specific information for each new and changed feature in the *Cisco Nexus 9000 Series NX-OS Label Switching Configuration Guide, Release 9.3(x)*.

New and Changed Information

This table summarizes the new and changed features for the *Cisco Nexus 9000 Series NX-OS Label Switching Configuration Guide, Release 9.3(x)*.

Table 1: New and Changed Features

Feature	Description	Changed in Release	Where Documented
MPLS QoS	Added support on Cisco Nexus 9300-GX platform switches.	9.3(5)	Guidelines and Limitations for MPLS QoS, on page 99
MPLS strip	Added support on Cisco Nexus 9300-GX platform switches.	9.3(5)	Guidelines and Limitations for Segment Routing, on page 113
Proportional Multipath for VNF for Segment Routing	Added the ability to advertise the VNF of a service network in the EVPN address family. This feature is supported on Cisco Nexus 9300-EX, 9300-FX/FX2, 9300-GX, and 9500 switches with 9700-EX, and 9700-FX line cards.	9.3(5)	• About Proportional Multipath for VNF for Segment Routing, on page 166 • Enabling Proportional Multipath for VNF for Segment Routing, on page 166
Segment routing	Added support for Layer 2 EVPN on Cisco Nexus 9300-GX platform switches.	9.3(5)	Guidelines and Limitations for Layer 2 EVPN over Segment Routing MPLS, on page 154

Feature	Description	Changed in Release	Where Documented
SRv6 Static Per-Prefix TE	Added the ability to advertise multiple prefixes as a part of the same VRF through the route map. This feature is supported on Cisco Nexus 9300-GX platform switches.	9.3(5)	• About SRv6 Static Per-Prefix TE, on page 160 • #unique_16
MVPNs	Introduced this feature.	9.3(3)	Configuring MVPNs, on page 185
Segment routing MPLS	Added support on Cisco Nexus 9364C-GX, Cisco Nexus 9316D-GX, and Cisco Nexus 93600CD-GX switches.	9.3(3)	See the relevant chapters in this guide.
MPLS Adjacency Statistics	Added support for the **hardware profile mpls adjacency-stats bytes** command.	9.3(1)	Configuring Segment Routing Adjacency Statistics, on page 14 Verifying the Static MPLS Configuration, on page 16
MPLS Queuing	Added support on Cisco Nexus 9300-FX2 platform switches.	9.3(1)	Configuring MPLS QoS, on page 97
MPLS QoS	Added support for the default QoS service template to support the MPLS handoff.	9.3(1)	Configuring MPLS QoS, on page 97
NetFlow	Added support for NetFlow Collector over segment routing on Cisco Nexus 9300-EX, 9300-FX, 9300-FX2, 9500-EX, and the 9500-FX platform switches.	9.3(1)	NetFlow for MPLS, on page 112
MPLS ECMP Load Sharing	Added support for load sharing for port channel and ECMP based on labels.	9.3(1)	Configuring MPLS ECMP Load Sharing, on page 93 Verifying MPLS ECMP Load Sharing, on page 94

Feature	Description	Changed in Release	Where Documented
Segment Routing and GRE	Added support to enable both, MPLS and GRE on the same device.	9.3(1)	GRE Tunnels, on page 178 Segment Routing MPLS and GRE, on page 179 Guidelines and Limitations for Segment Routing MPLS and GRE, on page 179 Configuring Segment Routing MPLS and GRE, on page 180
Segment Routing OAM	Added support for routing protocols and modified the CLIs for Nil-FEC type.	9.3(1)	Configuring MPLS Segment Routing OAM, on page 199
Segment Routing	Changed the command for configuring segment routing from **segment-routing mpls** to **segment-routing**.	9.3(1)	Configuring Segment Routing, on page 111
Segment Routing	Added support for affinity and disjoint path constraints.	9.3(1)	Configuring Affinity Constraints, on page 137 Configuring Disjoint Paths, on page 139
Segment Routing	Added support for explicit paths.	9.3(1)	SR-TE Policy Paths, on page 134 Configuring SR-TE, on page 136
Segment Routing	Added support for explicit SR-TE policy.	9.3(1)	SR-TE Policies, on page 134 Configuring SR-TE, on page 136
Segment Routing	Added support for SR-TE ODN for a IPv6 overlay.	9.3(1)	Guidelines and Limitations for SR-TE, on page 135
Segment Routing	Added support for adjacency SIDs on OSPFv2.	9.3(1)	Configuring Segment Routing, on page 111
Segment Routing	Added support for Layer 2 EVPN on Cisco Nexus 9300 FX2 platform switches.	9.3(1)	Configuring Layer 2 EVPN over Segment Routing MPLS, on page 154
sFlow	Added support for a sFlow collector over segment routing for both default and non-default VRF.	9.3(1)	sFlow Collector, on page 112

Feature	Description	Changed in Release	Where Documented
vPC multihoming	Added support for vPC-based multihoming over MPLS on Cisco Nexus 9300-FX2 Series switches.	9.3(1)	About Multihoming, on page 168

CHAPTER 2

Overview

Licensing Requirements

For a complete explanation of Cisco NX-OS licensing recommendations and how to obtain and apply licenses, see the *Cisco NX-OS Licensing Guide* and the *Cisco NX-OS Licensing Options Guide*.

Supported Platforms

Starting with Cisco NX-OS release 7.0(3)I7(1), use the Nexus Switch Platform Support Matrix to know from which Cisco NX-OS releases various Cisco Nexus 9000 and 3000 switches support a selected feature.

CHAPTER 3

Configuring Static MPLS

This chapter contains information on how to configure static multiprotocol label switching (MPLS).

Licensing Requirements

For a complete explanation of Cisco NX-OS licensing recommendations and how to obtain and apply licenses, see the *Cisco NX-OS Licensing Guide* and the *Cisco NX-OS Licensing Options Guide*.

About Static MPLS

Generally, label switching routers (LSRs) use a label distribution protocol to dynamically learn the labels that they should use to label-switch packets. Examples of such protocols include:

- Label Distribution Protocol (LDP), the Internet Engineering Task Force (IETF) standard that is used to bind labels to network addresses
- Resource Reservation Protocol (RSVP), which is used to distribute labels for traffic engineering (TE)
- Border Gateway Protocol (BGP), which is used to distribute labels for MPLS virtual private networks (VPNs)

To use a learned label to label-switch packets, an LSR installs the label into its Label Forwarding Information Base (LFIB).

The static MPLS feature enables you to statically configure the following:

- The binding between a label and an IPv4 or IPv6 prefix
- The action corresponding to the binding between a label and an IPv4 or IPv6 prefix (label swap or pop)
- The contents of an LFIB cross-connect entry

Label Swap and Pop

As a labeled packet traverses the MPLS domain, the outermost label of the label stack is examined at each hop. Depending on the contents of the label, a swap or pop (dispose) operation is performed on the label stack. Forwarding decisions are made by performing an MPLS table lookup for the label carried in the packet header. The packet header does not need to be reevaluated during packet transit through the network. Because the label has a fixed length and is unstructured, the MPLS forwarding table lookup process is both straightforward and fast.

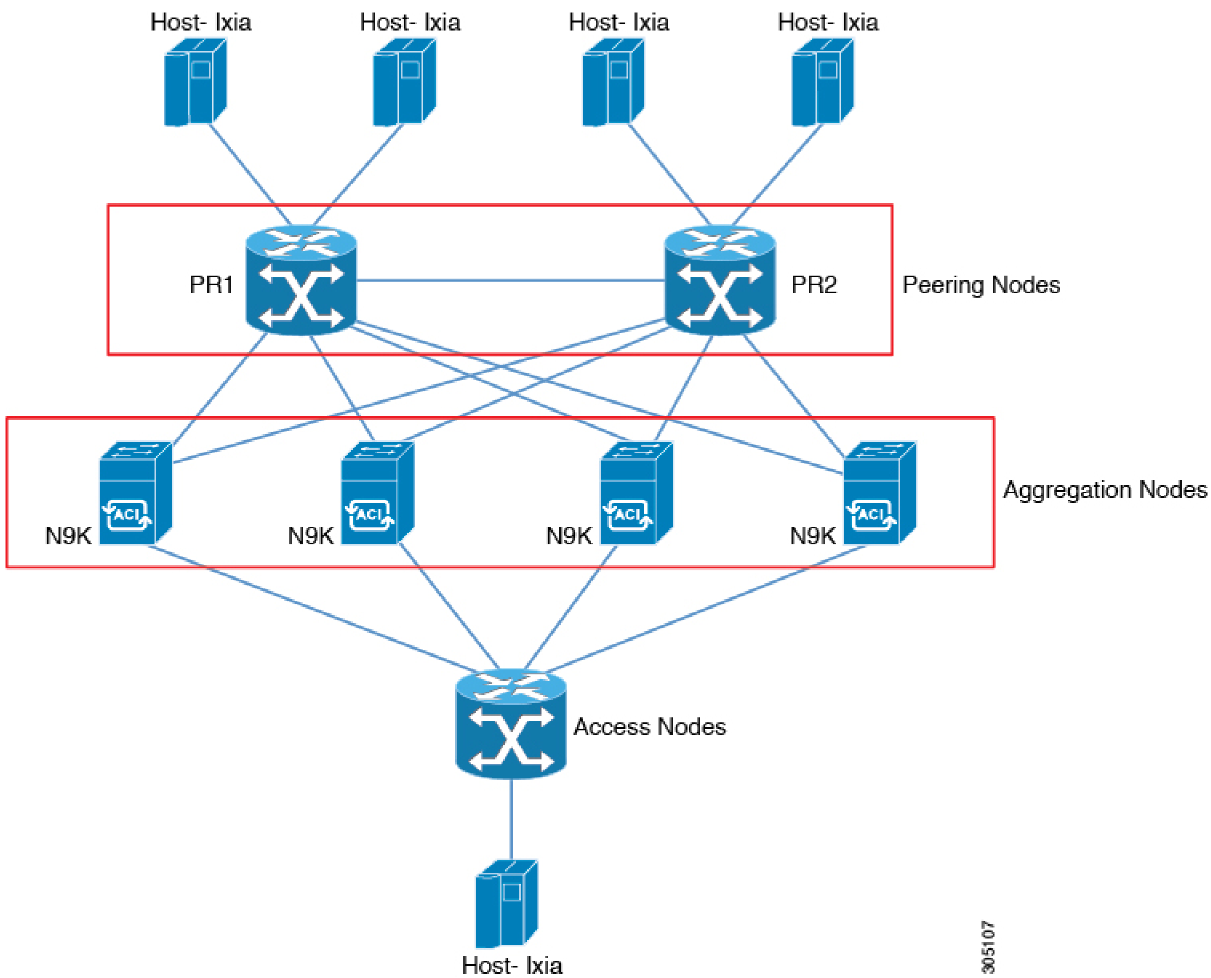
In a swap operation, the label is swapped with a new label, and the packet is forwarded to the next hop that is determined by the incoming label.

In a pop operation, the label is removed from the packet, which may reveal an inner label below. If the popped label was the last label on the label stack, the packet exits the MPLS domain. Typically, this process occurs at the egress LSR. A failure of the primary link in the aggregator reroutes the MPLS traffic to the backup link and results in a swap operation.

Static MPLS Topology

This diagram illustrates the static MPLS source routing topology. The access nodes perform the swap operation, and the aggregation nodes perform the pop operation for the primary path and the swap operation for the backup path.

Figure 1: Static MPLS Topology



Benefits of Static MPLS

- Static bindings between labels and IPv4 or IPv6 prefixes can be configured to support MPLS hop-by-hop forwarding through neighbor routers that do not implement LDP label distribution.
- Static cross-connects can be configured to support MPLS label switched path (LSP) midpoints when neighbor routers do not implement either LDP or RSVP label distribution but do implement an MPLS forwarding path.

High Availability for Static MPLS

Cisco Nexus 9500 Series switches support stateful switchovers (SSOs) for static MPLS. After an SSO, static MPLS returns to the state it was in previously.

Static MPLS supports zero traffic loss during SSO. MPLS static restarts are not supported.

> **Note** The Cisco Nexus 9300 Series switches do not support SSO.

Prerequisites for Static MPLS

Static MPLS has the following prerequisites:

- For Cisco Nexus 9300 and 9500 Series switches and the Cisco Nexus 3164Q, 31128PQ, 3232C, and 3264Q switches, you must configure the ACL TCAM region size for MPLS, save the configuration, and reload the switch. (For more information, see the "Using Templates to Configure ACL TCAM Region Sizes" and "Configuring ACL TCAM Region Sizes" sections in the Cisco Nexus 9000 Series NX-OS Security Configuration Guide.) The Cisco Nexus 9200 Series switches do not require TCAM carving for static MPLS.

> **Note** By default the mpls region size is zero. You need to configure this region to 256 in order to support static MPLS.

Guidelines and Limitations for Static MPLS

Static MPLS has the following guidelines and limitations:

- Static MPLS is supported on Cisco Nexus 3100, 3200, 9200, 9300, 9300-EX, FX, FX2 and 9500 switches with the 9400, 9500, 9600, and 9700-EX line cards.
- Beginning with Cisco NX-OS Release 9.3(3), static MPLS is supported on Cisco Nexus 9364C-GX, Cisco Nexus 9316D-GX, and Cisco Nexus 93600CD-GX switches.
- Static MPLS, MPLS segment routing, and MPLS stripping cannot be enabled at the same time.
- Equal-cost multipath (ECMP) is not supported with Label pop.
- Label pop and swap operations are supported, but label push operations are not.
- MPLS packets are forwarded as long as the ingress label matches the configured label and the configured FEC (prefix) is in the routing table.
- The device generally performs as a label switching router (LSR). If you install the explicit null label as the out-label in the label FIB (LFIB) by an LSR before the packet is passed to an adjacent LER, the device performs as a label edge router (LER) for penultimate hop popping. Meaning that a label switching router (LSR) functions with one or more labels.

> **Note** If you intentionally use implicit-null CLI on LSR, the output packet going to the LER, it contains an explicit-null and the inner label.

- Static MPLS supports up to 128 labels.

- The backup path is supported only for a single adjacency and not for ECMP.
- Cisco Nexus 9300 Series switches support backup path Fast Reroute (FRR) subsecond convergence whereas Cisco Nexus 9500 Series switches support a limited backup path FRR convergence.
- The output for most of the MPLS commands can be generated in XML or JSON. See Verifying the Static MPLS Configuration, on page 16 for an example.
- VRFs, vPCs, FEX, and VXLAN are not supported with static MPLS.
- When sub-interfaces are used to connect to the remote vpnv4 neighbors, the parent interface needs to enable "mpls ip forwarding" command.
- Command "mpls ip forwarding" cannot be configured under a sub-interface.
- Subinterfaces are not supported for static MPLS.
- The Forwarding Equivalence Class (FEC) must match routes in the routing table.
- Static MPLS is enabled and cannot be disabled on the X9536PQ, X9564PX, and X9564TX line cards and the M12PQ generic expansion module (GEM).
- When you configure Fast Reroute (backup), you can specify only the connected next hop (and not the recursive next hop) as the next-hop prefix in the backup configuration.
- When multiple FECs are sharing the backup (the same next-hop and interface), any change to the backup configuration requires a reconfiguration of all the other FECs that are sharing the backup configuration.
- When the backup path is active, the **show mpls switching labels** command will not show the out label/out interface/next hop and related statistics. You can use the **show forwarding mpls label** *label* **stats platform** command to check the statistics.
- If traffic ingresses or egresses on a non-default unit (where the default unit is unit0), the corresponding ULIB statistics will not be displayed in the output of the **show mpls switching labels** *low-label-value* [*high-label-value*] **detail** command. You can use the **show forwarding mpls label** *label* **stats platform** command to check the statistics.
- If the backup and primary paths are pointing to the same interface, the backup action swap takes precedence.
- Physical (Ethernet) and port channels are supported only for backup.
- The following guidelines and limitations apply to Cisco Nexus 9200 Series switches:
    - ECMP hashing is supported only on inner fields.
    - MTU checks are not supported for packets with an MPLS header.

Configuring Static MPLS

Enabling Static MPLS

You must install and enable the MPLS feature set and then enable the MPLS static feature before you can configure MPLS static labels.

Procedure

	Command or Action	Purpose

Step 1	**configure terminal**
	Example:
	`switch# configure terminal`
	`switch(config)#`
		Enters global configuration mode.

Step 2	[**no**] **install feature-set mpls** **Example:** `switch(config)# install feature-set mpls`	Installs the MPLS feature set. The **no** form of this command uninstalls the MPLS feature set.
Step 3	[**no**] **feature-set mpls** **Example:** `switch(config)# feature-set mpls`	Enables the MPLS feature set. The **no** form of this command disables the MPLS feature set.
Step 4	[**no**] **feature mpls static** **Example:** `switch(config)# feature mpls static`	Enables the static MPLS feature. The **no** form of this command disables the static MPLS feature.

Step 5	(Optional) **show feature-set**
	Example:
	`switch(config)# show feature-set`
	`Feature Set Name      ID        State`
	
	`--------------------  --------  --------`
	`mpls                  4         enabled`
		Displays the status of the MPLS feature set.

Step 6	(Optional) **show feature | inc mpls_static**
	Example:
	`switch(config)# show feature | inc`
	`mpls_static`
	`mpls_static           1         enabled`
		Displays the status of static MPLS.

Reserving Labels for Static Assignment

You can reserve the labels that are to be statically assigned so that they are not dynamically assigned.

Before you begin

Ensure that the static MPLS feature is enabled.

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:**	Enters global configuration mode.

	Command or Action	Purpose
	`switch# configure terminal` `switch(config)#`	
Step 2	[**no**] **mpls label range** *min-value max-value* [**static** *min-static-value max-static-value*] **Example:** `switch(config)# mpls label range 17 99 static 100 10000`	Reserves a range of labels for static label assignment. The range for the minimum and maximum values is from 16 to 471804.
Step 3	(Optional) **show mpls label range** **Example:** `switch(config)# show mpls label range`	Displays the label range that is configured for static MPLS.
Step 4	(Optional) **copy running-config startup-config** **Example:** `switch(config)# copy running-config startup-config`	Copies the running configuration to the startup configuration.

Configuring Static Label and Prefix Binding Using the Swap and Pop Operations

In a top-of-rack configuration, the outer label is swapped to the specified new label. The packet is forwarded to the next-hop address, which is auto-resolved by the new label.

In an aggregator configuration, the outer label is popped, and the packet with the remaining label is forwarded to the next-hop address. Pop operations are performed in the primary path, and swap operations are performed in the backup path.

Before you begin

Ensure that the static MPLS feature is enabled.

Procedure

	Command or Action	Purpose

Step 1	**configure terminal**
	Example:
	`switch# configure terminal`
	`switch(config)#`
		Enters global configuration mode.

Step 2	**interface** *type slot/port*
	Example:
	`switch(config)# interface ethernet 2/2`
	`switch(config-if)#`
		Enters the interface configuration mode for the specified interface.

Step 3	[**no**] **mpls ip forwarding** **Example:** `switch(config-if)# mpls ip forwarding`	Enables MPLS on the specified interface. The **no** form of this command disables MPLS on the specified interface.

	Command or Action	Purpose

Step 4	**mpls static configuration**
	Example:
	`switch(config-if)# mpls static configuration`
	`switch(config-mpls-static)#`
		Enters MPLS static global configuration mode.

Step 5	**address-family** {**ipv4** | **ipv6**} **unicast**
	Example:
	`switch(config-mpls-static)# address-family ipv4 unicast`
	`switch(config-mpls-static-af)#`
		Enters global address family configuration mode for the specified IPv4 or IPv6 address family.

Step 6	**local-label** *local-label-value* **prefix** *destination-prefix destination-prefix-mask*
	Example:
	`switch(config-mpls-static-af)# local-label 2000 prefix 1.255.200.0 255.255.255.25`
	`switch(config-mpls-static-af-lbl)#`
		Specifies static binding of incoming labels to IPv4 or IPv6 prefixes. The *local-label-value* is the range of the static MPLS label defined in the **mpls label range** command.

Step 7	**next-hop** {**auto-resolve** | *destination-ip-next-hop* **out-label implicit-null** | **backup** *local-egress-interface destination-ip-next-hop* **out-label** *output-label-value*} **Example:** `switch(config-mpls-static-af-lbl)# next-hop auto-resolve`	Specifies the next hop. These options are available: • **next-hop auto-resolve**—Use this option for label swap operations. • **next-hop** *destination-ip-next-hop* **out-label implicit-null**—Use this option for the primary path in label pop operations. • **next-hop backup** *local-egress-interface destination-ip-next-hop* **out-label** *output-label-value*—Use this option for the backup path in label pop operations.
Step 8	(Optional) **copy running-config startup-config** **Example:** `switch(config-mpls-static-af-lbl)# copy running-config startup-config`	Copies the running configuration to the startup configuration.

Configuring Segment Routing Adjacency Statistics

By default, the statistics collection mode accumulates the number of packets that egress out of a given adjacency. Beginning Cisco NX-OS Release 9.3(1), you can configure the statistics collection mode to accumulate the number of bytes for an adjacency.

This mode is available when you enable the MPLS segment routing feature, however you must configure the collection mode to accumulate bytes.

Procedure

	Command or Action	Purpose

Step 1	**configure terminal**
	Example:
	`switch# configure terminal`
	`switch(config)#`
		Enters global configuration mode.

Step 2	[**no**] **install feature-set mpls** **Example:** `switch(config)# install feature-set mpls`	Installs the MPLS feature set. The **no** form of this command uninstalls the MPLS feature set.
Step 3	[**no**] **feature-set mpls** **Example:** `switch(config)# feature-set mpls`	Enables the MPLS feature set. The **no** form of this command disables the MPLS feature set.
Step 4	[**no**] **feature mpls segment-routing** **Example:** `switch(config)# feature mpls segment-routing`	Enables the MPLS segment routing feature. The **no** form of this command disables the MPLS segment routing feature.
Step 5	[**no**] **hardware profile mpls adjacency-stats bytes** **Example:** `switch(config)# hardware profile mpls adjacency-stats bytes`	Configures the statistics collection mode for the output statistics to accumulate the count of bytes for a given adjacency. The **no** form of this command resets the collection mode to accumulate the packet count.

Step 6	(Optional) **show running-config | grep adjacency stats**
	Example:
	`witch(config)# show running-config | grep adjacency-stats`
	`hardware profile mpls adjacency-stats bytes`
	`switch(config)#`
		Displays the knob configuration.

Step 7	(Optional) **show feature-set**
	Example:
	`switch(config)# show feature-set`
	`Feature Set Name      ID        State`
	`--------------------  --------  --------`
	`mpls                  4         enabled`
		Displays the status of the MPLS feature set.

Step 8	(Optional) **show feature | grep segment-routing**
	Example:
	`switch(config)# show feature | grep segment-routing`
	`segment-routing        1        enabled`
		Displays the status of MPLS segment routing.

	Command or Action	Purpose

Step 9	**show forwarding mpls** [**label** *label*] **stats**
	Example:
	`switch(config)# show forwarding mpls label 22 stats`
	
	`slot 1`
	`=======`
	
	`------+---------+--------------+-------------+-----------+------`
	`Local |Prefix |FEC |Next-Hop |Interface |Out`
	`Label |Table Id |(Prefix/Tunnel id) | | |Label`
	`------+---------+--------------+-------------+-----------+------`
	`22 |0x1 |182.1.1.7/32 |30.1.8.1 |Po11 |0 SWAP`
	
	`Input Pkts : 488482 Input Bytes : 250102784`
	`SWAP Output Pkts: 0 SWAP Output Bytes: 84215808`
	`TUNNEL Output Pkts: 0 TUNNEL Output Bytes: 0`
	`switch(config)#`
		Displays the adjacency statistics.

Verifying the Static MPLS Configuration

To display the static MPLS configuration, perform one of the following tasks:

Command	Purpose
show feature | inc mpls_static	Displays the status of static MPLS.
show feature-set	Displays the status of the MPLS feature set.
show ip route	Displays routes from the unicast Routing Information Base (RIB).
show mpls label range	Displays the label range that is configured for static MPLS.
show mpls static binding {**all** | **ipv4** | **ipv6**}	Displays the configured static prefix or label bindings.
show mpls switching [**detail**]	Displays MPLS switching information.
show mpls switching label [**detail**]	Displays the MPLS switching label information.
show forwarding mpls [**label** *label*] **stats**	Displays the adjacency statistics based on the label enabled.
show forwarding adjacency mpls stats	Displays the adjacency statistics

This example shows sample output for the **show mpls static binding all** command:

```
1.255.200.0/32: (vrf: default) Incoming label: 2000
  Outgoing labels:
      1.21.1.1 implicit-null
      backup 1.24.1.1 2001

2000:1:255:201::1/128: (vrf: default) Incoming label: 3000
  Outgoing labels:
      2000:1111:2121:1111:1111:1111:1111:1 implicit-null
      backup 2000:1:24:1::1 3001
```

This example shows sample output for the **show mpls switching detail** command:

```
VRF default

IPv4 FEC
 In-Label                       : 2000
 Out-Label stack                : Pop Label
 FEC                            : 1.255.200.0/32
 Out interface                  : Po21
 Next hop                       : 1.21.1.1
 Input traffic statistics       : 0 packets, 0 bytes
 Output statistics per label    : 0 packets, 0 bytes
IPv6 FEC
 In-Label                       : 3000
 Out-Label stack                : Pop Label
 FEC                            : 2000:1:255:201::1/128
 Out interface                  : port-channel21
 Next hop                       : 2000:1111:2121:1111:1111:1111:1111:1
 Input traffic statistics       : 0 packets, 0 bytes
 Output statistics per label    : 0 packets, 0 bytes
```

This example shows normal, XML, and JSON sample output for the **show mpls switching** command when the switch is configured with a static IPv4 prefix:

```
switch# show run mpls static | sec 'ipv4 unicast'
address-family ipv4 unicast
local-label 100 prefix 192.168.0.1 255.255.255.255 next-hop auto-resolve out-label 200

switch# show mpls switching
Legend:
(P)=Protected, (F)=FRR active, (*)=more labels in stack.
IPV4:
In-Label   Out-Label  FEC name           Out-Interface      Next-Hop

VRF default
100        200        192.168.0.1/32     Eth1/23            1.12.23.2


switch# show mpls switching | xml
<?xml version="1.0" encoding="ISO-8859-1"?> <nf:rpc-reply
xmlns:nf="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns="http://w
ww.cisco.com/nxos:1.0:ulib">
 <nf:data>
  <show>
   <mpls>
    <switching>
     <__XML__OPT_Cmd_ulib_show_switching_cmd_labels>
      <__XML__OPT_Cmd_ulib_show_switching_cmd_detail>
       <__XML__OPT_Cmd_ulib_show_switching_cmd_vrf>
        <__XML__OPT_Cmd_ulib_show_switching_cmd___readonly__>
         <__readonly__>
          <TABLE_vrf>
```

```
              <ROW_vrf>
               <vrf_name>default</vrf_name>
               <TABLE_inlabel>
                <ROW_inlabel>
                 <in_label>100</in_label>
                 <out_label_stack>200</out_label_stack>
                 <ipv4_prefix>192.168.0.1/32</ipv4_prefix>
                 <out_interface>Eth1/23</out_interface>
                 <ipv4_next_hop>1.12.23.2</ipv4_next_hop>
                 <nhlfe_p2p_flag> </nhlfe_p2p_flag>
                </ROW_inlabel>
               </TABLE_inlabel>
              </ROW_vrf>
             </TABLE_vrf>
            </__readonly__>
           </__XML__OPT_Cmd_ulib_show_switching_cmd___readonly__>
          </__XML__OPT_Cmd_ulib_show_switching_cmd_vrf>
         </__XML__OPT_Cmd_ulib_show_switching_cmd_detail>
        </__XML__OPT_Cmd_ulib_show_switching_cmd_labels>
       </switching>
      </mpls>
     </show>
    </nf:data>
   </nf:rpc-reply>
   ]]>]]>
```


```
switch# show mpls switching | json
```


```
{"TABLE_vrf": {"ROW_vrf": {"vrf_name": "default", "TABLE_inlabel":
{"ROW_inlabel
": {"in_label": "100", "out_label_stack": "200", "ipv4_prefix":
"192.168.0.1/32"
, "out_interface": "Eth1/23", "ipv4_next_hop": "1.12.23.2",
"nhlfe_p2p_flag": nu
ll}}}}}
```


Displaying Static MPLS Statistics

To monitor static MPLS statistics, perform one of the following tasks:

Command	Purpose
show forwarding [**ipv6**] **adjacency mpls stats**	Displays MPLS IPv4 or IPv6 adjacency statistics.
show forwarding mpls drop-stats	Displays the MPLS forwarding packet drop statistics.
show forwarding mpls ecmp [**module** *slot* | **platform**]	Displays the MPLS forwarding statistics for equal-cost multipath (ECMP).
show forwarding mpls label *label* **stats** [**platform**]	Displays MPLS label forwarding statistics.
show mpls forwarding statistics [**interface** *type slot/port*]	Displays MPLS forwarding statistics.
show mpls switching labels *low-label-value* [*high-label-value*] [**detail**]	Displays the MPLS label switching statistics. The range for the label value is from 0 to 524286.

This example shows sample output for the **show forwarding adjacency mpls stats** command:

```
FEC             next-hop   interface  tx packets  tx bytes  Label info
--------------- ---------- ---------- ----------- --------- ----------
1.255.200.0/32  1.21.1.1   Po21       87388       10836236  POP 3
1.255.200.0/32  1.24.1.1   Po24       0           0         SWAP 2001
switch(config)#
switch(config)# show forwarding mpls drop-stats

Dropped packets : 73454
Dropped bytes : 9399304
```

This example shows sample output for the **show forwarding ipv6 adjacency mpls stats** command:

```
FEC                    next-hop        interface  tx packets  tx bytes  Label info
---------------------- --------------  ---------- ----------- --------- -----------
2000:1:255:201::1/128  2000:1.21.1.1   Po21       46604       5778896   POP 3
2000:1:255:201::1/128  2000:1:24:1::1  Po24       0           0         SWAP 3001
```

This example shows sample output for the **show forwarding mpls label 2000 stats** command:

```
--------+-----------+-------------------+----------------+-------------+-------
Local   |Prefix     |FEC                |Next-Hop        |Interface    |Out
Label   |Table Id   |(Prefix/Tunnel id) |                |             |Label
--------+-----------+-------------------+----------------+-------------+-------
2000    |0x1        |1.255.200.0/32     |1.21.1.1        |Po21         |Pop Label
 HH: 100008, Refcount: 1
Input Pkts : 77129               Input Bytes : 9872512
Output Pkts: 77223               Output Bytes: 9575652
```

This example shows sample output for the **show mpls forwarding statistics** command:

```
MPLS software forwarding stats summary:
    Packets/Bytes sent           : 0/0
    Packets/Bytes received       : 0/0
    Packets/Bytes forwarded      : 0/0
    Packets/Bytes originated     : 0/0
    Packets/Bytes consumed       : 0/0
    Packets/Bytes input dropped  : 0/0
    Packets/Bytes output dropped : 0/0
```

Clearing Static MPLS Statistics

To clear the static MPLS statistics, perform these tasks:

Command	Purpose
clear forwarding [**ipv6**] **adjacency mpls stats**	Clears the MPLS IPv4 or IPv6 adjacency statistics.
clear forwarding mpls drop-stats	Clears the MPLS forwarding packet drop statistics.
clear forwarding mpls stats	Clears the ingress MPLS forwarding statistics.
clear mpls forwarding statistics	Clears the MPLS forwarding statistics.

Command	Purpose
clear mpls switching label statistics [**interface** *type slot/port*]	Clears the MPLS switching label statistics.

Configuration Examples for Static MPLS

This example shows how to reserve labels for static assignment:

```
switch# configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
switch(config)# mpls label range 17 99 static 100 10000
switch(config)# show mpls label range
Downstream Generic label region: Min/Max label: 17/99
Range for static labels: Min/Max Number: 100/10000
```

This example shows how to configure MPLS static label and IPv4 prefix binding in a top-of-rack configuration (swap configuration):

```
switch# configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
switch(config)# interface ethernet 1/1
switch(config-if)# mpls ip forwarding
switch(config-if)# mpls static configuration
switch(config-mpls-static)# address-family ipv4 unicast
switch(config-mpls-static-af)# local-label 2000 prefix 1.255.200.0/32
switch(config-mpls-static-af-lbl)# next-hop auto-resolve out-label 2000
```

This example shows how to configure MPLS static label and IPv6 prefix binding in a top-of-rack configuration (swap configuration):

```
switch# configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
switch(config)# interface ethernet 1/1
switch(config-if)# mpls ip forwarding
switch(config-if)# mpls static configuration
switch(config-mpls-static)# address-family ipv6 unicast
switch(config-mpls-static-af)# local-label 3001 prefix 2000:1:255:201::1/128
switch(config-mpls-static-af-lbl)# next-hop auto-resolve out-label 3001
```

This example shows how to configure MPLS static label and IPv4 prefix binding in an aggregator configuration (pop configuration):

```
switch# configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
switch(config)# interface ethernet 1/1
switch(config-if)# mpls ip forwarding
switch(config-if)# mpls static configuration
switch(config-mpls-static)# address-family ipv4 unicast
switch(config-mpls-static-af)# local-label 2000 prefix 1.255.200.0/32
switch(config-mpls-static-af-lbl)# next-hop 1.31.1.1 out-label implicit-null
switch(config-mpls-static-af-lbl)# next-hop backup Po34 1.34.1.1 out-label 2000
```

This example shows how to configure MPLS static label and IPv6 prefix binding in an aggregator configuration (pop configuration):

```
switch# configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
switch(config)# interface ethernet 1/1
switch(config-if)# mpls ip forwarding
switch(config-if)# mpls static configuration
switch(config-mpls-static)# address-family ipv6 unicast
switch(config-mpls-static-af)# local-label 3001 prefix 2000:1:255:201::1/128
switch(config-mpls-static-af-lbl)# next-hop 2000:1:31:1::1 out-label implicit-null
switch(config-mpls-static-af-lbl)# next-hop backup Po34 2000:1:34:1::1 out-label 3001
```

Additional References

Related Documents

Related Topic	Document Title
MPLS TCAM regions	See the *Using Templates to Configure ACL TCAM Region Sizes* section in the Cisco Nexus 9000 Series NX-OS Security Configuration Guide.

CHAPTER 4

Configuring MPLS Label Imposition

This chapter contains information on how to configure multiprotocol label switching (MPLS) label imposition.

About MPLS Label Imposition

An outgoing label stack having one or more labels can be statically provisioned using the MPLS Label Stack Imposition feature. The outgoing label stack is used in the following two types of statically configured MPLS bindings:

- Prefix and Label to Label Stack - Here an IP prefix or an incoming label is mapped to an outgoing stack, similar to static MPLS. An incoming prefix is mapped to out-label-stack for IP-only ingress traffic.
- Label to Label Stack - Here only an incoming label is mapped to an outgoing stack without any prefix.

The new MPLS binding types are implemented in the static MPLS component and are available only when the **feature mpls segment-routing** command is enabled.

If configured next-hops of MPLS label imposition are SR recursive next-hops (RNH), then they are resolved to actual next-hops using RIB. The outer label of the out-label stack is imposed automatically from the SR allocated labels.

ECMP is also supported by adding a number of path configurations.

Note The static MPLS process is started when either the **feature mpls segment-routing** command or the **feature mpls static** command is run. Certain standard static MPLS commands will not be available when static MPLS is run using the **feature mpls segment-routing** command, and the commands for MPLS bindings will not be available when the **feature mpls static** command is run.

Guidelines and Limitations for MPLS Label Imposition

MPLS label imposition has the following guidelines and limitations:

- MPLS label imposition is supported for the following:
    - Cisco Nexus 9200, 9300, 9300-EX, 9300-FX and 9500 platform switches with the 9400, 9500, 9600, 9700-EX, and 9700-FX line cards.
    - Cisco Nexus 3164Q, 31128PQ, 3232C, and 3264Q switches.
    - Beginning with Cisco NX-OS Release 9.2(1) release, it is supported on Cisco Nexus 9364C Switch.
    - Beginning with Cisco NX-OS Release 9.3(3), it is supported on Cisco Nexus 9364C-GX, 9316D-GX, and 93600CD-GX switches.
- MPLS label imposition supports only IPv4.
- The maximum number of labels in an out-label stack is five for Cisco Nexus 9200, 9300-EX, and 9300-FX platform switches and three for Cisco Nexus 9300 and 9500 platform switches and Cisco Nexus 3164Q, 31128PQ, 3232C, and 3264Q switches. If you try to impose more labels, the trailing label is truncated automatically, and a syslog error message appears signaling to correct the configuration.
- Multicast is not supported for MPLS label imposition.
- In the multi-label stack configuration, changing an outgoing path is allowed only for Cisco Nexus 9200 and 9300-EX Series switches.
- Subinterfaces and port channels are not supported for MPLS label imposition.
- Prefixes and associated subnet masks learned from routing protocols (including from static routes) cannot be used as part of the label stack imposition policy.
- For label stack imposition verified scalability limits, see the Verified Scalability Guide for your device.

Configuring MPLS Label Imposition

Enabling MPLS Label Imposition

You must install and enable the MPLS feature set and then enable the MPLS segment routing feature before you can configure MPLS label imposition.

Procedure

	Command or Action	Purpose

Step 1	**configure terminal**
	Example:
	`switch# configure terminal`
	`switch(config)#`
		Enters global configuration mode.

	Command or Action	Purpose
Step 2	[**no**] **install feature-set mpls** **Example:** `switch(config)# install feature-set mpls`	Installs the MPLS feature set. The **no** form of this command uninstalls the MPLS feature set.
Step 3	[**no**] **feature-set mpls** **Example:** `switch(config)# feature-set mpls`	Enables the MPLS feature set. The **no** form of this command disables the MPLS feature set.
Step 4	[**no**] **feature mpls segment-routing** **Example:** `switch(config)# feature mpls segment-routing`	Enables the MPLS segment routing feature. The **no** form of this command disables the MPLS segment routing feature.

Step 5	(Optional) **show feature-set**
	Example:
	`switch(config)# show feature-set`
	`Feature Set Name      ID        State`
	`--------------------  --------  --------`
	`mpls                  4         enabled`
		Displays the status of the MPLS feature set.

Step 6	(Optional) **show feature | grep segment-routing**
	Example:
	`switch(config)# show feature | grep segment-routing`
	`segment-routing        1        enabled`
		Displays the status of MPLS segment routing.

Reserving Labels for MPLS Label Imposition

You can reserve the labels that are to be statically assigned. Dynamic label allocation is not supported.

Before you begin

Ensure that the MPLS segment routing feature is enabled.

Procedure

	Command or Action	Purpose

Step 1	**configure terminal**
	Example:
	`switch# configure terminal`
	`switch(config)#`
		Enters global configuration mode.

Step 2	[**no**] **mpls label range** *min-value max-value* [**static** *min-static-value max-static-value*]	Reserves a range of labels for static label assignment.

	Command or Action	Purpose
	Example: `switch(config)# mpls label range 17 99 static 100 10000`	The range for the minimum and maximum values is from 16 to 471804.
Step 3	(Optional) **show mpls label range** **Example:** `switch(config)# show mpls label range`	Displays the label range that is configured for static MPLS.
Step 4	(Optional) **copy running-config startup-config** **Example:** `switch(config)# copy running-config startup-config`	Copies the running configuration to the startup configuration.

Configuring MPLS Label Imposition

You can configure MPLS label imposition on the device.

> **Note** The **feature mpls segment-routing** command cannot be enabled when the following commands are in use: **feature nv overlay**, **nv overlay evpn**, **feature vpc**, and **feature vn-segment-vlan-based**.

Before you begin

Ensure that the MPLS segment routing feature is enabled.

Set a static label range as follows: **mpls label range 16 16 static 17 50000**.

Procedure

	Command or Action	Purpose

Step 1	**configure terminal**
	Example:
	`switch# configure terminal`
	`switch(config)#`
		Enters global configuration mode.

Step 2	**interface** *type slot/port*
	Example:
	`switch(config)# interface ethernet 2/2`
	`switch(config-if)#`
		Enters the interface configuration mode for the specified interface.

Step 3	[**no**] **mpls ip forwarding** **Example:** `switch(config-if)# mpls ip forwarding`	Enables MPLS on the specified interface. The **no** form of this command disables MPLS on the specified interface.

	Command or Action	Purpose

Step 4	**mpls static configuration**
	Example:
	`switch(config-if)# mpls static configuration`
	`switch(config-mpls-static)#`
		Enters MPLS static global configuration mode.

Step 5	**address-family ipv4 unicast**
	Example:
	`switch(config-mpls-static)# address-family ipv4 unicast`
	`switch(config-mpls-static-af)#`
		Enters global address family configuration mode for the specified IPv4 address family.

Step 6	**lsp** *name*
	Example:
	`switch(config-mpls-static-af)# lsp lsp1`
	`switch(config-mpls-static-lsp)#`
		Specifies a name for LSP.

Step 7	**in-label** *value* **allocate policy** *prefix*
	Example:
	`switch(config-mpls-static-lsp)# in-label 8100 allocate policy 15.15.1.0/24`
	`switch(config-mpls-static-lsp-inlabel)#`
		Configures an in-label value and a prefix value (optional).

Step 8	**forward**
	Example:
	`switch(config-mpls-static-lsp-inlabel)# forward`
	`switch(config-mpls-static-lsp-inlabel-forw)#`
		Enters the forward mode.

Step 9	**path** *number* **next-hop** *ip-address* **out-label-stack** *label-id label-id* **Example:** `switch(config-mpls-static-lsp-inlabel-forw)# path 1 next-hop 13.13.13.13 out-label-stack 16 3000`	Specifies the path. The maximum number of supported paths is 32.
Step 10	(Optional) **copy running-config startup-config** **Example:** `switch(config-mpls-static-lsp-inlabel-forw)# copy running-config startup-config`	Copies the running configuration to the startup configuration.

Verifying the MPLS Label Imposition Configuration

To display the MPLS label imposition configuration, perform one of the following tasks:

Command	Purpose
show feature | grep segment-routing	Displays the status of MPLS label imposition.
show feature-set	Displays the status of the MPLS feature set.
show forwarding mpls label *label*	Displays MPLS label forwarding statistics for a particular label.
show mpls label range	Displays the label range that is configured for MPLS label imposition.
show mpls static binding {**all** | **ipv4**}	Displays the configured static prefix or label bindings.
show mpls switching [**detail**]	Displays MPLS label switching information.
show running-config mpls static	Displays the running static MPLS configuration.

This example shows sample output for the **show forwarding mpls label 8100** command:

```
slot 1
=======
--------+-----------+-------------------+----------------+-------------+----------------------
Local|Prefix|FEC          |Next-Hop  |Interface | Out Label |Table Id |(Prefix/Tunnel
id)|Label
--------+-----------+-------------------+----------------+-------------+----------------------
8100 |0x1   |25.25.0.0/16 |12.12.1.2 |Po121     |3131 SWAP  |         |                  |
 17
"    |0x1   |25.25.0.0/16 |12.12.2.2 |Eth1/51   |3131 SWAP  |         |                  |
 17
"    |0x1   |25.25.0.0/16 |12.12.3.2 |Vlan122   |3131 SWAP  |         |                  |
 17
"    |0x1   |25.25.0.0/16 |12.12.4.2 |Vlan123   |3131 SWAP  |         |                  |
 17
```

This example shows sample output for the **show mpls static binding all** command:

```
LI_TEST1 25.25.0.0/16: (vrf: default) Incoming label: 8100
LSP Type: POLICY
  Outgoing labels:
     (path 1)  12.12.1.2 3131,17
     (path 2)  12.12.2.2 3131,17
     (path 3)  12.12.3.2 3131,17
     (path 4)  12.12.4.2 3131,17

LI_TEST2 (vrf: default) Incoming label: 8200
LSP Type: XC
  Outgoing labels:
     (path 1)  12.12.3.2 3132,16
     (path 2)  12.12.4.2 3132,16
     (path 3)  12.12.1.2 3132,16
     (path 4)  12.12.2.2 3132,16
```

This example shows sample output for the **show mpls switching** command:

```
Legend:
(P)=Protected, (F)=FRR active, (*)=more labels in stack.

Local      Out-Label  FEC                                               Out-Interface
Next-Hop
```

```
8200       3132       Label 8200
12.12.3.2                                        *
8200       3132       Label 8200
12.12.4.2                                        *
8200       3132       Label 8200
12.12.1.2                                        *
8200       3132       Label 8200
12.12.2.2                                        *

Local      Out-Label  FEC                                          Out-Interface
Next-Hop
8100       3131       Pol 25.25.0.0/16
12.12.1.2                                        *
8100       3131       Pol 25.25.0.0/16
12.12.2.2                                        *
8100       3131       Pol 25.25.0.0/16
12.12.3.2                                        *
8100       3131       Pol 25.25.0.0/16
12.12.4.2                                        *
```

This example shows sample output for the **show running-config mpls static** command:

```
mpls static configuration
  address-family ipv4 unicast
    lsp LI_TEST2
      in-label 8100 allocate policy 25.25.0.0 255.255.0.0
        forward
          path 1 next-hop 12.12.1.2 out-label-stack 3131 17
          path 2 next-hop 12.12.2.2 out-label-stack 3131 17
          path 3 next-hop 12.12.3.2 out-label-stack 3131 17
          path 4 next-hop 12.12.4.2 out-label-stack 3131 17
```

This example shows sample output for the **show running-config mpls static all** command.

```
switch# show running-config mpls static all

!Command: show running-config mpls static all
!Time: Mon Aug 21 14:59:46 2017

version 7.0(3)I7(1)
logging level mpls static 5
mpls static configuration
address-family ipv4 unicast
lsp 9_label_stack_LPM
in-label 72000 allocate policy 71.200.11.0 255.255.255.0
forward
path 1 next-hop 27.1.32.4 out-label-stack 21901 29701 27401 24501 25801
lsp 9_label_stack_LPM_01
in-label 72001 allocate policy 72.201.1.1 255.255.255.255
lsp DRV-01
in-label 71011 allocate policy 71.111.21.0 255.255.255.0
forward
path 1 next-hop 27.1.31.4 out-label-stack implicit-null
lsp DRV-02
in-label 71012 allocate policy 71.111.22.0 255.255.255.0
forward
path 1 next-hop 8.8.8.8 out-label-stack 28901
lsp DRV-03
switch# show forwarding mpls label 72000

slot 1
=======
```

```
--------+-----------+------------------+---------------+------------+-------
Local |Prefix |FEC |Next-Hop |Interface |Out
Label |Table Id |(Prefix/Tunnel id) | | |Label
--------+-----------+------------------+---------------+------------+-------
72000 |0x1 |71.200.11.0/24 |27.1.32.4 |Eth1/21 |21901 SWAP
| | | | | 29701
| | | | | 27401
| | | | | 24501
| | | | | 25801
```

Displaying MPLS Label Imposition Statistics

To monitor MPLS label imposition statistics, perform one of the following tasks:

Command	Purpose
show forwarding [**ipv4**] **adjacency mpls stats**	Displays MPLS IPv4 adjacency statistics (both, packets and bytes). **Note** The Cisco Nexus 9200 and 9300-EX Series switches do not support this command.
show forwarding mpls label *label* **stats** [**platform**]	Displays MPLS label forwarding statistics.
show mpls forwarding statistics [**interface** *type slot/port*]	Displays MPLS forwarding statistics.
show mpls switching labels *low-label-value* [*high-label-value*] [**detail**]	Displays MPLS label switching statistics. The range for the label value is from 0 to 524286.

This example shows sample output for the **show forwarding adjacency mpls stats** command:

```
slot  1
=======

 FEC   next-hop    interface  tx packets   tx bytes  Label info
------ ----------- ---------- ------------ --------- --------------
       12.12.3.2   Vlan122      0           0         SWAP 3131 17
       12.12.3.2   Vlan122      0           0         SWAP 3132 16
       12.12.4.2   Vlan123      0           0         SWAP 3131 17
       12.12.4.2   Vlan123      0           0         SWAP 3132 16
       12.12.1.2   Po121        0           0         SWAP 3131 17
       12.12.1.2   Po121        0           0         SWAP 3132 16
       12.12.2.2   Eth1/51      0           0         SWAP 3131 17
       12.12.2.2   Eth1/51      0           0         SWAP 3132 16
```

This example shows sample output for the **show forwarding mpls label 8100 stats** command:

```
slot  1
=======
--------+-----------+------------------+---------------+------------+-------
Local   |Prefix     |FEC               |Next-Hop       |Interface   |Out
Label   |Table Id   |(Prefix/Tunnel id) |              |            |Label
--------+-----------+------------------+---------------+------------+-------
8100    |0x1        |25.25.0.0/16      |12.12.1.2      |Po121       |3131
  SWAP
```

```
        |           |                   |                |             | 17
    "   |0x1        |25.25.0.0/16       |12.12.2.2       |Eth1/51       |3131
  SWAP
        |           |                   |                |             | 17
    "   |0x1        |25.25.0.0/16       |12.12.3.2       |Vlan122       |3131
  SWAP
        |           |                   |                |             | 17
    "   |0x1        |25.25.0.0/16       |12.12.4.2       |Vlan123       |3131
  SWAP
        |           |                   |                |             | 17

Input Pkts : 126906012            Input Bytes : 64975876096
SWAP Output Pkts: 126959183       SWAP Output Bytes: 65764550340
TUNNEL Output Pkts: 126959053     TUNNEL Output Bytes: 66272319384
```

This example shows sample output for the **show mpls forwarding statistics** command:

```
MPLS software forwarding stats summary:
    Packets/Bytes sent           : 0/0
    Packets/Bytes received       : 0/0
    Packets/Bytes forwarded      : 0/0
    Packets/Bytes originated     : 0/0
    Packets/Bytes consumed       : 0/0
    Packets/Bytes input dropped  : 0/0
    Packets/Bytes output dropped : 0/0
```

Clearing MPLS Label Imposition Statistics

To clear the MPLS label imposition statistics, perform these tasks:

Command	Purpose
clear forwarding [**ipv4**] **adjacency mpls stats**	Clears the MPLS IPv4 adjacency statistics.
clear forwarding mpls stats	Clears the ingress MPLS forwarding statistics.
clear mpls forwarding statistics	Clears the MPLS forwarding statistics.
clear mpls switching label statistics [**interface** *type slot/port*]	Clears the MPLS switching label statistics.

Configuration Examples for MPLS Label Imposition

This example shows how to configure MPLS label imposition by allocating a prefix and an incoming-label to out-label-stack binding:

```
switch(config-if)# mpls static configuration
switch(config-mpls-static)# address-family ipv4 unicast
switch(config-mpls-static-af)# lsp LI_TEST1
switch(config-mpls-static-lsp)# in-label 8100 allocate policy 25.25.0.0/16
switch(config-mpls-static-lsp-inlabel)# forward
switch(config-mpls-static-lsp-inlabel-forw)# path 1 next-hop 12.12.1.2 out-label-stack 3131
 17
switch(config-mpls-static-lsp-inlabel-forw)# path 2 next-hop 12.12.2.2 out-label-stack 3131
```

```
 17
switch(config-mpls-static-lsp-inlabel-forw)# path 3 next-hop 12.12.3.2 out-label-stack 3131
 17
switch(config-mpls-static-lsp-inlabel-forw)# path 4 next-hop 12.12.4.2 out-label-stack 3131
 17
```

To remove a next-hop, you can use

no path 1

To remove the named lsp, you can use

no lsp LI_TEST1

This example shows how to configure MPLS label imposition by allocating an incoming-label to out-label-stack binding (no prefix):

```
switch(config-if)# mpls static configuration
switch(config-mpls-static)# address-family ipv4 unicast
switch(config-mpls-static-af)# lsp LI_TEST1
switch(config-mpls-static-lsp)# in-label 8200 allocate
switch(config-mpls-static-lsp-inlabel)# forward
switch(config-mpls-static-lsp-inlabel-forw)# path 1 next-hop 12.12.3.2 out-label-stack 3132
 16
switch(config-mpls-static-lsp-inlabel-forw)# path 2 next-hop 12.12.4.2 out-label-stack 3132
 16
switch(config-mpls-static-lsp-inlabel-forw)# path 3 next-hop 12.12.1.2 out-label-stack 3132
 16
switch(config-mpls-static-lsp-inlabel-forw)# path 4 next-hop 12.12.2.2 out-label-stack 3132
 16
```

CHAPTER 5

Configuring MPLS Layer 3 VPNs

This chapter describes how to configure Multiprotocol Label Switching (MPLS) Layer 3 Virtual Private Networks (VPNs) on Cisco Nexus 9508 switches.

Information About MPLS Layer 3 VPNs

An MPLS Layer 3 VPN consists of a set of sites that are interconnected by an MPLS provider core network. At each customer site, one or more customer edge (CE) routers or Layer 2 switches attach to one or more provider edge (PE) routers. This section includes the following topics:

- MPLS Layer 3 VPN Definition
- How an MPLS Layer 3 VPN Works
- Components of MPLS Layer 3 VPNs
- Hub-and-Spoke Topology
- OSPF Sham-Link Support for MPLS VPN

MPLS Layer 3 VPN Definition

MPLS-based Layer 3 VPNs are based on a peer model that enables the provider and the customer to exchange Layer 3 routing information. The provider relays the data between the customer sites without direct customer involvement.

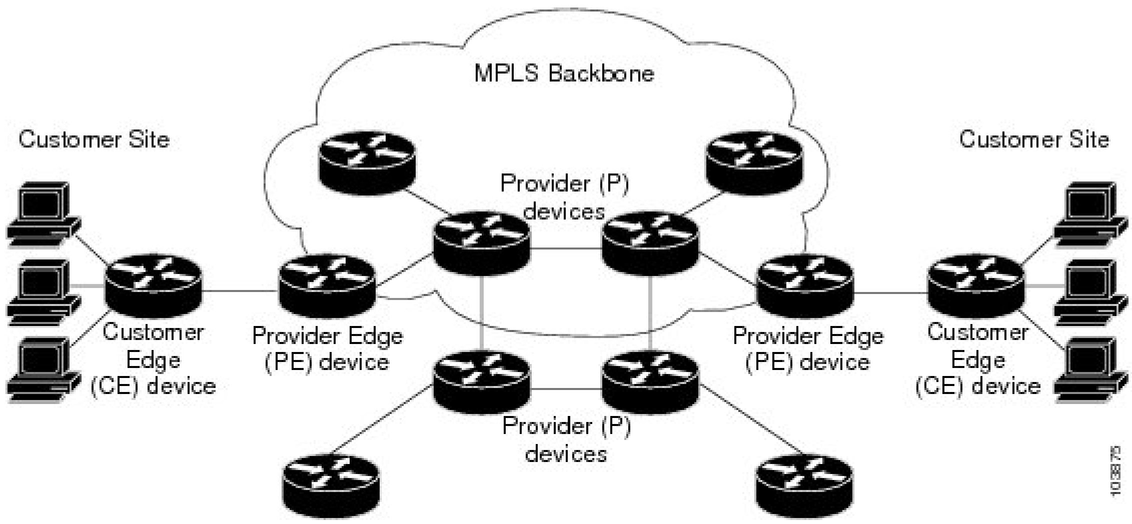
When you add a new site to an MPLS Layer 3 VPN, you must update the provider edge router that provides services to the customer site.

MPLS Layer 3 VPNs include the following components:

- Provider (P) router—A router in the core of the provider network. P routers run MPLS switching and do not attach VPN labels (an MPLS label in each route assigned by the PE router) to routed packets.

- Provider edge (PE) router—A router that attaches the VPN label to incoming packets that are based on the interface or subinterface on which they are received. A PE router attaches directly to a CE router.
- Customer edge (CE) router—An edge router on the network of the provider that connects to the PE router on the network. A CE router must interface with a PE router.

Figure 2: Basic MPLS Layer 3 VPN Terminology



How an MPLS Layer 3 VPN Works

MPLS Layer 3 VPN functionality is enabled at the edge of an MPLS network. The PE router performs the following tasks:

- Exchanges routing updates with the CE router
- Translates the CE routing information into VPN routes
- Exchanges Layer 3 VPN routes with other PE routers through the Multiprotocol Border Gateway Protocol (MP-BGP)

Components of MPLS Layer 3 VPNs

An MPLS-based Layer 3 VPN network has three components:

1. VPN route target communities—A VPN route target community is a list of all members of a Layer 3 VPN community. You must configure the VPN route targets for each Layer 3 VPN community member.
2. Multiprotocol BGP peering of VPN community PE routers—Multiprotocol BGP propagates VRF reachability information to all members of a VPN community. You must configure Multiprotocol BGP peering in all PE routers within a VPN community.
3. MPLS forwarding—MPLS transports all traffic between all VPN community members across a VPN enterprise or service provider network.

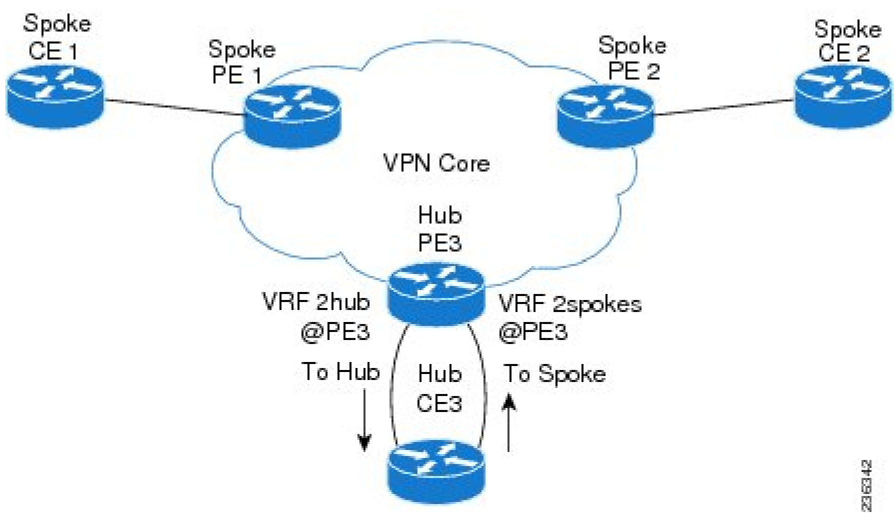
A one-to-one relationship does not necessarily exist between customer sites and VPNs. A site can be a member of multiple VPNs. However, a site can associate with only one VRF. A customer-site VRF contains all the routes that are available to the site from the VPNs of which it is a member.

Hub-and-Spoke Topology

A hub-and-spoke topology prevents local connectivity between subscribers at the spoke provider edge (PE) routers and ensures that a hub site provides subscriber connectivity. Any sites that connect to the same PE router must forward intersite traffic using the hub site. This topology ensures that the routing at the spoke sites moves from the access-side interface to the network-side interface or from the network-side interface to the access-side interface but never from the access-side interface to the access-side interface. A hub-and-spoke topology allows you to maintain access restrictions between sites.

A hub-and-spoke topology prevents situations where the PE router locally switches the spokes without passing the traffic through the hub site. This topology prevents subscribers from directly connecting to each other. A hub-and-spoke topology does not require one VRF for each spoke.

Figure 3: Hub-and-Spoke Topology



As shown in the figure, a hub-and-spoke topology is typically set up with a hub PE that is configured with two VRFs:

- VRF 2hub with a dedicated link connected to the hub customer edge (CE)
- VRF 2spokes with another dedicated link connected to the hub CE.

Interior Gateway Protocol (IGP) or external BGP (eBGP) sessions are usually set up through the hub PE-CE links. The VRF 2hub imports all the exported route targets from all the spoke PEs. The hub CE learns all routes from the spoke sites and readvertises them back to the VRF 2spoke of the hub PE. The VRF 2spoke exports all these routes to the spoke PEs.

If you use eBGP between the hub PE and hub CE, you must allow duplicate autonomous system (AS) numbers in the path which is normally prohibited. You can configure the router to allow this duplicate AS number at the neighbor of VRF 2spokes of the hub PE and also for VPN address family neighbors at all the spoke PEs. In addition, you must disable the peer AS number check at the hub CE when distributing routes to the neighbor at VRF 2spokes of the hub PE.

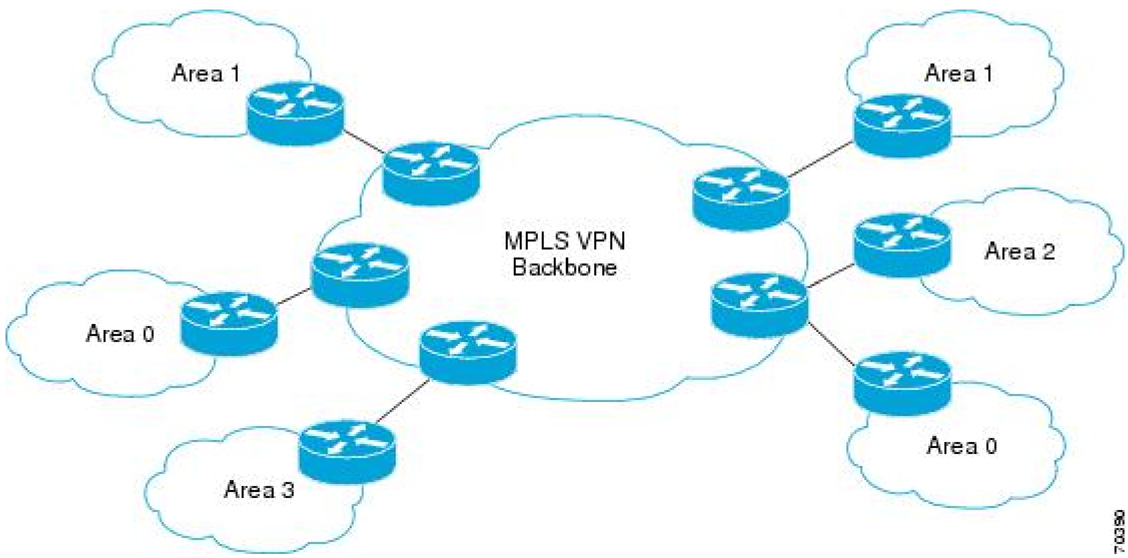
OSPF Sham-Link Support for MPLS VPN

In a Multiprotocol Label Switching (MPLS) VPN configuration, you can use the Open Shortest Path First (OSPF) protocol to connect customer edge (CE) devices to service provider edge (PE) devices in the VPN backbone. Many customers run OSPF as their intrasite routing protocol, subscribe to a VPN service, and want to exchange routing information between their sites using OSPF (during migration or on a permanent basis) over an MPLS VPN backbone.

The benefits of the OSPF sham-link support for MPLS VPN are as follows:

- Client site connection across the MPLS VPN Backbone—A sham link ensures that OSPF client sites that share a backdoor link can communicate over the MPLS VPN backbone and participate in VPN services.
- Flexible routing in an MPLS VPN configuration—In an MPLS VPN configuration, the OSPF cost that is configured with a sham link allows you to decide if OSPF client site traffic is routed over a backdoor link or through the VPN backbone.

The figure below shows an example of how VPN client sites that run OSPF can connect over an MPLS VPN backbone.



When you use OSPF to connect PE and CE devices, all routing information learned from a VPN site is placed in the VPN routing and forwarding (VRF) instance that is associated with the incoming interface. The PE devices that attach to the VPN use the Border Gateway Protocol (BGP) to distribute VPN routes to each other. A CE device can learn the routes to other sites in the VPN by peering with its attached PE device. The MPLS VPN super backbone provides an additional level of routing hierarchy to interconnect the VPN sites that are running OSPF.

When OSPF routes are propagated over the MPLS VPN backbone, additional information about the prefix in the form of BGP extended communities (route type, domain ID extended communities) is appended to the BGP update. This community information is used by the receiving PE device to decide the type of link-state advertisement (LSA) to be generated when the BGP route is redistributed to the OSPF PE-CE process. In this way, internal OSPF routes that belong to the same VPN and are advertised over the VPN backbone are seen as interarea routes on the remote sites.

Prerequisites for MPLS Layer 3 VPNs

MPLS Layer 3 VPNs has the following prerequisites:

- Ensure that you have configured MPLS and Label Distribution Protocol (LDP) in your network. All routers in the core, including the PE routers, must be able to support MPLS forwarding.
- Ensure that you have installed the correct license for MPLS and any other features you will be using with MPLS.

Guidelines and Limitations for MPLS Layer 3 VPNs

MPLS Layer 3 VPNs have the following configuration guidelines and limitations:

- You can configure MPLS Layer 3 VPN (LDP) on Cisco Nexus 3600-R and Cisco Nexus 9504 and 9508 platform switches with the N9K-X9636C-RX, N9K-X9636C-R, N9K-X96136YC-R, and N9K-X9636Q-R line cards.
- Ensure that MPLS IP forwarding is not enabled on the interface which terminates tunnel endpoint, as it is not supported.
- You must enable MPLS IP forwarding on interfaces where the forwarding decisions are made based on the labels of incoming packets. If a VPN label is allocated by per prefix mode, MPLS IP forwarding must be enabled on the link between PE and CE.
- Because of the hardware limitation on the trap resolution on Cisco Nexus 9508 platform switches with the N9K-X9636C-R and N9K-X9636Q-R line cards, uRPF may not be applied on supervisor bound packets via in-band.
- On Cisco Nexus 9500 platform switches with the -R series line cards, RACL is applied only to routed traffic so that the bridge traffic does not hit RACL. This applies to all Multicast OSPF control traffic.
- On Cisco Nexus 9500 platform switches with the -R series line cards, Control Packets with Explicit-NULL label is not prioritized when sending to SUP. This may result in control protocols flapping when explicit-NULL is configured.
- Per-label statistics at a scale of 500K is not supported on Cisco Nexus 9500 platform switches with the -R series line cards because of the hardware limitation.
- ARP scaling on Cisco Nexus 9500 platform switches with the -R series line cards is limited to 64K if all the 64K MACs are different. This limitation also applies if there are several Equal Cost Multiple Paths (ECMP) configured on the interface.
- Packets with MPLS Explicit-NULL may not be parsed correctly with default line card profile.
- MPLS Layer 3 VPNs support the following CE-PE routing protocols:
  - BGP (IPv4 and IPv6)
  - Enhanced Interior Gateway Protocol (EIGRP) (IPv4)
  - Open Shortest Path First (OSPFv2)
  - Routing Information Protocol (RIPv2)

- Set statements in an import route map are ignored.
- The BGP minimum route advertisement interval (MRAI) value for all iBGP and eBGP sessions is zero and is not configurable.
- In a high scale setup with many BGP routes getting redistributed into EIGRP, modify the EIGRP signal timer to ensure that the EIGRP convergence time is higher than the BGP convergence time. This process allows all the BGP routes to be redistributed into EIGRP, before EIGRP signals convergence.
- MPLS Layer 3 VPNs are supported on M3 Series modules.
- When OSPF is used as a protocol between PE and CE devices, the OSPF metric is preserved when routes are advertised over the VPN backbone. The metric is used on the remote PE devices to select the correct route. Do not modify the metric value when OSPF is redistributed to BGP and when BGP is redistributed to OSPF. If you modify the metric value, routing loops might occur.
- MPLS Traffic Engineering (RSVP) is not supported on Cisco Nexus 9508 platform switches with the N9K-X9636C-R and N9K-X9636Q-R line cards, .
- Beginning Cisco NX-OS Release 9.3(1), the behavior of the BGP pre-best path point of insertion (POI) is changed. In this release, the NX-OS RPM, BGP, and HMM software use a single cost community ID (either 128 for internal routes or 129 for external routes) to identify a BGP VPNv4 route as an EIGRP originated route. Only the routes that have the pre-best path value set to cost community ID 128 or 129 are installed in the URIB along with the cost extcommunity. Any non-EIGRP originated route carrying the above described cost community ID would be installed in URIB along with pre-best path cost community. As a result, URIB would use this cost to identify the better route between the route learnt via the iBGP and backdoor-EIGRP instead of the admin distance.

  Only the routes that have the pre-best path value set to cost community ID 128 or 129 are installed in the URIB along with the cost extcommunity.
- The Egress RACL (e-RACL) TCAM and MPLS Extended ECMP features are mutually exclusive. To enable MPLS Extended ECMP (**hardware profile mpls extended-ecmp)** on the Cisco Nexus N9K-X9636C-RX line card, set the e-RACL TCAM carving to 0.

Default Settings for MPLS Layer 3 VPNs

Table 2: Default MPLS Layer 3 VPN Parameters

Parameters	Default
L3VPN feature	Disabled
L3VPN SNMP notifications	Disabled
allowas-in (for a hub-and-spoke topology)	0
disable-peer-as-check (for a hub-and-spoke topology)	Disabled

Configuring MPLS Layer 3 VPNs

About OSPF Domain IDs and Tags

You can set the domain_ID for an OSPF router instance within a VRF. In OSPF, Cisco NX-OS uses the domain_ID and domain tag to control aspects of BGP route redistribution at the provider edge (PE) or customer edge (CE).

- You can configure a primary and secondary domain_ID for the redistributed OSPF routes.
- OSPF also uses a domain tag to identify the OSPF process ID.

The Cisco NX-OS implementation of domain IDs and domain tags complies with RFC 4577.

Note The OSPF primary and secondary domain_IDs and the domain tag are available only when MPLS L3VPN feature is enabled.

Configuring OSPF at the PE and CE Boundary

By using, domain IDs and domain tags, you can configure NX-OS to redistribute OSPF routes into BGP networks, and receive BGP redistributed routes into OSPF at the PE and CE boundary. See the following topics:

- About OSPF Domain IDs and Tags, on page 39
- Configuring the OSPF Domain ID, on page 40
- Configuring the Secondary Domain ID, on page 41
- Configuring the OSPF Domain Tag, on page 39

Configuring the OSPF Domain Tag

The domain tag specifies the OSPF process instance number that NX-OS redistributes into BGP at the PE or CE.

Before you begin

Make sure that MPLS and OSPFv2 are enabled.

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:**	Enters the configuration terminal.

	Command or Action	Purpose

	`switch-1# configure terminal`
	`Enter configuration commands, one per line. End with CNTL/Z.`
	`switch-1(config)#`	

Step 2	**router ospf** *process-tag*
	Example:
	`switch-1(config)# router ospf 101`
	`switch-1(config-router)#`
		Enters router configuration mode to configure the OSPF router instance. The process tag is an alphanumeric string from 1 through 20 characters that identifies the router.

Step 3	**vrf** *vrf-name*
	Example:
	`switch-1(config-router)# vrf pubstest`
	`switch-1(config-router-vrf)#`
		Enter the specific VRF instance for OSPF. The VRF name is an alphanumeric string from 1 through 32 characters that identifies the VRF.

Step 4	**ospf domain-tag** *as-number*
	Example:
	`switch-1(config-router-vrf)# domain-tag 9999`
	`nxosv2(config-router-vrf)#`
		Sets the domain tag. The domain tag is an alphanumeric string from 0 through 2147483647 that identifies the AS number.

Configuring the OSPF Domain ID

You can set the domain_ID for an OSPF router instance within a VRF to control BGP route redistribution into OSPF at the CE or PE.

To remove this feature, use the **no domain-id** command.

Before you begin

Both the MPLS L3VPN and OSPFv2 feature must be enabled to use the OSPF domain_ID feature.

Procedure

	Command or Action	Purpose

Step 1	**configure terminal**
	Example:
	`switch-1# configure terminal`
	`Enter configuration commands, one per line. End with CNTL/Z.`
	`switch-1(config)#`
		Enters the configuration terminal.

Step 2	**router ospf** *process-tag*
	Example:
	`switch-1(config)# router ospf 101`
	`switch-1(config-router)#`
		Enters router configuration mode to configure the OSPF router instance. The process tag is an alphanumeric string from 1 through 20 characters that identifies the router.

	Command or Action	Purpose

Step 3	**vrf** *vrf-name*
	Example:
	`switch-1(config-router)# vrf pubstest`
	`switch-1(config-router-vrf)#`
		Enter the specific VRF instance for OSPF. The VRF name is an alphanumeric string from 1 through 32 characters that identifies the VRF.

Step 4	**domain-id** { *id* | **type** *domain-type* **value** *value* | **Null** } **Example:** `switch-1(config-router-vrf)# domain-id 19.0.2.0`	Sets the domain_ID and additional parameters: • *id* specifies the domain ID in dotted decimal notation, for example, 1.2.3.4 • *type* specifies the domain type in four-byte notation, for example, 0005. • *value* specifies the domain value in 6 bytes of hexadecimal notation, for example, 0x0005. You can use the Null argument to clear the domain_ID.

Configuring the Secondary Domain ID

You can set a secondary domain_ID for an OSPF router instance within a VRF to control BGP route redistribution into OSPF at the CE or PE.

Use the **domain-id Null** command to unconfigure the domain_ID.

Before you begin

Make sure that OSPFv2 and MPLS features are enabled.

Procedure

	Command or Action	Purpose

Step 1	**configure terminal**
	Example:
	`switch-1# configure terminal`
	`Enter configuration commands, one per line. End with CNTL/Z.`
	`switch-1(config)#`
		Enters the configuration terminal.

Step 2	**router ospf** *process-tag*
	Example:
	`switch-1(config)# router ospf 101`
	`switch-1(config-router)#`
		Enters router configuration mode to configure the OSPF router instance. The process tag is an alphanumeric string from 1 through 20 characters that identifies the router.

	Command or Action	Purpose

Step 3	**vrf** *vrf-name*
	Example:
	`switch-1(config-router)# vrf pubstest`
	`switch-1(config-router-vrf)#`
		Enters the specific VRF instance for OSPF. The VRF name is an alphanumeric string from 1 through 32 characters that identifies the VRF.

Step 4	**domain-id** { *id* | **type** *domain-type* **value** *value* | **Null** } **Example:** `switch-1(config-router-vrf)# domain-id 19.0.2.0`	Sets the domain_ID for the autonomous system.

Configuring the Core Network

Assessing the Needs of MPLS Layer 3 VPN Customers

You can identify the core network topology so that it can best serve MPLS Layer 3 VPN customers.

- Identify the size of the network:
    - Identify the following to determine the number of routers and ports you need:
    - How many customers do you need to support?
    - How many VPNs are needed per customer?
    - How many virtual routing and forwarding instances are there for each VPN?
- Determine which routing protocols you need in the core network.
- Determine if you need MPLS VPN high availability support.

> **Note** MPLS VPN nonstop forwarding and graceful restart are supported on select routers and Cisco NX-OS releases. You need to make sure that graceful restart for BGP and LDP is enabled.

- Configure the routing protocols in the core network.
- Determine if you need BGP load sharing and redundant paths in the MPLS Layer 3 VPN core.

Configuring MPLS in the Core

To enable MPLS on all routers in the core, you must configure a label distribution protocol. You can use either of the following as a label distribution protocol:

- MPLS Label Distribution Protocol (LDP).
- MPLS Traffic Engineering Resource Reservation Protocol (RSVP).

Configuring Multiprotocol BGP on the PE Routers and Route Reflectors

You can configure multiprotocol BGP connectivity on the PE routers and route reflectors.

Before you begin

- Ensure that graceful restart is enabled on all routers for BGP and LDP.

Procedure

	Command or Action	Purpose

Step 1	**configure terminal**
	Example:
	`switch# configure terminal`
	`switch(config)#`
		Enters global configuration mode.

Step 2	**feature bgp**
	Example:
	`switch(config)# feature bgp`
	`switch(config)#`
		Enables the BGP feature.

Step 3	**install feature-set mpls**
	Example:
	`switch(config)# install feature-set mpls`
	`switch(config)#`
		Installs the MPLS feature-set.

Step 4	**feature-set mpls**
	Example:
	`switch(config)# feature-set mpls`
	`switch(config)#`
		Enables the MPLS feature-set.

Step 5	**feature mpls l3vpn**
	Example:
	`switch(config)# feature mpls l3vpn`
	`switch(config)#`
		Enables the MPLS Layer 3 VPN feature.

Step 6	**router bgp** *as - number* **Example:** `switch(config)# router bgp 1.1`	Configures a BGP routing process and enters router configuration mode. The as-number argument indicates the number of an autonomous system that identifies the router to other BGP routers and tags the routing information. The AS number can be a 16-bit integer or a 32-bit integer in the form of a higher 16-bit decimal number and a lower 16-bit decimal number in xx.xx format.
Step 7	**router-id** *ip-address* **Example:**	(Optional) Configures the BGP router ID. This IP address identifies this BGP speaker. This command triggers an automatic notification

	Command or Action	Purpose
	`switch(config-router)# router-id 192.0.2.255`	and session reset for the BGP neighbor sessions.

Step 8	**neighbor** *ip-address* **remote-as** *as-number*
	Example:
	`switch(config-router)# neighbor 209.165.201.1 remote-as 1.1`
	`switch(config-router-neighbor)#`
		Adds an entry to the iBGP neighbor table. The ip-address argument specifies the IP address of the neighbor in dotted decimal notation.

Step 9	**address-family { vpnv4 | vpnv6 } unicast**
	Example:
	`switch(config-router-neighbor)# address-family vpnv4 unicast`
	`switch(config-router-neighbor-af)#`
		Enters address family configuration mode for configuring routing sessions, such as BGP, that uses standard VPNv4 or VPNv6 address prefixes.

Step 10	**send-community extended** **Example:** `switch(config-router-neighbor-af)# send-community extended`	Specifies that a communities attribute should be sent to a BGP neighbor.
Step 11	**show bgp { vpnv4 | vpnv6 } unicast neighbors** **Example:** `switch(config-router-neighbor-af)# show bgp vpnv4 unicast neighbors`	(Optional) Displays information about BGP neighbors.
Step 12	**copy running-config startup-config** **Example:** `switch(config-router-vrf)# copy running-config startup-config`	(Optional) Copies the running configuration to the startup configuration.

Connecting the MPLS VPN Customers

Defining VRFs on the PE Routers to Enable Customer Connectivity

You must create VRFs on the PE routers to enable customer connectivity. You configure route targets to control which IP prefixes are imported into the customer VPN site and which IP prefixes are exported to the BGP network. You can optionally use an import or export route map to provide more fine-grained control over the IP prefixes that are imported into the customer VPN site or exported out of the VPN site. You can use a route map to filter routes that are eligible for import or export in a VRF, based on the route target extended community attributes of the route. The route map might, for example, deny access to selected routes from a community that is on the import route target list.

Procedure

	Command or Action	Purpose

Step 1	**configure terminal**
	Example:
	`switch# configure terminal`
	`switch(config)#`
		Enters global configuration mode.

Step 2	**install feature-set mpls**
	Example:
	`switch(config)# install feature-set mpls`
	`switch(config)#`
		Installs the MPLS feature-set.

Step 3	**feature-set mpls**
	Example:
	`switch(config)# feature-set mpls`
	`switch(config)#`
		Enables the MPLS feature-set.

Step 4	**feature-set mpls l3vpn**
	Example:
	`switch(config)# feature-set mpls l3vpn`
	`switch(config)#`
		Enables the MPLS Layer 3 VPN feature.

Step 5	**vrf context** *vrf-name*
	Example:
	`switch(config)# vrf context vpn1`
	
	`switch(config-vrf)#`
		Defines the VPN routing instance by assigning a VRF name and enters VRF configuration mode. The vrf-name argument is any case-sensitive, alphanumeric string up to 32 characters.

Step 6	**rd** *route-distinguisher*
	Example:
	`switch(config-vrf)# rd 1.2:1`
	
	`switch(config-vrf)#`
		Configures the route distinguisher. The route-distinguisher argument adds an 8-byte value to an IPv4 prefix to create a VPN IPv4 prefix. You can enter an RD in either of these formats:
		• 16-bit or 32-bit AS number: your 32-bit number, for example, 1.2:3
		• 32-bit IP address: your 16-bit number, for example, 192.0.2.1:1

Step 7	**address-family { ipv4 | ipv6 } unicast**
	Example:
	`switch(config-vrf)# address-family ipv4 unicast`
	
	`switch(config-vrf-af-ipv4)#`
		Specifies the IPv4 address family type and enters address family configuration mode.

Step 8	**route-target** { **import** | **export** } *route-target-ext-community* **}**	Specifies a route-target extended community for a VRF as follows:

	Command or Action	Purpose

	Example:
	`switch(config-vrf-af-ipv4)# route-target import 1.0:1`
		• The import keyword imports routing information from the target VPN extended community.
		• The export keyword exports routing information to the target VPN extended community.
		• The route-target-ext-community argument adds the route-target extended community attributes to the VRF's list of import or export route-target extended communities. You can enter the route-target-ext-community argument in either of these formats:
		  • 16-bit or 32-bit AS number: your 32-bit number, for example, 1.2:3
		  • 32-bit IP address: your 16-bit number, for example, 192.0.2.1:1

Step 9	**maximum routes** *max-routes* [**threshold** *value*] [**reinstall**] **Example:** `switch(config-vrf-af-ipv4)# maximum routes 10000`	(Optional) Configures the maximum number of routes that can be stored in the VRF route table. The max-routes range is from 1 to 4294967295. The threshold value range is from 1 to 100.
Step 10	**import** [**vrf default** *max-prefix*] **map** *route-map* **Example:** `switch(config-vrf-af-ipv4)# import vrf default map vpn1-route-map`	(Optional) Configures an import policy for a VRF to import prefixes from the default VRF as follows: • The max-prefix range is from 1 to 2147483647. The default is 1000 prefixes. • The route-map argument specifies the route map to be used as an import route map for the VRF and can be any case-sensitive, alphanumeric string up to 63 characters.
Step 11	**show vrf** *vrf-name* **Example:** `switch(config-vrf-af-ipv4)# show vrf vpn1`	(Optional) Displays information about a VRF. The vrf-name argument is any case-sensitive, alphanumeric string up to 32 characters.
Step 12	**copy running-config startup-config** **Example:** `switch(config-router-vrf)# copy running-config startup-config`	(Optional) Copies the running configuration to the startup configuration.

Configuring VRF Interfaces on PE Routers for Each VPN Customer

You can associate a virtual routing and forwarding instance (VRF) with an interface or subinterface on the PE routers.

Procedure

	Command or Action	Purpose

Step 1	**configure terminal**
	Example:
	`switch# configure terminal`
	`switch(config)#`
		Enters global configuration mode.

Step 2	**interface** *type number*
	Example:
	`switch(config)# interface Ethernet 5/0`
	`switch(config-if)#`
		Specifies the interface to configure and enters interface configuration mode as follows:
		• The type argument specifies the type of interface to be configured.
		• The number argument specifies the port, connector, or interface card number.

Step 3	**vrf member** *vrf-name* **Example:** `switch(config-if)# vrf member vpn1`	Associates a VRF with the specified interface or subinterface. The vrf-name argument is the name assigned to a VRF.
Step 4	**show vrf** *vrf-name* **interface** **Example:** `switch(config-if)# show vrf vpn1 interface`	(Optional) Displays information about interfaces associated with a VRF. The vrf-name argument is any case-sensitive alphanumeric string up to 32 characters.
Step 5	**copy running-config startup-config** **Example:** `switch(config-router-vrf)# copy running-config startup-config`	(Optional) Copies the running configuration to the startup configuration.

Configuring Routing Protocols Between the PE and CE Routers

Configuring Static or Directly Connected Routes Between the PE and CE Routers

You can configure the PE router for PE-to-CE routing sessions that use static routes.

Procedure

	Command or Action	Purpose

Step 1	**configure terminal**
	Example:
	`switch# configure terminal`
	`switch(config)#`
		Enters global configuration mode.

	Command or Action	Purpose

Step 2	**vrf context** *vrf-name*
	Example:
	`switch(config)# vrf context vpn1`
	`switch(config-vrf)#`
		Defines the VPN routing instance by assigning a VRF name and enters VRF configuration mode. The vrf-name argument is any case-sensitive, alphanumeric string up to 32 characters.

Step 3	{ **ip ipv6** } **route** *prefix nexthop* **Example:** `switch(config-vrf)# ip route 192.0.2.1/28 ethernet 2/1`	Defines static route parameters for every PE-to-CE session. The prefix and nexthop are as follows: • IPv4—in dotted decimal notation • IPv6—in hex format.

Step 4	**address-family { ipv4 | ipv6 } unicast**
	Example:
	`switch(config-vrf)# address-family ipv4 unicast`
	`switch(config-vrf-af)#`
		Specifies the IPv4 address family type and enters address family configuration mode.

Step 5	**feature bgp** *as - number*
	Example:
	`switch(config-vrf-af)# feature bgp`
	`switch(config)#`
		Enables the BGP feature.

Step 6	**router bgp** *as - number* **Example:** `switch(config)# router bgp 1.1`	Configures a BGP routing process and enters router configuration mode. The as-number argument indicates the number of an autonomous system that identifies the router to other BGP routers and tags the routing information. The AS number can be a 16-bit integer or a 32-bit integer in the form of a higher 16-bit decimal number and a lower 16-bit decimal number in xx.xx format.

Step 7	**vrf** *vrf-name*
	Example:
	`switch(config-router)# vrf vpn1`
	`switch(config--router-vrf)#`
		Associates the BGP process with a VRF.
		The vrf-name argument is any case-sensitive, alphanumeric string up to 32 characters.

Step 8	**address-family { ipv4 | ipv6 } unicast**
	Example:
	`switch(config-vrf)# address-family ipv4 unicast`
	`switch(config-vrf-af)#`
		Specifies the IPv4 address family type and enters address family configuration mode.

	Command or Action	Purpose
Step 9	**redistribute static route-map** *map-name* **Example:** `switch(config-router-vrf-af)# redistribute static route-map StaticMap`	Redistributes static routes into BGP. The map-name can be any case-sensitive, alphanumeric string up to 63 characters.
Step 10	**redistribute direct route-map** *map-name* **Example:** `switch(config-router-vrf-af)# redistribute direct route-map StaticMap`	Redistributes directly connected routes into BGP. The map-name can be any case-sensitive, alphanumeric string up to 63 characters.
Step 11	**show { ipv4 | ipv6 } route vrf** *vrf-name* **Example:** `switch(config-router-vrf-af)# show ip ipv4 route vrf vpn1`	(Optional) Displays information about routes. The vrf-name argument is any case-sensitive, alphanumeric string up to 32 characters.
Step 12	**copy running-config startup-config** **Example:** `switch(config-router-vrf)# copy running-config startup-config`	(Optional) Copies the running configuration to the startup configuration.

Configuring BGP as the Routing Protocol Between the PE and CE Routers

You can use eBGP to configure the PE router for PE-to-CE routing sessions.

Procedure

	Command or Action	Purpose

Step 1	**configure terminal**
	Example:
	`switch# configure terminal`
	`switch(config)#`
		Enters global configuration mode.

Step 2	**feature bgp**
	Example:
	`switch(config)# feature bgp`
	`switch(config)#`
		Enables the BGP feature.

Step 3	**router bgp** *as - number*
	Example:
	`switch(config)# router bgp 1.1`
	`switch(config-router)#`
		Configures a BGP routing process and enters router configuration mode.
		The as-number argument indicates the number of an autonomous system that identifies the router to other BGP routers and tags the routing information passed along. The AS number can be a 16-bit integer or a 32-bit integer in the form of a higher 16-bit decimal number and a lower 16-bit decimal number in xx.xx format.

	Command or Action	Purpose

Step 4	**vrf** *vrf-name*
	Example:
	`switch(config-router)# vrf vpn1`
	`switch(config--router-vrf)#`
		Associates the BGP process with a VRF.
		The vrf-name argument is any case-sensitive, alphanumeric string up to 32 characters.

Step 5	**neighbor** *ip-address***remote-as** *as-number*
	Example:
	`switch(config-router)# neighbor 209.165.201.1 remote-as 1.1`
	`switch(config-router-neighbor)#`
		Adds an entry to the iBGP neighbor table. The ip-address argument specifies the IP address of the neighbor in dotted decimal notation. The as-number argument specifies the autonomous system to which the neighbor belongs.

Step 6	**address-family { ipv4 | ipv6 } unicast**
	Example:
	`switch(config-vrf)# address-family ipv4 unicast`
	`switch(config-vrf-af)#`
		Enters address family configuration mode for configuring routing sessions, such as BGP, that use standard IPv4 or IPv6 address prefixes.

Step 7	**show bgp { vpnv4 | vpnv6 } unicast neighbors vrf** *vrf-name* **Example:** `switch(config-router-neighbor-af)# show bgp vpnv4 unicast neighbors`	(Optional) Displays information about BGP neighbors. The vrf-name argument is any case-sensitive alphanumeric string up to 32 characters.
Step 8	**copy running-config startup-config** **Example:** `switch(config-router-vrf)# copy running-config startup-config`	(Optional) Copies the running configuration to the startup configuration.

Configuring RIPv2 Between the PE and CE Routers

You can use RIP to configure the PE router for PE-to-CE routing sessions.

Procedure

	Command or Action	Purpose

Step 1	**configure terminal**
	Example:
	`switch# configure terminal`
	`switch(config)#`
		Enters global configuration mode.

Step 2	**feature rip**
	Example:
	`switch(config)# feature rip`
	`switch(config)#`
		Enables the RIP feature.

	Command or Action	Purpose
Step 3	**router rip** *instance-tag* **Example:** `switch(config)# router rip Test1`	Enables RIP and enters router configuration mode. The instance-tag can be any case-sensitive, alphanumeric string up to 20 characters.

Step 4	**vrf** *vrf-name*
	Example:
	`switch(config-router)# vrf vpn1`
	`switch(config--router-vrf)#`
		Associates the RIP process with a VRF.
		The vrf-name argument is any case-sensitive, alphanumeric string up to 32 characters.

Step 5	**address-family ipv4 unicast**
	Example:
	`switch(config-router-vrf)# address-family ipv4 unicast`
	`switch(config-router-vrf-af)#`
		Specifies the address family type and enters address family configuration mode.

Step 6	**redistribute** { **bgp** as | **direct** | { **egrip** | **ospf** | **rip** } *instance-tag* | static } **route-map** *map-name vrf-name* **Example:** `switch(config-router-vrf-af)# show ip rip vrf vpn1`	Redistributes routes from one routing domain into another routing domain. The as number can be a 16-bit integer or a 32-bit integer in the form of a higher 16-bit decimal number and a lower 16-bit decimal number in xx.xx format. The instance-tag can be any case-sensitive alphanumeric string up to 20 characters.
Step 7	**show ip rip vrf** *vrf-name* **Example:** `switch(config-router-vrf-af)# show ip rip vrf vpn1`	(Optional) Displays information about RIP. The vrf-name argument is any case-sensitive, alphanumeric string up to 32 characters.
Step 8	**copy running-config startup-config** **Example:** `switch(config-router-vrf)# copy running-config startup-config`	(Optional) Copies the running configuration to the startup configuration.

Configuring OSPF Between the PE and CE Routers

You can use OSPFv2 to configure the PE router for PE-to-CE routing sessions. You can optionally create an OSPF sham link if you have OSPF back door links that are not part of the MPLS network.

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:**	Enters global configuration mode.

	Command or Action	Purpose
	`switch# configure terminal` `switch(config)#`	

Step 2	**feature ospf**
	Example:
	`switch(config)# feature ospf`
	`switch(config)#`
		Enables the OSPF feature.

Step 3	**router ospf** *instance-tag* **Example:** `switch(config)# router ospf Test1`	Enables OSPF and enters router configuration mode. The instance-tag can be any case-sensitive, alphanumeric string up to 20 characters.

Step 4	**vrf** *vrf-name*
	Example:
	`switch(config-router)# vrf vpn1`
	`switch(config--router-vrf)#`
		Enters router VRF configuration mode.
		The vrf-name argument is any case-sensitive, alphanumeric string up to 32 characters.

Step 5	**area** area-id **sham-link** *source-address destination-address* **Example:** `switch(config-router-vrf)# area 1 sham-link 10.2.1.1 10.2.1.2`	(Optional) Configures the sham link on the PE interface within a specified OSPF area and with the loopback interfaces specified by the IP addresses as endpoints. You must configure the sham link at both PE endpoints.

Step 6	**address-family { ipv4 | ipv6 } unicast**
	Example:
	`switch(config-router)# address-family ipv4 unicast`
	`switch(config-router-vrf-af)#`
		Specifies the address family type and enters address family configuration mode.

Step 7	**redistribute** { **bgp** as | **direct** | { **egrip** | **ospf** | **rip** } *instance-tag* | static } **route-map** *map-name* **Example:** `switch(config-router-vrf-af)# redistribute bgp 1.0 route-map BGPMap`	Redistributes BGP into the EIGRP. The autonomous system number of the BGP network is configured in this step. BGP must be redistributed into EIGRP for the CE site to accept the BGP routes that carry the EIGRP information. A metric must also be specified for the BGP network. The map-name can be any case-sensitive, alphanumeric string up to 63 characters.

Step 8	**autonomous-system** *as-number*
	Example:
	`switch(config-router-vrf-af)#`
	`autonomous-system 1.3`
		(Optional) Specifies the autonomous system number for this address family for the customer site.
		The as-number argument indicates the number of an autonomous system that identifies the

	Command or Action	Purpose
		router to other BGP routers and tags the routing information passed along. The AS number can be a 16-bit integer or a 32-bit integer in the form of a higher 16-bit decimal number and a lower 16-bit decimal number in xx.xx format.
Step 9	**show ip egrip vrf** *vrf-name* **Example:** `switch(config-router-vrf-af)# show ipv4 eigrp vrf vpn1`	(Optional) Displays information about EIGRP in this VRF. The vrf-name can be any case-sensitive, alphanumeric string up to 32 characters
Step 10	**copy running-config startup-config** **Example:** `switch(config-router-vrf)# copy running-config startup-config`	(Optional) Copies the running configuration to the startup configuration.

Configuring EIGRP Between the PE and CE Routers

You can configure the PE router to use Enhanced Interior Gateway Routing Protocol (EIGRP) between the PE and CE routers to transparently connect EIGRP customer networks through an MPLS-enabled BGP core network so that EIGRP routes are redistributed through the VPN across the BGP network as internal BGP (iBGP) routes.

Before you begin

You must configure BGP in the network core.

Procedure

	Command or Action	Purpose

Step 1	**configure terminal**
	Example:
	`switch# configure terminal`
	`switch(config)#`
		Enters global configuration mode.

Step 2	**feature eigrp**
	Example:
	`switch(config)# feature eigrp`
	`switch(config)#`
		Enables the EIGRP feature.

Step 3	**router eigrp** *instance-tag* **Example:** `switch(config)# router eigrp Test1`	Configures an EIGRP instance and enters router configuration mode. The instance-tag can be any case-sensitive, alphanumeric string up to 20 characters.
Step 4	**vrf** *vrf-name* **Example:**	Enters router VRF configuration mode.

	Command or Action	Purpose
	`switch(config-router)# vrf vpn1` `switch(config-router-vrf)#`	The vrf-name argument is any case-sensitive, alphanumeric string up to 32 characters.

Step 5	**address-family ipv4 unicast**
	Example:
	`switch(config-router-vrf)# address-family ipv4 unicast`
	`switch(config-router-vrf-af)#`
		(Optional) Enters address family configuration mode for configuring routing sessions that use standard IPv4 address prefixes.

Step 6	**redistribute bgp** *as-number* **route-map** *map-name* **Example:** `switch(config-router-vrf-af)# redistribute bgp 235354 route-map mtest1`	Redistributes routes from one routing domain into another routing domain. The *as number* can be a 16-bit integer or a 32-bit integer in the form of a higher 16-bit decimal number and a lower 16-bit decimal number in xx.xx format. The instance-tag can be any case-sensitive alphanumeric string up to 20 characters
Step 7	**show ip ospf** *instance-tag* **vrf** *vrf-name* **Example:** `switch(config-router-vrf-af)# show ip rip vrf vpn1`	(Optional) Displays information about OSPF.
Step 8	**copy running-config startup-config** **Example:** `switch(config-router-vrf)# copy running-config startup-config`	(Optional) Copies the running configuration to the startup configuration.

Configuring PE-CE Redistribution in BGP for the MPLS VPN

You must configure BGP to distribute the PE-CE routing protocol on every PE router that provides MPLS Layer 3 VPN services if the PE-CE protocol is not BGP.

Procedure

	Command or Action	Purpose

Step 1	**configure terminal**
	Example:
	`switch# configure terminal`
	`switch(config)#`
		Enters global configuration mode.

Step 2	**feature bgp**
	Example:
	`switch(config)# feature bgp`
	`switch(config)#`
		Enables the BGP feature.

	Command or Action	Purpose

Step 3	**router bgp** *instance-tag*
	Example:
	`switch(config)# router bgp 1.1`
	`switch(config-router)#`
		Configures a BGP routing process and enters router configuration mode. The as-number argument indicates the number of an autonomous system that identifies the router to other BGP routers and tags the routing information passed along. The AS number can be a 16-bit integer or a 32-bit integer in the form of a higher 16-bit decimal number and a lower 16-bit decimal number in xx.xx format.

Step 4	**router id** *ip-address*
	Example:
	`switch(config-router)# router-id 192.0.2.255 1`
	`switch(config-router)#`
		(Optional) Configures the BGP router ID. This IP address identifies this BGP speaker. This command triggers an automatic notification and session reset for the BGP neighbor sessions.

Step 5	**router id** *ip-address* **remote-as** *as-number*
	Example:
	`switch(config-router)# neighbor 209.165.201.1 remote-as 1.2`
	`switch(config-router-neighbor)#`
		Adds an entry to the BGP or multiprotocol BGP neighbor table. The ip-address argument specifies the IP address of the neighbor in dotted decimal notation. The as-number argument specifies the autonomous system to which the neighbor belongs.

Step 6	**update-source loopback [0 | 1]** **Example:** `switch(config-router-neighbor)# update-source loopback 0#`	Specifies the source address of the BGP session.

Step 7	**address-family { ipv4 | ipv6 } unicast**
	Example:
	`switch(config-router-neighbor)# address-family vpnv4`
	`switch(config-router-neighbor-af)#`
		Enters address family configuration mode for configuring routing sessions, such as BGP, that use standard VPNv4 or VPNv6 address prefixes. The optional unicast keyword specifies VPNv4 or VPNv6 unicast address prefixes.

Step 8	**send-community extended** **Example:** `switch(config-router-neighbor-af)# send-community extended`	Specifies that a communities attribute should be sent to a BGP neighbor.

Step 9	**vrf** *vrf-name*
	Example:
	`switch(config-router-neighbor-af)# vrf vpn1`
	`switch(config-router-vrf)#`
		Enters router VRF configuration mode.
		The vrf-name argument is any case-sensitive, alphanumeric string up to 32 characters.

	Command or Action	Purpose
Step 10	**address-family { ipv4 | ipv6 } unicast** **Example:** `switch(config-router-vrf)# address-family ipv4 unicast switch(config-router-vrf-af)#`	Enters address family configuration mode for configuring routing sessions that use standard IPv4 or IPv6 address prefixes.
Step 11	**redistribute** { **direct** | { **egrip** | **ospfv3** | **ospfv3** |**rip** } *instance-tag* | static } **route-map** *map-name* **Example:** `switch(config-router-af-vrf)# redistribute eigrp Test2 route-map EigrpMap`	Redistributes routes from one routing domain into another routing domain. The as number can be a 16-bit integer or a 32-bit integer in the form of a higher 16-bit decimal number and a lower 16-bit decimal number in xx.xx format. The instance-tag can be any case-sensitive, alphanumeric string up to 20 characters. The map-name can be any case-sensitive alphanumeric string up to 63 characters.
Step 12	**show bgp { ipv4 | ipv6 } unicast vrf** *vrf-name* **Example:** `switch(config-router--vrf-af)# show bgp ipv4 unicast vrf vpn1vpn1`	(Optional) Displays information about BGP. The vrf-name argument is any case-sensitive, alphanumeric string up to 32 characters.
Step 13	**copy running-config startup-config** **Example:** `switch(config-router-vrf)# copy running-config startup-config`	(Optional) Copies the running configuration to the startup configuration.

Configuring a Hub-and-Spoke Topology

Configuring VRFs on the Hub PE Router

You can configure hub and spoke VRFs on the hub PE router.

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `switch# configure terminal switch(config)#`	Enters global configuration mode.
Step 2	**install feature-set mpls** **Example:** `switch(config)# install feature-set mpls switch(config)#`	Installs the MPLS feature-set.

	Command or Action	Purpose

Step 3	**feature-set mpls**
	Example:
	`switch(config)# feature-set mpls`
	`switch(config)#`
		Enables the MPLS feature-set.

Step 4	**feature-set mpls l3vpn**
	Example:
	`switch(config)# feature-set mpls l3vpn`
	`switch(config)#`
		Enables the MPLS Layer 3 VPN feature.

Step 5	**vrf context** *vrf-hub*
	Example:
	`switch(config)# vrf context 2hub`
	`switch(config-vrf)#`
		Defines the VPN routing instance for the PE hub by assigning a VRF name and enters VRF configuration mode. The vrf-hub argument is any case-sensitive alphanumeric string up to 32 characters.

Step 6	**rd** *route-distinguisher*
	Example:
	`switch(config-vrf)# rd 1.2:1`
	`switch(config-vrf)#`
		Configures the route distinguisher. The route-distinguisher argument adds an 8-byte value to an IPv4 prefix to create a VPN IPv4 prefix. You can enter an RD in either of these formats:
		• 16-bit or 32-bit AS number: your 32-bit number, for example, 1.2:3
		• 32-bit IP address: your 16-bit number, for example, 192.0.2.1:1

Step 7	**address-family { ipv4 | ipv6 } unicast**
	Example:
	`switch(config-vrf)# address-family ipv4 unicast`
	`switch(config-vrf-af-ipv4)#`
		Specifies the IPv4 address family type and enters address family configuration mode.

Step 8	**route-target** { **import** | **export** } route-target-ext-community **}** **Example:** `switch(config-vrf-af-ipv4)# route-target import 1.0:1`	Specifies a route-target extended community for a VRF as follows: • The **import** keyword imports routing information from the target VPN extended community. • The **export** keyword exports routing information to the target VPN extended community. • The route-target-ext-community argument adds the route-target extended community attributes to the VRF's list of import or export route-target extended communities. You can enter the

	Command or Action	Purpose
		route-target-ext-community argument in either of these formats: • 16-bit or 32-bit AS number: your 32-bit number, for example, 1.2:3 • 32-bit IP address: your 16-bit number, for example, 192.0.2.1:1

Step 9	**vrf context** *vrf-spoke*
	Example:
	`switch(config-vrf-af-ipv4)# vrf context 2spokes`
	`switch(config-vrf)#`
		Defines the VPN routing instance for the PE spoke by assigning a VRF name and enters VRF configuration mode. The vrf-spoke argument is any case-sensitive, alphanumeric string up to 32 characters.

Step 10	**address-family { ipv4 | ipv6 } unicast**
	Example:
	`switch(config-vrf)# address-family ipv4 unicast`
	`switch(config-vrf-af-ipv4)#`
		Specifies the IPv4 address family type and enters address family configuration mode.

Step 11	**route-target** { **import** | **export** } *route-target-ext-community* **}** **Example:** `switch(config-vrf-af-ipv4)# route-target export 1:100`	Specifies a route-target extended community for a VRF as follows: • Creates a route-target extended community for a VRF. The **import** keyword imports routing information from the target VPN extended community. The **export** keyword exports routing information to the target VPN extended community. The route-target-ext-community argument adds the route-target extended community attributes to the VRF's list of import or export route-target extended communities. You can enter the route-target-ext-community argument in either of these formats: • 16-bit or 32-bit AS number: your 32-bit number, for example, 1.2:3 • 32-bit IP address: your 16-bit number, for example, 192.0.2.1:1
Step 12	**show running-config vrf** *vrf-name* **Example:**	(Optional) Displays the running configuration for the VRF.

	Command or Action	Purpose
	`switch(config-vrf-af-ipv4)# show running-config vrf 2spokes`	The vrf-name argument is any case-sensitive, alphanumeric string up to 32 characters. .
Step 13	**copy running-config startup-config** **Example:** `switch(config-router-vrf)# copy running-config startup-config`	(Optional) Copies the running configuration to the startup configuration.

Configuring eBGP on the Hub PE Router

You can use eBGP to configure PE-to-CE hub routing sessions.

Note If all CE sites are using the same BGP AS number, you must perform the following tasks:

- Configure either the BGP **as-override** command at the PE (hub) or the **allowas-in** command at the receiving CE router.
- To advertise BGP routes learned from one ASN back to the same ASN, configure the **disable-peer-as-check** command at the PE router to prevent loopback.

Procedure

	Command or Action	Purpose

Step 1	**configure terminal**
	
	Example:
	
	`switch# configure terminal`
	`switch(config)#`
		Enters global configuration mode.

Step 2	**feature-set mpls** **Example:** `switch(config)# feature-set mpls`	Enables the MPLS feature-set.
Step 3	**feature mpls l3vpn** **Example:** `switch(config)# feature mpls l3vpn`	Enables the MPLS Layer 3 VPN feature.

Step 4	**feature bgp**
	
	Example:
	
	`switch(config)# feature bgp`
	`switch(config)#`
		Enables the BGP feature.

Step 5	**router bgp** *as - number* **Example:**	Configures a BGP routing process and enters router configuration mode.

	Command or Action	Purpose
	`switch(config)# router bgp 1.1` `switch(config-router)#`	The *as-number* argument indicates the number of an autonomous system that identifies the router to other BGP routers and tags the routing information passed along. The AS number can be a 16-bit integer or a 32-bit integer in the form of a higher 16-bit decimal number and a lower 16-bit decimal number in xx.xx format.

Step 6	**neighbor** *ip-address* **remote-as** *as-number*
	Example:
	`switch(config-router)# neighbor 209.165.201.1 remote-as 1.2`
	`switch(config-router-neighbor)#`
		Adds an entry to the iBGP neighbor table.
		• The ip-address argument specifies the IP address of the neighbor in dotted decimal notation.
		• The as-number argument specifies the autonomous system to which the neighbor belongs.

Step 7	**address-family** { **ipv4** | **ipv6** } **unicast**
	Example:
	`switch(config-router-vrf-neighbor)# address-family ipv4 unicast`
	`switch(config-router-neighbor-af)#`
		Specifies the IP address family type and enters address family configuration mode.

Step 8	**send-community extended** **Example:** `switch(config-router-neighbor-af)# send-community extended`	(Optional) Configures BGP to advertise extended community lists.

Step 9	**vrf** *vrf-hub*
	Example:
	`switch(config-router-neighbor-af)# vrf 2hub`
	`switch(config-router-vrf)#`
		Enters VRF configuration mode. The *vrf-hub* argument is any case-sensitive, alphanumeric string up to 32 characters.

Step 10	**neighbor** *ip-address* **remote-as** *as-number*
	Example:
	`switch(config-router-vrf)# neighbor 33.0.0.33 1 remote-as 150`
	`switch(config-router-vrf-neighbor)#`
		Adds an entry to the BGP or multiprotocol BGP neighbor table for this VRF.
		• The ip-address argument specifies the IP address of the neighbor in dotted decimal notation.
		• The as-number argument specifies the autonomous system to which the neighbor belongs.

Step 11	**address-family** { **ipv4** | **ipv6** } **unicast** **Example:**	Specifies the IP address family type and enters address family configuration mode.

	Command or Action	Purpose
	`switch(config-router-vrf-neighbor)# address-family ipv4 unicast switch(config-router--vrf-neighbor-af)#`	
Step 12	**as-override** **Example:** `switch(config-router-vrf-neighbor-af)# as-override`	(Optional) Overrides the AS-number when sending an update. If all BGP sites are using the same AS number, of the following commands: • Configure the BGP as-override command at the PE (hub) or • Configure the allowas-in command at the receiving CE router.
Step 13	**vrf** *vrf-spoke* **Example:** `switch(config-router-vrf-neighbor-af)# vrf 2spokes switch(config-router-vrf)#`	Enters VRF configuration mode. The vrf-spoke argument is any case-sensitive, alphanumeric string up to 32 characters.
Step 14	**neighbor** *ip-address* **remote-as** *as-number* **Example:** `switch(config-router-vrf)# neighbor 33.0.0.33 1 remote-as 150 switch(config-router-vrf-neighbor)#`	Adds an entry to the BGP or multiprotocol BGP neighbor table for this VRF. • The ip-address argument specifies the IP address of the neighbor in dotted decimal notation. • The as-number argument specifies the autonomous system to which the neighbor belongs.
Step 15	**address-family** { **ipv4** | **ipv6** } **unicast** **Example:** `switch(config-router-vrf-neighbor)# address-family ipv4 unicast switch(config-router--vrf-neighbor-af)#`	Specifies the IP address family type and enters address family configuration mode.
Step 16	**allowas-in** [*number*] **Example:** `switch(config-router-vrf-neighbor-af)# allowas-in 3`	(Optional) Allows duplicate AS numbers in the AS path. Configure this parameter in the VPN address family configuration mode at the PE spokes and at the neighbor mode at the PE hub.
Step 17	**show running-config bgp** *vrf-name* **Example:** `switch(config-router-vrf-neighbor-af)# show running-config bgp`	(Optional) Displays the running configuration for BGP.

	Command or Action	Purpose
Step 18	**copy running-config startup-config** **Example:** `switch(config-router-vrf)# copy running-config startup-config`	(Optional) Copies the running configuration to the startup configuration.

Configuring eBGP on the Hub CE Router

You can use eBGP to configure PE-to-CE hub routing sessions.

Note If all CE sites are using the same BGP AS number, you must perform the following tasks:

- Configure either the as-override command at the PE (hub) or the allowas-in command at the receiving CE router.
- Configure the disable-peer-as-check command at the CE router.
- To advertise BGP routes learned from one ASN back to the same ASN, configure the disable-peer-as-check command at the PE router to prevent loopback.

Procedure

	Command or Action	Purpose

Step 1	**configure terminal**
	Example:
	`switch# configure terminal`
	`switch(config)#`
		Enters global configuration mode.

Step 2	**feature-set mpls** **Example:** `switch(config)# feature-set mpls`	Enables the MPLS feature-set.
Step 3	**feature mpls l3vpn** **Example:** `switch(config)# feature mpls l3vpn`	Enables the MPLS Layer 3 VPN feature.

Step 4	**feature bgp**
	Example:
	`switch(config)# feature bgp`
	`switch(config)#`
		Enables the BGP feature.

Step 5	**router bgp** *as - number*
	Example:
	`switch(config)# router bgp 1.1`
	`switch(config-router)#`
		Configures a BGP routing process and enters router configuration mode.
		The *as-number* argument indicates the number of an autonomous system that identifies the router to other BGP routers and tags the

	Command or Action	Purpose
		routing information passed along. The AS number can be a 16-bit integer or a 32-bit integer in the form of a higher 16-bit decimal number and a lower 16-bit decimal number in xx.xx format.

Step 6	**neighbor** *ip-address***remote-as** *as-number*
	Example:
	`switch(config-router)# neighbor 209.165.201.1 remote-as 1.2`
	`switch(config-router-neighbor)#`
		Adds an entry to the iBGP neighbor table.
		• The ip-address argument specifies the IP address of the neighbor in dotted decimal notation.
		• The as-number argument specifies the autonomous system to which the neighbor belongs.

Step 7	**address-family { ipv4 | ipv6 } unicast**
	Example:
	`switch(config-router-vrf-neighbor)# address-family ipv4 unicast`
	`switch(config-router-neighbor-af)#`
		Specifies the IP address family type and enters address family configuration mode.

Step 8	**send-community extended** **Example:** `switch(config-router-neighbor-af)# send-community extended`	(Optional) Configures BGP to advertise extended community lists.

Step 9	**vrf** *vrf-hub*
	Example:
	`switch(config-router-neighbor-af)# vrf 2hub`
	`switch(config-router-vrf)#`
		Enters VRF configuration mode. The *vrf-hub* argument is any case-sensitive, alphanumeric string up to 32 characters.

Step 10	**neighbor** *ip-address***remote-as** *as-number*
	Example:
	`switch(config-router-vrf)# neighbor 33.0.0.33 1 remote-as 150`
	`switch(config-router-vrf-neighbor)#`
		Adds an entry to the BGP or multiprotocol BGP neighbor table for this VRF.
		• The ip-address argument specifies the IP address of the neighbor in dotted decimal notation.
		• The as-number argument specifies the autonomous system to which the neighbor belongs.

Step 11	**address-family { ipv4 | ipv6 } unicast**
	Example:
	`switch(config-router-vrf-neighbor)# address-family ipv4 unicast`
	`switch(config-router--vrf-neighbor-af)#`
		Specifies the IP address family type and enters address family configuration mode.

	Command or Action	Purpose
Step 12	**as-override** **Example:** `switch(config-router-vrf-neighbor-af)# as-override`	(Optional) Overrides the AS-number when sending an update. If all BGP sites are using the same AS number, of the following commands: • Configure the BGP **as-override** command at the PE (hub) or • Configure the **allowas-in** command at the receiving CE router.

Step 13	**vrf** *vrf-spoke*
	Example:
	`switch(config-router-vrf-neighbor-af)# vrf 2spokes`
	`switch(config-router-vrf)#`
		Enters VRF configuration mode. The vrf-spoke argument is any case-sensitive, alphanumeric string up to 32 characters.

Step 14	**neighbor** *ip-address***remote-as** *as-number*
	Example:
	`switch(config-router-vrf)# neighbor 33.0.0.33 1 remote-as 150`
	`switch(config-router-vrf-neighbor)#`
		Adds an entry to the BGP or multiprotocol BGP neighbor table for this VRF.
		• The ip-address argument specifies the IP address of the neighbor in dotted decimal notation.
		• The as-number argument specifies the autonomous system to which the neighbor belongs.

Step 15	**address-family { ipv4 | ipv6 } unicast**
	Example:
	`switch(config-router-vrf-neighbor)# address-family ipv4 unicast`
	`switch(config-router--vrf-neighbor-af)#`
		Specifies the IP address family type and enters address family configuration mode.

Step 16	**allowas-in** [*number*] **Example:** `switch(config-router-vrf-neighbor-af)# allowas-in 3`	(Optional) Allows duplicate AS numbers in the AS path. Configure this parameter in the VPN address family configuration mode at the PE spokes and at the neighbor mode at the PE hub.
Step 17	**show running-config bgp** *vrf-name* **Example:** `switch(config-router-vrf-neighbor-af)# show running-config bgp`	(Optional) Displays the running configuration for BGP.
Step 18	**copy running-config startup-config** **Example:**	(Optional) Copies the running configuration to the startup configuration.

	Command or Action	Purpose
	`switch(config-router-vrf)# copy running-config startup-config`	

Configuring VRFs on the Spoke PE Router

You can configure hub and spoke VRFs on the spoke PE router.

Procedure

	Command or Action	Purpose

Step 1	**configure terminal**
	Example:
	`switch# configure terminal`
	`switch(config)#`
		Enters global configuration mode.

Step 2	**install feature-set mpls**
	Example:
	`switch(config)# install feature-set mpls`
	`switch(config)#`
		Installs the MPLS feature set.

Step 3	**feature-set mpls**
	Example:
	`switch(config)# feature-set mpls`
	`switch(config)#`
		Enables the MPLS feature-set.

Step 4	**feature-set mpls l3vpn**
	Example:
	`switch(config)# feature-set mpls l3vpn`
	`switch(config)#`
		Enables the MPLS Layer 3 VPN feature.

Step 5	**vrf context** *vrf-spoke*
	Example:
	`switch(config)# vrf context spoke`
	`switch(config-vrf)#`
		Defines the VPN routing instance for the PE spoke by assigning a VRF name and enters VRF configuration mode. The vrf-spoke argument is any case-sensitive, alphanumeric string up to 32 characters.

Step 6	**rd** *route-distinguisher*
	Example:
	`switch(config-vrf)# rd 1.101`
	`switch(config-vrf)#`
		Configures the route distinguisher. The route-distinguisher argument adds an 8-byte value to an IPv4 prefix to create a VPN IPv4 prefix. You can enter an RD in either of these formats:
		• 16-bit or 32-bit AS number: your 32-bit number, for example, 1.2:3
		• 32-bit IP address: your 16-bit number, for example, 192.0.2.1:1

	Command or Action	Purpose

Step 7	**address-family { ipv4 | ipv6 } unicast**
	Example:
	`switch(config-vrf)# address-family ipv4 unicast`
	`switch(config-vrf-af-ipv4)#`
		Specifies the IPv4 address family type and enters address family configuration mode.

Step 8	**route-target** { **import** | **export** } route-target-ext-community **}**
	Example:
	`switch(config-vrf-af-ipv4)# route-target import 1.0:1`
		Specifies a route-target extended community for a VRF as follows:
		• The **import** keyword imports routing information from the target VPN extended community.
		• The **export** keyword exports routing information to the target VPN extended community.
		• The route-target-ext-community argument adds the route-target extended community attributes to the VRF's list of import or export route-target extended communities. You can enter the route-target-ext-community argument in either of these formats:
		  • 16-bit or 32-bit AS number: your 32-bit number, for example, 1.2:3
		  • 32-bit IP address: your 16-bit number, for example, 192.0.2.1:1

Step 9	**show running-config vrf** *vrf-name* **Example:** `switch(config-vrf-af-ipv4)# show running-config vrf 2spokes`	(Optional) Displays the running configuration for the VRF. The vrf-name argument is any case-sensitive, alphanumeric string up to 32 characters. .
Step 10	**copy running-config startup-config** **Example:** `switch(config-router-vrf)# copy running-config startup-config`	(Optional) Copies the running configuration to the startup configuration.

Configuring eBGP on the Spoke PE Router

You can use eBGP to configure PE spoke routing sessions.

Note If all CE sites are using the same BGP AS number, you must perform the following tasks:

- Configure the allowas-in command at the perceiving spoke router.

Procedure

	Command or Action	Purpose

Step 1	**configure terminal**
	Example:
	`switch# configure terminal`
	`switch(config)#`
		Enters global configuration mode.

Step 2	**feature-set mpls** **Example:** `switch(config)# feature-set mpls`	Enables the MPLS feature-set.
Step 3	**feature mpls l3vpn** **Example:** `switch(config)# feature mpls l3vpn`	Enables the MPLS Layer 3 VPN feature.

Step 4	**feature bgp**
	Example:
	`switch(config)# feature bgp`
	`switch(config)#`
		Enables the BGP feature.

Step 5	**router bgp** *as - number*
	Example:
	`switch(config)# router bgp 100`
	`switch(config-router)#`
		Configures a BGP routing process and enters router configuration mode.
		The *as-number* argument indicates the number of an autonomous system that identifies the router to other BGP routers and tags the routing information passed along. The AS number can be a 16-bit integer or a 32-bit integer in the form of a higher 16-bit decimal number and a lower 16-bit decimal number in xx.xx format.

Step 6	**neighbor** *ip-address***remote-as** *as-number*
	Example:
	`switch(config-router)# neighbor 63.63.0.63 remote-as 100`
	`switch(config-router-neighbor)#`
		Adds an entry to the iBGP neighbor table.
		• The ip-address argument specifies the IP address of the neighbor in dotted decimal notation.
		• The as-number argument specifies the autonomous system to which the neighbor belongs.

	Command or Action	Purpose
Step 7	**address-family { ipv4 | ipv6 } unicast** **Example:** `switch(config-router-vrf-neighbor)# address-family ipv4 unicast switch(config-router-neighbor-af)#`	Specifies the IPv4 or IPv6 address family type and enters address family configuration mode.
Step 8	**allowas-in** *number* **Example:** `switch(config-router-vrf-neighbor-af)# allowas-in 3`	(Optional) Allows an AS path with the PE ASN for a specified number of times. • The range is from 1 to 10. • If all BGP sites are using the same AS number, configure the following commands: **Note** Configure the **BGP as-override** command at the PE (hub) or Configure the **allowas-in** command at the receiving CE router. The *as-number* argument indicates the number of an autonomous system that identifies the router to other BGP routers and tags the routing information passed along. The AS number can be a 16-bit integer or a 32-bit integer in the form of a higher 16-bit decimal number and a lower 16-bit decimal number in xx.xx format.
Step 9	**send-community extended** **Example:** `switch(config-router-neighbor)# send-community extended`	(Optional) Configures BGP to advertise extended community lists.
Step 10	**show running-config bgp** **Example:** `switch(config-router-vrf-neighbor-af)# show running-config bgp`	(Optional) Displays the running configuration for BGP.
Step 11	**copy running-config startup-config** **Example:** `switch(config-router-vrf)# copy running-config startup-config`	(Optional) Copies the running configuration to the startup configuration.

Configuring MPLS using Hardware Profile Command

Beginning with release 7.0(3)F3(3), Cisco Nexus 9508 switches with N9K-X9636C-R, N9K-X9636C-RX, and N9K-X9636Q-R line cards supports multiple hardware profiles. You can configure MPLS and/or VXLAN using hardware profile configuration command in a switch. The hardware profile configuration command invokes appropriate configuration files that are available on the switch. VXLAN is enabled by default

Before you begin

Procedure

	Command or Action	Purpose

Step 1	**configure terminal**
	Example:
	`switch# configure terminal`
	`switch(config)#`
		Enters global configuration mode.

Step 2	**feature bgp**
	Example:
	`switch(config)# feature bgp`
	`switch(config)#`
		Enables the BGP feature.

Step 3	**hardware profile** [**vxlan** | **mpls**] **module all** **Example:** `switch(config)# hardware profile mpls module all`	Enables MPLS on all the switch modules. .

Step 4	**show hardware profile module** [**all** | *number*]
	Example:
	`switch(config)# show hardware profile module all`
	`switch(config)#`
		Displays the hardware profile of all the modules or specific module.

Step 5	**show module internal sw info** | [**i** | **mpls**] **Example:** `switch(config)# show module internal sw info`	Displays the switch software information.
Step 6	**show running configuration** | [**i** | **mpls**] **Example:** `switch(config)# show module internal sw info`	Displays the running configuration.

CHAPTER 6

Configuring MPLS Layer 3 VPN Label Allocation

This chapter describes how to configure label allocation for Multiprotocol Label Switching (MPLS) Layer 3 virtual private networks (L3VPNs) on Cisco Nexus 9508 switches.

About MPLS Layer 3 VPN Label Allocation

The MPLS provider edge (PE) router stores both local and remote routes and includes a label entry for each route. By default, Cisco NX-OS uses per-prefix label allocation which means that each prefix is assigned a label. For distributed platforms, the per-prefix labels consume memory. When there are many VPN routing and forwarding instances (VRFs) and routes, the amount of memory that the per-prefix labels consume can become an issue.

You can enable per-VRF label allocation to advertise a single VPN label for local routes throughout the entire VRF. The router uses a new VPN label for the VRF decoding and IP-based lookup to learn where to forward packets for the PE or customer edge (CE) interfaces.

You can enable different label allocation modes for Border Gateway Protocol (BGP) Layer 3 VPN routes to meet different requirements and to achieve trade-offs between scalability and performance. All labels are allocated within the global label space. Cisco NX-OS supports the following label allocation modes:

- Per-prefix—A label is allocated for each VPN prefix. VPN packets received from remote PEs can be directly forwarded to the connected CE that advertised the prefix, based on the label forwarding table. However, this mode also uses many labels. This mode is the only mode available when VPN packets sent from PE to CE are label switched. This is the default label allocation mode.
- Per-VRF—A single label is assigned to all local VPN routes in a VRF. This mode requires an IPv4 or IPv6 lookup in the VRF forwarding table once the VPN label is removed at the egress PE. This mode is the most efficient in terms of label space as well as BGP advertisements, and the lookup does not result

in any performance degradation. Cisco NX-OS uses the same per-VRF label for both IPv4 and IPv6 prefixes.

Note EIBGP load balancing is not supported for a VRF that uses per-VRF label mode

- Aggregate Labels—BGP can allocate and advertise a local label for an aggregate prefix. Forwarding requires an IPv4 or IPv6 lookup that is similar to the per-VRF scenario. A single per-VRF label is allocated and used for all prefixes that need a lookup.
- VRF connected routes—When directly connected routes are redistributed and exported, an aggregate label is allocated for each route. The packets that come in from the core are decapsulated and a lookup is done in the VRF IPv4 or IPv6 table to determine whether the packet is for the local router or for another router or host that is directly connected. A single per-VRF label is allocated for all such routes.
- Label hold down—When a local label is no longer associated with a prefix, to allow time for updates to be sent to other PEs, the local label is not released immediately. A ten minute hold down timer is started per label. Within this hold down period, the label can be reclaimed for the prefix. When the timer expires, BGP releases the label.

IPv6 Label Allocation

IPv6 prefixes are advertised with the allocated label to iBGP peers that have the labeled-unicast address-family enabled. The received eBGP next hop is not propagated to such peers; instead, the local IPv4 session address is sent as an IPv4-mapped IPv6 next hop. The remote peer resolves this next hop through one or more IPv4 MPLS LSPs in the core network.

You can use a route reflector to advertise the labeled 6PE prefixes between PEs. You must enable the labeled-unicast address-family between the route reflector and all such peers. The route reflector does not need to be in the forwarding path and propagates the received next hop as is to iBGP peers and route reflector clients.

Note 6PE also supports both per-prefix and per-VRF label allocation modes, as in 6VPE

Per-VRF Label Allocation Mode

The following conditions apply when you configure per-VRF label allocation:

- The VRF uses one label for all local routes.
- When you enable per-VRF label allocation, any existing per-VRF aggregate label is used. If no per-VRF aggregate label is present, the software creates a new per-VRF label.

  The CE does not lose data when you disable per-VRF label allocation because the configuration reverts to the default per-prefix labeling configuration.
- A per-VRF label forwarding entry is deleted only if the VRF, BGP, or address family configuration is removed.

About Labeled and Unlabeled Unicast Paths

Subsequent Address Family Identifier (SAFI) is an indication of the BGP route. Example 1 is for an unlabeled route and 4 for a labeled route.

- Unlabeled unicast (U) for IPv4 is SAFI 1.
- Labeled unicast (LU) for IPv4 is SAFI 4.
- Unlabeled unicast (U) for IPv6 is AFI 2 and SAFI 1.
- Labeled unicast (LU) for IPv6 is AFI 2 and SAFI 4.

Cisco NX-OS Release 9.2(2) supports both, IPv4 and IPv6 unlabeled and labeled unicast on one BGP session. This behavior is the same irrespective of whether one or both SAFI-1 and SAFI-4 are enabled on the same session or not.

This behavior is applicable for all eBGP, iBGP, and redistributed paths and the eBGP and iBGP neighbors.

Prerequisites for MPLS Layer 3 VPN Label Allocation

Layer 3 VPN label allocation has the following prerequisites:

- Ensure that you have configured MPLS, and LDP or RSVP TE in your network. All routers in the core, including the PE routers, must be able to support MPLS forwarding.
- Ensure that you have installed the correct license for MPLS and any other features you will be using with MPLS.
- Ensure that you disable the external/internal Border Gateway Protocol (BGP) multipath feature if it is enabled before you configure per-VRF label allocation mode.
- Before configuring a 6VPE per VRF label, ensure that the IPv6 address family is configured on that VRF.

Guidelines and Limitations for MPLS Layer 3 VPN Label Allocation

Layer 3 VPN label allocation has the following configuration guidelines and limitations:

- Enabling per-VRF label allocation causes BGP reconvergence, which can result in data loss for traffic coming from the MPLS VPN core.

Note You can minimize network disruption by enabling per-VRF label allocation during a scheduled MPLS maintenance window. Also, if possible, avoid enabling this feature on a live router.

- Aggregate prefixes for per-prefix label allocation share the same label in a given VRF.

Default Settings for MPLS Layer 3 VPN Label Allocation

Table 3: Default Layer 3 VPN Label Allocation Parameters

Parameters	Default
Layer 3 VPN feature	Disabled
Label allocation mode	Per prefix

Configuring MPLS Layer 3 VPN Label Allocation

Configuring Per-VRF Layer 3 VPN Label Allocation Mode

You can configure per-VRF Layer 3 VPN label allocation mode for Layer 3 VPNs.

Procedure

	Command or Action	Purpose

Step 1	**configure terminal**
	Example:
	`switch# configure terminal`
	`switch(config)#`
		Enters global configuration mode.

Step 2	**feature bgp**
	Example:
	`switch(config)# feature bgp`
	`switch(config)#`
		Enables the BGP feature.

Step 3	**feature-set mpls**
	Example:
	`switch(config)# feature-set mpls`
	`switch(config)#`
		Enables the MPLS feature-set.

Step 4	**feature-set mpls l3vpn**
	Example:
	`switch(config)# feature-set mpls l3vpn`
	`switch(config)#`
		Enables the MPLS Layer 3 VPN feature.

Step 5	**router bgp** *as - number* **Example:** `switch(config)# router bgp 1.1`	Configures a BGP routing process and enters router configuration mode. The as-number argument indicates the number of an autonomous system that identifies the router to other BGP routers and tags the routing information. The AS number can be a 16-bit

	Command or Action	Purpose
		integer or a 32-bit integer in the form of a higher 16-bit decimal number and a lower 16-bit decimal number in xx.xx format.
Step 6	**vrf** *vrf-name* **Example:** `switch(config-router)# vrf vpn1`	Enters router VRF configuration mode. The vrf-name can be any case-sensitive, alphanumeric string up to 32 characters.
Step 7	**address-family { ipv4 | ipv6 } unicast | multicast }** **Example:** `switch(config-router-vrf)# address-family ipv6 unicast`	Specifies the IP address family type and enters address family configuration mode.
Step 8	**label-allocation-mode per-vrf** **Example:** `switch(config-router-vrf-af)# label-allocation-mode per-vrf`	Allocates labels on a per-VRF basis.
Step 9	**show bgp l3vpn detail vrf** *vrf-name* **Example:** `switch(config-router-vrf-af)# show bgp l3vpn detail vrf vpn1`	(Optional) Displays information about Layer 3 VPN configuration on BGP for this VRF. The vrf-name can be any case-sensitive, alphanumeric string up to 32 characters.
Step 10	**copy running-config startup-config** **Example:** `switch(config-router-vrf)# copy running-config startup-config`	(Optional) Copies the running configuration to the startup configuration.

Allocating Labels for IPv6 Prefixes in the Default VRF

If you are running IPv6 over an IPv4 MPLS core network (6PE), you can allocate labels for the IPv6 prefixes in the default VRF.

Note By default, labels are not allocated for IPv6 prefixes in the default VRF.

Procedure

	Command or Action	Purpose

Step 1	**configure terminal**
	Example:
	`switch# configure terminal`
	`switch(config)#`
		Enters global configuration mode.

	Command or Action	Purpose

Step 2	**feature bgp**
	Example:
	`switch(config)# feature bgp`
	`switch(config)#`
		Enables the BGP feature.

Step 3	**feature-set mpls**
	Example:
	`switch(config)# feature-set mpls`
	`switch(config)#`
		Enables the MPLS feature-set.

Step 4	**feature-set mpls l3vpn**
	Example:
	`switch(config)# feature-set mpls l3vpn`
	`switch(config)#`
		Enables the MPLS Layer 3 VPN feature.

Step 5	**router bgp** *as - number* **Example:** `switch(config)# router bgp 1.1`	Configures a BGP routing process and enters router configuration mode. The as-number argument indicates the number of an autonomous system that identifies the router to other BGP routers and tags the routing information. The AS number can be a 16-bit integer or a 32-bit integer in the form of a higher 16-bit decimal number and a lower 16-bit decimal number in xx.xx format.
Step 6	**address-family { ipv4 | ipv6 } unicast | multicast }** **Example:** `switch(config-router-vrf)# address-family ipv6 unicast`	Specifies the IP address family type and enters address family configuration mode.
Step 7	**allocate-label { all | route-map** *route-map* **}** **Example:** `switch(config-router-af)# allocate-label all`	Allocates labels for IPv6 prefixes in the default VRF. • The **all** keyword allocates labels for all IPv6 prefixes. • The **route-map** keyword allocates labels for IPv6 prefixes matched in the specified route map. The route-map can be any case-sensitive alphanumeric string up to 63 characters.
Step 8	**show running-config bgp** **Example:** `switch(config-router-af)# show running-config bgp`	(Optional) Displays information about the BGP configuration.

	Command or Action	Purpose
Step 9	**copy running-config startup-config** **Example:** `switch(config-router-vrf)# copy running-config startup-config`	(Optional) Copies the running configuration to the startup configuration.

Enabling Sending MPLS Labels in IPv6 over an IPv4 MPLS Core Network (6PE) for iBGP Neighbors

6PE advertises IPv6 prefixes in global VRF over IPv4 based MPLS network with the allocated label to iBGP peers that have the labeled-unicast address-family enabled. PE requires LDP enabled on core facing interfaces to transport IPv6 traffic over IPv4 based MPLS network and “address-family ipv6 labeled-unicast” under BGP to exchange label for IPv6 prefixes between PEs.

Note The **address-family ipv6 labeled-unicast** command is supported only for iBGP neighbors. You cannot use this command with the **address-family ipv6 unicast** command.

Procedure

	Command or Action	Purpose

Step 1	**configure terminal**
	Example:
	`switch# configure terminal`
	`switch(config)#`
		Enters global configuration mode.

Step 2	**feature bgp**
	Example:
	`switch(config)# feature bgp`
	`switch(config)#`
		Enables the BGP feature.

Step 3	**feature-set mpls**
	Example:
	`switch(config)# feature-set mpls`
	`switch(config)#`
		Enables the MPLS feature-set.

Step 4	**feature-set mpls l3vpn**
	Example:
	`switch(config)# feature-set mpls l3vpn`
	`switch(config)#`
		Enables the MPLS Layer 3 VPN feature.

Step 5	**router bgp** *as - number* **Example:** `switch(config)# router bgp 1.1`	Configures a BGP routing process and enters router configuration mode. The as-number argument indicates the number of an autonomous system that identifies the router to

	Command or Action	Purpose
		other BGP routers and tags the routing information. The AS number can be a 16-bit integer or a 32-bit integer in the form of a higher 16-bit decimal number and a lower 16-bit decimal number in xx.xx format.

Step 6	**neighbor** *ip-address*
	Example:
	`switch(config-router)# neighbor 209.165.201.1`
	`switch(config-router-neighbor)#`
		Adds an entry to the BGP or multiprotocol BGP neighbor table. The ip-address argument specifies the IP address of the neighbor in dotted decimal notation.

Step 7	**address-family ipv6 labeled-unicast**
	Example:
	`switch(config-router-neighbor)# address-family ipv6 labeled-unicast`
	`switch(config-router-neighbor-af)#`
		Specifies IPv6 labeled unicast address prefixes. This command is accepted only for iBGP neighbors.

Step 8	**show running-config bgp** **Example:** `switch(config-router-af)# show running-config bgp`	(Optional) Displays information about the BGP configuration.
Step 9	**copy running-config startup-config** **Example:** `switch(config-router-vrf)# copy running-config startup-config`	(Optional) Copies the running configuration to the startup configuration.

Advertisement and Withdraw Rules

The following table shows the advertisement and withdraw behavior for different scenarios.

Table 4: Advertisement and Withdraw Rules

Case	Bestpath/ Addpath Type	Local Label Present?	NHS or NHU	Update-group SAFI	Advertise o withdraw?
1	Unlabeled path. For example, no RX label.	Yes	NHS	SAFI-1	Advertise b default.
2				SAFI-4	Advertise
3			NHU	SAFI-1	Advertise
4				SAFI-4	Withdraw
5		No	NHS	SAFI-1	Advertise
6				SAFI-4	Withdraw
7			NHU	SAFI-1	Advertise
8				SAFI-4	Withdraw
9	Labeled path. For example, with an RX label.	Yes	NHS	SAFI-1	Advertise b default. Withdraw w NbrKnob.
10				SAFI-4	Advertise

Case	Bestpath/ Addpath Type	Local Label Present?	NHS or NHU	Update-group SAFI	Advertise or withdraw?
11			NHU	SAFI-1	Withdraw
12				SAFI-4	Advertise
13		No	NHS	SAFI-1	Advertise
14				SAFI-4	Withdraw
15			NHU	SAFI-1	Withdraw
				SAFI-4	Advertise

Enabling Local Label Allocation

Procedure

	Command or Action	Purpose

Step 1	**configure terminal**
	Example:
	`switch# configure terminal`
	`switch(config)#`
		Enters global configuration mode.

Step 2	**feature bgp**
	Example:
	`switch(config)# feature bgp`
	`switch(config)#`
		Enables the BGP feature.

	Command or Action	Purpose

Step 3	**feature-set mpls**
	Example:
	`switch(config)# feature-set mpls`
	`switch(config)#`
		Enables the MPLS feature-set.

Step 4	**router bgp** *as - number* **Example:** `switch(config)# router bgp 1.1`	Configures a BGP routing process and enters router configuration mode. The as-number argument indicates the number of an autonomous system that identifies the router to other BGP routers and tags the routing information. The AS number can be a 16-bit integer or a 32-bit integer in the form of a higher 16-bit decimal number and a lower 16-bit decimal number in xx.xx format.
Step 5	**address-family { ipv4 | ipv6 } unicast | multicast }** **Example:** `switch(config-router-vrf)# address-family ipv4 unicast`	Specifies the IP address family type and enters the address family configuration mode.
Step 6	**allocate-label { all | route-map** *route-map* **}** **Example:** `switch(config-router-af)# allocate-label all`	Allocates labels for IPv6 prefixes in the default VRF. • The **all** keyword allocates labels for all IPv6 prefixes. • The **route-map** keyword allocates labels for IPv6 prefixes matched in the specified route map. The route-map can be any case-sensitive alphanumeric string up to 63 characters.

Step 7	**neighbor** *ip-address*
	Example:
	`switch(config-router)# neighbor 209.165.201.1`
	
	`switch(config-router-neighbor)#`
		Adds an entry to the BGP or multiprotocol BGP neighbor table. The ip-address argument specifies the IP address of the neighbor in dotted decimal notation.

Step 8	[**no**] **advertise local-labeled-route** **Example:** `switch(config-router-neighbor)# advertise local-labeled-route`	Indicates whether to advertise an IPv4 or IPv6 route with a local label to the BGP neighbor via the IPv4 or IPv6 unicast SAFI (SAFI-1). The default is enabled so that it can be advertised to the BGP neighbor.
Step 9	**address-family { ipv4 | ipv6 } unicast | multicast }** **Example:**	Specifies the IP address family type and enters the address family configuration mode.

	Command or Action	Purpose
	`switch(config-router-vrf)# address-family ipv6 unicast`	
Step 10	[**no**] **advertise local-labeled-route** **Example:** `switch(config-router-neighbor)# advertise local-labeled-route`	Indicates whether to advertise an IPv4 or IPv6 route with a local label to the BGP neighbor via the IPv4 or IPv6 unicast SAFI (SAFI-1). The default is enabled so that it can be advertised to the BGP neighbor.
Step 11	**route-map label_routemap permit 10** **Example:** `switch(config-router-vrf)# route-map label_routemap permit 10`	
Step 12	**show running-config bgp** **Example:** `switch(config-router-af)# show running-config bgp`	(Optional) Displays information about the BGP configuration.
Step 13	**copy running-config startup-config** **Example:** `switch(config-router-vrf)# copy running-config startup-config`	(Optional) Copies the running configuration to the startup configuration.

Verifying MPLS Layer 3 VPN Label Allocation Configuration

To display the Layer 3 VPN label allocation configuration, perform one of the following tasks:

Table 5: Verifying MPLS Layer 3 VPN Label Allocation Configuration

Command	Purpose
show bgp l3vpn [**detail**] [**vrf** *v rf-name*]	Displays Layer 3 VPN information for BGP in a VRF.
show bgp vpnv4 unicast labels [**vrf** *v rf-name*]	Displays label information for BGP.
show ip route [**vrf** *v rf-name*]	Displays label information for routes.

Configuration Examples for MPLS Layer 3 VPN Label Allocation

The following example shows how to configure per-VRF label allocation for an IPv4 MPLS network.

```
PE1
-----
vrf context vpn1
rd 100:1
address-family ipv4 unicast
route-target export 200:1
```

```
router bgp 100
neighbor 10.1.1.2 remote-as 100
address-family vpnv4 unicast
send-community extended
update-source loopback10
vrf vpn1
address-family ipv4 unicast
label-allocation-mode per-vrf
neighbor 36.0.0.2 remote-as 300
address-family ipv4 unicast
```

CHAPTER 7

Configuring MPLS Layer 3 VPN Load Balancing

This chapter describes how to configure load balancing for Multiprotocol Label Switching (MPLS) Layer 3 virtual private networks (VPNs) on Cisco Nexus 9508 switches.

Information About MPLS Layer 3 VPN Load Balancing

Load balancing distributes traffic so that no individual router is overburdened. In an MPLS Layer 3 network, you can achieve load balancing by using the Border Gateway Protocol (BGP). When multiple iBGP paths are installed in a routing table, a route reflector advertises only one path (next hop). If a router is behind a route reflector, all routes that are connected to multihomed sites are not advertised unless a different route distinguisher is configured for each virtual routing and forwarding instance (VRF). (A route reflector passes learned routes to neighbors so that all iBGP peers do not need to be fully meshed.)

iBGP Load Balancing

When a BGP-speaking router configured with no local policy receives multiple network layer reachability information (NLRI) from the internal BGP (iBGP) for the same destination, the router chooses one iBGP path as the best path and installs the best path in its IP routing table. iBGP load balancing enables the BGP-speaking router to select multiple iBGP paths as the best paths to a destination and to install multiple best paths in its IP routing table.

eBGP Load Balancing

When a router learns two identical eBGP paths for a prefix from a neighboring autonomous system, it chooses the path with the lower route ID as the best path. The router installs this best path in the IP routing table. You can enable eBGP load balancing to install multiple paths in the IP routing table when the eBGP paths are learned from a neighboring autonomous system instead of picking one best path.

During packet switching, depending on the switching mode, the router performs either per-packet or per-destination load balancing among the multiple paths.

Layer 3 VPN Load Balancing

Layer 3 VPN load balancing for both eBGP and iBGP allows you to configure multihomed autonomous systems and provider edge (PE) routers to distribute traffic across both external BGP (eBGP) and iBGP multipaths.

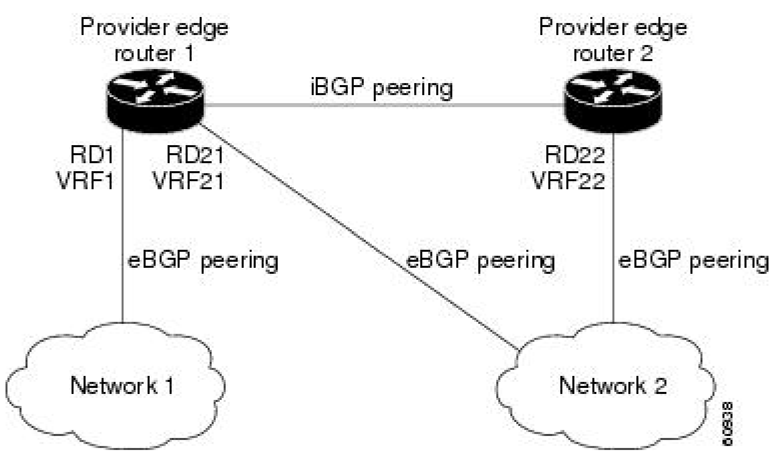
Layer 3 VPN load balancing supports IPv4 and IPv6 for the PE routers and VPNs.

BGP installs up to the maximum number of multipaths allowed. BGP uses the best path algorithm to select one path as the best path, inserts the best path into the routing information base (RIB) and advertises the best path to BGP peers. The router can insert other paths into the RIB but selects only one path as the best path.

Layer 3 VPNs load balance on a per-packet or per-source or destination pair basis. To enable load balancing, configure the router with Layer 3 VPNs that contain VPN routing and forwarding instances (VRFs) that import both eBGP and iBGP paths. You can configure the number of paths separately for each VRF.

The following figure shows an MPLS provider network that uses BGP. In the figure, two remote networks are connected to PE1 and PE2, which are both configured for VPN unicast iBGP peering. Network 2 is a multihomed network that is connected to PE1 and PE2. Network 2 also has extranet VPN services configured with Network 1. Both Network 1 and Network 2 are configured for eBGP peering with the PE routers.

Figure 4: Provider MPLS Network Using BGP



You can configure PE1 so that it can select both iBGP and eBGP paths as multipaths and import these paths into the VPN routing and forwarding instance (VRF) of Network 1 to perform load balancing.

Traffic is distributed as follows:

- IP traffic that is sent from Network 2 to PE1 and PE2 is sent across the eBGP paths as IP traffic.
- IP traffic that is sent from PE1 to PE2 is sent across the iBGP path as MPLS traffic.
- Traffic that is sent across an eBGP path is sent as IP traffic.

Any prefix that is advertised from Network 2 will be received by PE1 through route distinguisher (RD) 21 and RD22.

- The advertisement through RD21 is carried in IP packets.

- The advertisement through RD22 is carried in MPLS packets.

The router can select both paths as multipaths for VRF1 and insert these paths into the VRF1 RIB.

Layer 3 VPN Load Balancing with Route Reflectors

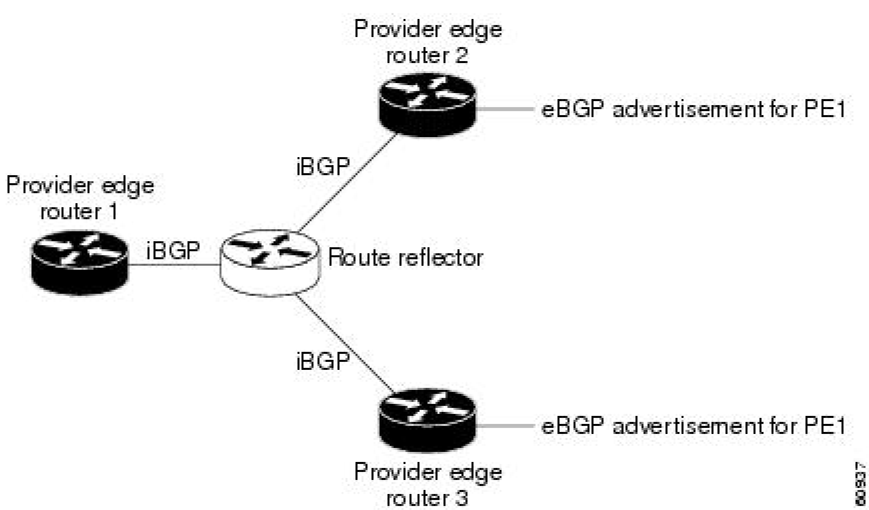
Route reflectors reduce the number of sessions on PE routers and increase the scalability of Layer 3 VPN networks. Route reflectors hold on to all received VPN routes to peer with PE routers. Different PEs can require different route target-tagged VPNv4 and VPNv6 routes. The route reflector may also need to send a refresh for a specific route target to a PE when the VRF configuration has changed. Storing all routes increases the scalability requirements on a route reflector. You can configure a route reflector to only hold routes that have a defined set of route target communities.

You can configure route reflectors to service a different set of VPNs and configure a PE to peer with all route reflectors that service the VRFs configured on the PE. When you configure a new VRF with a route target that the PE does not already hold routes for, the PE issues route refreshes to the route reflectors and retrieves the relevant VPN routes.

The following figure shows a topology that contains three PE routers and a route reflector, all configured for iBGP peering. PE2 and PE3 each advertise an equal preference eBGP path to PE1. By default, the route reflector chooses only one path and advertises PE1.

Note The route reflectors do not need to be in the forwarding path, but you must configure unique route distinguisher (RDs) for VPN sites that are multihomed.

Figure 5: Topology with a Route Reflector



For all equal preference paths to PE1 to be advertised through the route reflector, you must configure each VRF with a different RD. The prefixes received by the route reflector are recognized differently and advertised to PE1.

Layer 2 Load Balancing Coexistence

The load balance method that is required in the Layer 2 VPN is different from the method that is used for Layer 3 VPN. Layer 3 VPN and Layer 2 VPN forwarding is performed independently using two different types of adjacencies. The forwarding is not impacted by using a different method of load balancing for the Layer 2 VPN.

Note Load balancing is not supported at the ingress PE for Layer 2 VPNs

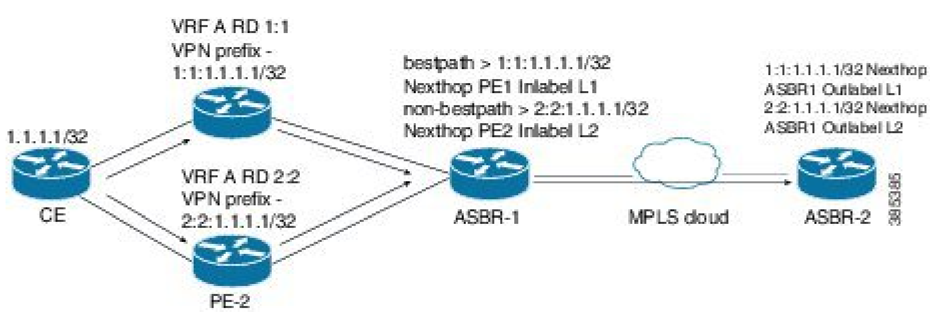
BGP VPNv4 Multipath

BGP VPNv4 Multipath feature helps to achieve Equal Cost Multi-Path (ECMP) for traffic flowing from an Autonomous System Border Router (ASBR) towards the Provider Edge (PE) device in an Multi-Protocol Label Switching (MPLS) cloud network by using a lower number of prefixes and MPLS labels. This feature configures the maximum number of multipaths for both eBGP and iBGP paths. This feature can be configured on PE devices and Route Reflectors in an MPLS topology.

Consider a scenario in which a dual homed Customer Edge (CE) device is connected to 2 PE devices and you have to utilize both the PE devices for traffic flow from ASBR-2 to the CE device.

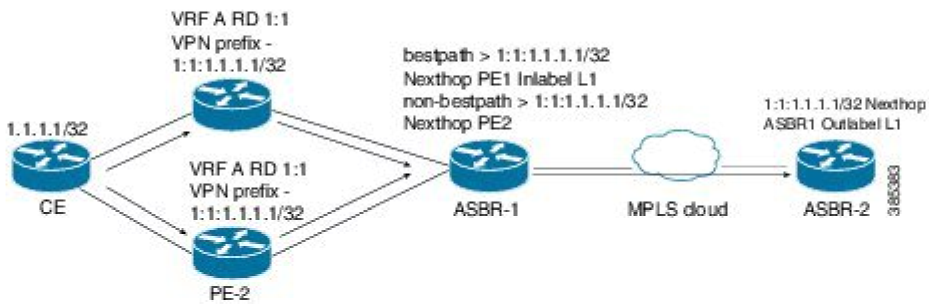
Currently, as shown in following figure, Virtual Routing and Forwarding (VRF) on each PE is configured using separate Route Distinguishers (RD). The CE device generates a BGP IPv4 prefix. The PE devices are configured with 2 separate RDs and generate two different VPN-IPv4 prefixes for the BGP IPv4 prefix sent by the CE device. ASBR-1 receives both the VPN-IPv4 prefixes and adds them to the routing table. ASBR-1 allocates Inter-AS option-B labels, Inlabel L1 and Inlabel L2, to both the VPN routes and then advertises both VPN routes to ASBR-2. To use both PE devices to maintain traffic flow, ASBR-1 has to utilize two Inter-AS option-B labels and two prefixes which limits the scale that can be supported.

Figure 6: Virtual Routing and Forwarding (VRF) on each PE configured using separate Route Distinguishers



Using the BGP VPN Multipath feature, as shown in Figure 22-4, you can enable the VRF on both PE devices to use the same RD. In such a scenario, ASBR-1 receives the same prefix from both the PE devices. ASBR-1 allocates only one Inter-AS option-B label, Inlabel L1, to the received prefix and advertises the VPN route to ASBR-2. In this case, the scale is enhanced as traffic flow using both PE devices is established with only one prefix and label on ASBR-1.

Figure 7: Enabling the VRF on both PE devices to use the same RD



BGP Cost Community

The BGP cost community is a nontransitive extended community attribute that is passed to iBGP and confederation peers but not to eBGP peers. (A confederation is a group of iBGP peers that use the same autonomous system number to communicate to external networks.) The BGP cost community attributes includes a cost community ID and a cost value. You can customize the BGP best path selection process for a local autonomous system or confederation by configuring the BGP cost community attribute. You configure the cost community attribute in a route map with a community ID and cost value. BGP prefers the path with the lowest community ID, or for identical community IDs, BGP prefers the path with the lowest cost value in the BGP cost community attribute.

BGP uses the best path selection process to determine which path is the best where multiple paths to the same destination are available. You can assign a preference to a specific path when multiple equal cost paths are available.

Since the administrative distance of iBGP is worse than the distance of most Interior Gateway Protocols (IGPs), the unicast Routing Information Base (RIB) may apply the same BGP cost community compare algorithm before using the normal distance or metric comparisons of the protocol or route. VPN routes that are learned through iBGP can be preferred over locally learned IGP routes.

The cost extended community attribute is propagated to iBGP peers when an extended community exchange is enabled.

How the BGP Cost Community Influences the Best Path Selection Process

The cost community attribute influences the BGP best path selection process at the point of insertion (POI). The POI follows the IGP metric comparison. When BGP receives multiple paths to the same destination, it uses the best path selection process to determine which path is the best path. BGP automatically makes the decision and installs the best path into the routing table. The POI allows you to assign a preference to a specific path when multiple equal cost paths are available. If the POI is not valid for local best path selection, the cost community attribute is silently ignored.

You can configure multiple paths with the cost community attribute for the same POI. The path with the lowest cost community ID is considered first. All of the cost community paths for a specific POI are considered, starting with the one with the lowest cost community ID. Paths that do not contain the cost community (for the POI and community ID being evaluated) are assigned with the default community cost value.

Applying the cost community attribute at the POI allows you to assign a value to a path originated or learned by a peer in any part of the local autonomous system or confederation. The router can use the cost community as a tie breaker during the best path selection process. You can configure multiple instances of the cost community for separate equal cost paths within the same autonomous system or confederation. For example, you can apply a lower cost community value to a specific exit path in a network with multiple equal cost exits points, and the BGP best path selection process prefers that specific exit path.

Cost Community and EIGRP PE-CE with Back-Door Links

BGP prefers back-door links in an Enhanced Interior Gateway Protocol (EIGRP) Layer 3 VPN topology if the back-door link is learned first. A back-door link, or a route, is a connection that is configured outside of the Layer 3 VPN between a remote and main site.

The pre-best path point of insertion (POI) in the BGP cost community supports mixed EIGRP Layer 3 VPN network topologies that contain VPN and back-door links. This POI is applied automatically to EIGRP routes that are redistributed into BGP. The pre-best path POI carries the EIGRP route type and metric. This POI influences the best-path calculation process by influencing BGP to consider this POI before any other comparison step.

Prerequisites for MPLS Layer 3 VPN Load Balancing

MPLS Layer 3 VPN load balancing has the following prerequisites:

- You must enable the MPLS and L3VPN features.
- You must install the correct license for MPLS.

Guidelines and Limitations for MPLS Layer 3 VPN Load Balancing

MPLS Layer 3 VPN load balancing has the following configuration guidelines and limitations:

- You can configure MPLS Layer 3 VPN load balancing for Cisco Nexus 9508 platform switches with the N9K-X9636C-R, N9K-X9636C-RX, and N9K-X9636Q-R line cards.
- Beginning with Cisco NX-OS Release 9.3(3), you can configure MPLS Layer 3 VPN load balancing on Cisco Nexus 9364C-GX, Cisco Nexus 9316D-GX, and Cisco Nexus 93600CD-GX switches.
- If you place a router behind a route reflector and it is connected to multihomed sites, the router will not be advertised unless separate VRFs with different RDs are configured for each VRF.
- Each IP routing table entry for a BGP prefix that has multiple iBGP paths uses additional memory. We recommend that you do not use this feature on a router with a low amount of available memory or when it is carrying a full Internet routing table.
- You should not ignore the BGP cost community when a back-door link is present and EIGRP is the PE-CE routing protocol.
- A maximum of 16K VPN prefixes is supported on Cisco Nexus 9508 platform switches with N9K-X9636Q-R and N9K-X9636C-R line cards, and a maximum of 470K VPN prefixes is supported on Cisco Nexus 9508 platform switches with N9K-X9636C-RX line cards.

- 4K VRFs are supported.

Default Settings for MPLS Layer 3 VPN Load Balancing

The following table lists the default settings for MPLS Layer 3 VPN load balancing parameters.

Table 6: Default MPLS Layer 3 VPN Load Balancing Parameters

Parameters	Default
Layer 3 VPN feature	Disabled
BGP cost community ID	128
BGP cost community cost	2147483647
maximum multipaths	1
BGP VPNv4 Multipath	Disabled

Configuring MPLS Layer 3 VPN Load Balancing

Configuring BGP Load Balancing for eBGP and iBGP

You can configure a Layer 3 VPN load balancing for an eBGP or iBGP network.

Procedure

	Command or Action	Purpose

Step 1	**configure terminal**
	Example:
	`switch# configure terminal`
	`switch(config)#`
		Enters global configuration mode.

Step 2	**feature-set mpls** **Example:** `switch(config)# feature-set mpls`	Enables the MPLS feature-set.
Step 3	**feature mpls l3vpn** **Example:** `switch(config)# feature mpls l3vpn`	Enables the MPLS Layer 3 VPN feature.
Step 4	**feature bgp** **Example:**	Enables the BGP feature.

	Command or Action	Purpose

	`switch(config)# feature bgp`
	
	`switch(config)#`	

Step 5	**router bgp** *as - number*
	
	Example:
	`switch(config)# router bgp 1.1`
	
	`switch(config-router)#`
		Configures a BGP routing process and enters router configuration mode.
		
		The *as-number* argument indicates the number of an autonomous system that identifies the router to other BGP routers and tags the routing information passed along. The AS number can be a 16-bit integer or a 32-bit integer in the form of a higher 16-bit decimal number and a lower 16-bit decimal number in xx.xx format.

Step 6	**bestpath cost-community ignore remote-as** *as-number* **Example:** `switch(config-router)# bestpath cost-community ignore#`	(Optional) Ignores the cost community for BGP bestpath calculations.

Step 7	**address-family { ipv4 | ipv6 } unicast**
	
	Example:
	`switch(config-router)# address-family ipv4 unicast`
	
	`switch(config-router-af)#`
		Enters address family configuration mode for configuring IP routing sessions.

Step 8	**maximum-paths** [**bgp**] *number-of-paths* **Example:** `switch(config-router-af)# maximum-paths 4`	Configures the maximum number of multipaths allowed. Use the ibgp keyword to configure **iBGP** load balancing. The range is from 1 to 16.
Step 9	**show running-config bgp** **Example:** `switch(config-router-vrf-neighbor-af)# show running-config bgp`	(Optional) Displays the running configuration for BGP.
Step 10	**copy running-config startup-config** **Example:** `switch(config-router-vrf)# copy running-config startup-config`	(Optional) Copies the running configuration to the startup configuration.

Configuring BGPv4 Multipath

Procedure

	Command or Action	Purpose

Step 1	**configure terminal**
	Example:
	`switch# configure terminal`
	`switch(config)#`
		Enters global configuration mode.

Step 2	**feature bgp** **Example:** `switch(config)# feature bgp`	Enables the BGP feature.

Step 3	**router bgp** *as - number*
	Example:
	`switch(config)# router bgp 2`
	`switch(config-router)#`
		Assigns an autonomous system (AS) number to a router and enter the router BGP configuration mode.

Step 4	**address-family vpnv4 unicast**
	Example:
	`switch(config-router)# address-family vpnv4 unicast`
	`switch(config-router-af)#`
		Enters address family configuration mode for configuring routing sessions, such as BGP, that use standard VPNv4 address prefixes.

Step 5	**maximum-paths eibgp** *parallel-paths* **Example:** `switch(config-router-af)# maximum-paths eibgp 3`	Specifies the maximum number of BGP VPNv4 multipaths for both eBGP and iBGP paths. The range is from 1 to 32.

Configuring MPLS ECMP Load Sharing

Beginning Cisco NX-OS Release 9.3(1), you can configure MPLS ECMP load sharing based on labels. This feature is supported on Cisco Nexus 9200, Cisco Nexus 9300-EX, Cisco Nexus 9300-FX, and Cisco Nexus 9500 platform switches with Cisco Nexus N9K-X9700-EX and N9K-X9700-FX line cards.

Beginning with Cisco NX-OS Release 9.3(3), this feature is supported on Cisco Nexus 9364C-GX, Cisco Nexus 9316D-GX, and Cisco Nexus 93600CD-GX switches.

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:**	Enters global configuration mode.

	Command or Action	Purpose
	`switch# configure terminal` `switch(config)#`	
Step 2	**feature-set mpls** **Example:** `switch(config)# feature-set mpls`	Enables the MPLS feature-set.

Step 3	**mpls load-sharing** [**label-only** | [**label-ip]**
	Example:
	`switch(config)# mpls load-sharing label-only`
	`switch(config)# mpls load-sharing label-ip`
		Configures the load sharing based on the mpls labels. The label-only option configures the load sharing based on the labels, while the label-ip option configures it based on the label and the IP address.

Step 4	**copy running-config startup-config** **Example:** `switch(config)# copy running-config startup-config`	(Optional) Copies the running configuration to the startup configuration.

Verifying MPLS ECMP Load Sharing

To display the mpls ECMP load sharing configuration, perform one of the following tasks:

Table 7: Verifying MPLS ECMP Load Sharing Configuration

Command	Purpose
show mpls load-sharing	Displays the number of labels that are used for the mpls hashing and the IP fields that are used for the hashing.

Configuration Examples for MPLS Layer 3 VPN Load Balancing

Example: MPLS Layer 3 VPN Load Balancing

The following example shows how to configure iBGP load balancing:

```
 configure terminal
feature-set mpls
feature mpls l3vpn
feature bgp
router bgp 1.1
bestpath cost-community ignore
address-family ipv6 unicast
maximum-paths ibgp 4
```

Example: BGP VPNv4 Multipath

The following example shows how to configure a maximum of 3 BGP VPNv4 multipaths:

```
 configure terminal
router bgp 100
address-family vpnv4 unicast
maximum-paths eibgp 3
```

Example: MPLS Layer 3 VPN Cost Community

The following example shows how to configure the BGP cost community:

```
 configure terminal
feature-set mpls
feature mpls l3vpn
feature bgp
route-map CostMap permit
set extcommunity cost 1 100
router bgp 1.1
router-id 192.0.2.255
neighbor 192.0.2.1 remote-as 1.1
address-family vpnv4 unicast
send-community extended
route-map CostMap in
```

CHAPTER 8

Configuring MPLS QoS

This chapter describes how to configure Quality of Service for Multiprotocol Label Switching (MPLS) Layer 3 virtual private networks (VPNs).

About MPLS Quality of Service (QoS)

MPLS QoS enables you to provide differentiated types of service across an MPLS network. Differentiated types of service satisfy a range of requirements by supplying the service specified for each packet. QoS allows you to classify the network traffic, police and prioritize the traffic flow, and provide congestion avoidance.

This section includes the following topics:

- MPLS QoS Terminology, on page 97
- MPLS QoS Features, on page 98

MPLS QoS Terminology

This section defines some MPLS QoS terminology:

- Classification is the process that selects the traffic to be marked. Classification matches traffic with the selection criteria into multiple priority levels or classes of service. Traffic classification is the primary component of class-based QoS provisioning. The switch makes classification decisions based on the EXP bits in the topmost label of the received MPLS packets (after a policy is installed).
- Differentiated Services Code Point (DSCP):
  - Is the first six bits of the ToS byte in the IP header.
  - Only present in an IP packet.
  - Can be present in an IPv4 or an IPv6 packet.
  - Is the first 6 bits of the 8-bit Traffic Class octet in the IPv6 header.

- E-LSP is a label switched path (LSP) on which nodes infer the QoS treatment for MPLS packets exclusively from the experimental (EXP) bits in the MPLS header. Because the QoS treatment is inferred from the EXP (both class and drop precedence), several classes of traffic can be multiplexed onto a single LSP (use the same label). A single LSP can support up to eight classes of traffic because the EXP field is a 3-bit field.
- EXP bits define the QoS treatment (per-hop behavior) that a node should give to a packet. It is the equivalent of the DiffServ Code Point (DSCP) in the IP network. A DSCP defines a class and drop precedence. The EXP bits are generally used to carry all the information encoded in the IP DSCP. In some cases, however, the EXP bits are used exclusively to encode the dropping precedence.
- Marking is the process of setting a Layer 3 DSCP value in a packet. Marking is also the process of choosing different values for the MPLS EXP field to mark packets so that they have the priority that they require during periods of congestion.
- MPLS Experimental Field: Setting the MPLS experimental (EXP) field value satisfies the requirement of operators who do not want the value of the IP precedence field modified within IP packets transported through their networks. By choosing different values for the MPLS EXP field, you can mark packets so that packets have the priority that they require during periods of congestion. By default, the three most significant bits of the DSCP are copied into the MPLS EXP field during imposition. You can mark the MPLS EXP bits with an MPLS QoS policy.

MPLS QoS Features

QoS enables a network to provide improved service to selected network traffic. This section explains the following MPLS QoS features, which are supported in an MPLS network:

MPLS Experimental Field

Setting the MPLS experimental (EXP) field value satisfies the requirement of service providers who do not want the value of the IP precedence field modified within IP packets transported through their networks.

By choosing different values for the MPLS EXP field, you can mark packets so that packets have the priority that they require during periods of congestion.

By default, the IP precedence value is copied into the MPLS EXP field during imposition. You can mark the MPLS EXP bits with an MPLS QoS policy.

Classification

Classification is the process that selects the traffic to be marked. Classification accomplishes this by partitioning traffic into multiple priority levels, or classes of service. Traffic classification is the primary component of class-based QoS provisioning.

Policing and Marking

Policing causes traffic that exceeds the configured rate to be discarded or marked down to a higher drop precedence. Marking is a way to identify packet flows to differentiate them. Packet marking allows you to partition your network into multiple priority levels or classes of service.

The MPLS QoS policing and marking features that you can implement depend on the received traffic type and the forwarding operation applied to the traffic.

Guidelines and Limitations for MPLS QoS

MPLS Quality of Service (QoS) has the following configuration guidelines and limitations:

- When setting the QoS policy, the **topmost** keyword in the **set mpls experimental imposition** CLI is not supported.
- MPLS QoS does not support marking based on policing.
- L3 EVPN egress node - policing is not supported on a system level mpls-in-policy.
- Egress QoS classification that is based on MPLS EXP is not supported.
- EXP labels are only set for newly pushed or swapped labels. The EXP in the inner labels remains unchanged.
- When the traffic from the ingress line card takes the fabric module path to the line card, the line cards acting as the MPLS Ingress LSR node do not support ECN marking. This occurs for the Cisco Nexus 9500 platform switches with the N9K-X9700-EX and N9K-X9700-FX line cards.
- On the Label Edge Router (LER), policy match on EXP is not supported. Inner DSCP can be used to match the packets.
- Interface policy cannot be used to classify MPLS L3 EVPN packets on the Egress Label Edge Router (LER).System level MPLS-Default policy is used to classify the traffic.
- Explicit Congestion Notification (ECN) Marking is not supported on the label switching router transit node.
- Only the default QoS Service template is supported for the MPLS handoff in Cisco NX-OS Release 9.3(1). You cannot set the EXP labels on the MPLS.
- Beginning with Cisco NX-OS Release 9.3(5), MPLS QoS is supported on Cisco Nexus 9364C-GX, Cisco Nexus 9316D-GX, and Cisco Nexus 93600CD-GX switches.
- PFC is not supported for MPLS QoS and VXLAN MPLS DCI.
- Even after removing the queuing policy from an interface, previous micro-burst statistics remain. Use the clear queuing burst-detect command to clear the remaining records.
- RACL on an ingress port of egress PE (sr decap) is not supported.
- In order to write an EXP value in the label, an explicit poicy is necessary on the PE. In absence of a policy, the default EXP value is 7.

Configuring MPLS QoS

Note Be aware that the Cisco NX-OS commands for this feature may differ from those commands used in Cisco IOS.

Configuring MPLS Ingress Label Switched Router

To configure MPLS Ingress label switched router, perform the following:

MPLS Ingress LSR Classification

To match the value of the Differentiated Services Code Point (DSCP) field, use the **match dscp** command in QoS policy-map class configuration mode. To disable the setting, use the **no** form of this command.

Note Default entries are programmed to match on DSCP and mark EXP when no ingress QoS policy is configured (Uniform mode behavior at encap).

Before you begin

- You must enable MPLS configuration.
- Ensure that you are in the correct VDC (or use the switch to vdc command).

Procedure

	Command or Action	Purpose

Step 1	**configure terminal**
	Example:
	`switch# configure terminal`
	`switch(config)#`
		Enters global configuration mode.

Step 2	[**no**] **class-map type qos** *class-map-name*
	Example:
	`switch(config)# class-map type qos Class1`
	`switch(config-cmap-qos)#`
		Defines a class map, and enters class-map configuration mode.

Step 3	[**no**] **match** [**not**] **dscp** *dscp-list* **Example:** `switch(config)# switch(config-cmap-qos)# match dscp 2-4`	List of DSCP values. Specifies that the packets should be matched (or not) on the DSCP label in the MPLS header as follows: • **dscp-list**—The list can contain values and ranges. Values can range from 0 to 63.

Configuring MPLS Ingress Policing and Marking

To configure a policy-map value and set the EXP value on all imposed label entries, use the **set mpls experimental imposition** command in QoS policy-map class configuration mode. To disable the setting, use the **no** form of this command.

Procedure

	Command or Action	Purpose

Step 1	**configure terminal**
	
	Example:
	`switch# configure terminal`
	`switch(config)#`
		Enters global configuration mode.

Step 2	[**no**] **policy-map type qos** *policy-map-name*
	
	Example:
	`switch(config)# policy-map type qos pmap1`
	`switch(config-pmap-qos)#`
		Defines a policy map, and enters policy-map configuration mode.

Step 3	**class** *class-name* **Example:** `switch(config-pmap-qos)# class Class1`	Names the class-map.
Step 4	**set mpls experimental imposition** *exp_imposition_name* **Example:** `switch(config)# switch(config-pmap-qos)# set mpls experimental imposition 2`	MPLS experimental (EXP) values. Value range from 0 to 7.
Step 5	**set qos-group** *group-number* **Example:** `switch(config-cmap-qos)# set qos-group 1`	Identifies the qos-group number.
Step 6	**police cir** *burst-in-msec* **bc** *conform-burst-in-msec* **conform-action** *conform-action* **violate-action** *violate-action* **Example:** `switch(config-pmap-qos)# police cir 100 mbps bc 200 ms conform transmit violate drop`	Defines a policer for classified traffic in policy-map class configuration mode.

Step 7	**interface** *type slot/port*
	
	Example:
	`switch(config)# interface ethernet 2/2`
	`switch(config-if)#`
		Enters the interface configuration mode for the specified input interface, output interface, virtual circuit (VC), or a VC that will be used as the service policy for the interface or VC.

Step 8	**service-policy type qos input** *policy-map-name*
	
	Example:
	`switch(config-if)# service-policy type qos input pmap1`
	`switch(config-if)#`
		Attaches a policy map to an input interface, a virtual circuit (VC), an output interface, or a VC that will be used as the service policy for the interface or VC.

Configuring MPLS Transit Label Switching Router

To configure MPLS Transit Label Switching Routers, perform the following:

MPLS Transit LSR Classification

To map the value of the MPLS EXP field on all imposed label entries, use the **set mpls experimental topmost** command in QoS policy-map class configuration mode. To disable the setting, use the **no** form of this command.

Procedure

	Command or Action	Purpose

Step 1	**configure terminal**
	Example:
	`switch# configure terminal`
	`switch(config)#`
		Enters global configuration mode.

Step 2	[**no**] **class-map type qos** *class-map-name*
	Example:
	`switch(config)# class-map type qos Class1`
	`switch(config-cmap-qos)#`
		Defines a class map, and enters class-map configuration mode.

Step 3	[**no**] **match** [**not**] **mpls experimental topmost** *exp-list* **Example:** `switch(config)# switch(config-cmap-qos)# match mpls experimental topmost 2, 4-7`	List of MPLS experimental (EXP) values. Specifies that the packets should be matched (or not) on the 3-bit EXP field in the outermost (topmost) MPLS label in the MPLS header as follows: • **exp-list**—The list can contain values and ranges. Values can range from 0 to 7.

Configuring MPLS Transit Policing and Marking

To configure a policy-map value and set the EXP value on all imposed label entries, use the **service-policy type qos input pmap1** command in interface configuration mode. To disable the setting, use the **no** form of this command.

Procedure

	Command or Action	Purpose

Step 1	**configure terminal**
	Example:
	`switch# configure terminal`
	`switch(config)#`
		Enters global configuration mode.

Step 2	[**no**] **policy-map type qos** *policy-map-name* **Example:**	Defines a policy map, and enters policy-map configuration mode.

	Command or Action	Purpose
	`switch(config)# policy-map type qos Class1` `switch(config-pmap-qos)#`	
Step 3	**class** *class-name* **Example:** `switch(config-pmap-qos)# class Class1`	Names the class-map.
Step 4	**set mpls experimental imposition** *exp_imposition_name* **Example:** `switch(config)# switch(config-pmap-qos)# set mpls experimental imposition 2`	MPLS experimental (EXP) values. Value range from 0 to 7.
Step 5	**set qos-group** *group-number* **Example:** `switch(config-pmap-qos)# set qos-group 1`	Identifies the qos-group number.
Step 6	**police cir** *burst-in-msec* **bc** *conform-burst-in-msec* **conform-action** *conform-action* **violate-action** *violate-action* **Example:** `switch(config-pmap-qos)# police cir 100 mbps bc 200 ms conform transmit violate drop`	Defines a policer for classified traffic in policy-map class configuration mode. • violate-action - **drop** is the only supported keyword for Transit LSR

Step 7	**interface** *type slot/port*
	Example:
	`switch(config)# interface ethernet 2/2`
	`switch(config-if)#`
		Enters the interface configuration mode for the specified input interface, output interface, virtual circuit (VC), or a VC that will be used as the service policy for the interface or VC.

Step 8	**service-policy type qos input** *policy-map-name*
	Example:
	`switch(config-if)# service-policy type qos input pmap1`
	`switch(config-if)#`
		Attaches a policy map to an input interface, a virtual circuit (VC), an output interface, or a VC that is used as the service policy for the interface or VC.

Configuring MPLS Egress Label Switching Router

To configure MPLS Egress label switched router, perform the following:

MPLS Egress LSR Classification

To classify the incoming SR MPLS traffic to egress queue, use the match on Differentiated Services Code Point (DSCP) field.

Procedure

	Command or Action	Purpose

Step 1	**configure terminal**
	Example:
	`switch# configure terminal`
	`switch(config)#`
		Enters global configuration mode.

Step 2	[**no**] **class-map type qos** *class-map-name*
	Example:
	`switch(config)# class-map type qos Class1`
	`switch(config-cmap-qos)#`
		Defines a class map, and enters class-map configuration mode.

Step 3	[**no**] **match** [**not**] **dscp** *dscp-list* **Example:** `switch(config)# switch(config-cmap-qos)# match dscp 2-4`	List of DSCP values. Specifies that the packets should be matched (or not) on the DSCP label in the MPLS header as follows: • **dscp-list**—The list can contain values and ranges. Values can range from 0 to 63.

MPLS Egress LSR Classification - Default Policy Template

To classify the incoming traffic to the egress queue of an EVPN tunnel, use the default **default-mpls-in-policy** command at the system level. To disable the setting, use the **no** form of this command.

Procedure

	Command or Action	Purpose

Step 1	**configure terminal**
	Example:
	`switch# configure terminal`
	`switch(config)#`
		Enters global configuration mode.

Step 2	[**no**] **system qos**
	Example:
	`switch(config)# system qos`
	`switch(config-sys-qos)#`
		Enters system QoS configuration mode.

Step 3	[**no**] **service-policy type qos input default-mpls-in-policy** **Example:** `switch(config-sys-qos)# service-policy type qos input default-mpls-in-policy`	Specifies the “default-mpls-in-policy” at the system level to match on the incoming SR L3 EVPN MPLS traffic.

The following is the default MPLS in policy template configured with the **service-policy type qos input default-mpls-in-policy** command.

```
policy-map type qos default-mpls-in-policy
    class  c-dflt-mpls-qosgrp1
```

```
      set qos-group 1
    class  c-dflt-mpls-qosgrp2
      set qos-group 2
    class  c-dflt-mpls-qosgrp3
      set qos-group 3
    class  c-dflt-mpls-qosgrp4
      set qos-group 4
    class  c-dflt-mpls-qosgrp5
      set qos-group 5
    class  c-dflt-mpls-qosgrp6
      set qos-group 6
    class  c-dflt-mpls-qosgrp7
      set qos-group 7
    class  class-default
      set qos-group 0

class-map type qos match-any c-dflt-mpls-qosgrp1
  Description: This is an ingress default qos class-map that classify traffic with prec 1
  match precedence 1

class-map type qos match-any c-dflt-mpls-qosgrp2
  Description: This is an ingress default qos class-map that classify traffic with prec 2
  match precedence 2

class-map type qos match-any c-dflt-mpls-qosgrp3
  Description: This is an ingress default qos class-map that classify traffic with prec 3
  match precedence 3

class-map type qos match-any c-dflt-mpls-qosgrp4
  Description: This is an ingress default qos class-map that classify traffic with prec 4
  match precedence 4

class-map type qos match-any c-dflt-mpls-qosgrp5
  Description: This is an ingress default qos class-map that classify traffic with prec 5
  match precedence 5

class-map type qos match-any c-dflt-mpls-qosgrp6
  Description: This is an ingress default qos class-map that classify traffic with prec 6
  match precedence 6

class-map type qos match-any c-dflt-mpls-qosgrp7
  Description: This is an ingress default qos class-map that classify traffic with prec 7
  match precedence 7
```

Custom MPLS-in-Policy Mapping

You can override the queue mapping of incoming traffic by editing a local copy of the template provided. The system matching is always based on precedence, and requires the “mpls-in-policy” string to be part of the policy name. Marking with QoS is supported. Set can be qos-group, vlan-cos, or both.

```
class-map type qos match-all prec-1
      match precedence 1
    class-map type qos match-all prec-2
      match precedence 2

policy-map type qos test-mpls-in-policy
    class  prec-1
      set qos-group 3
    class  prec-2
      set qos-group 4
system qos
  service-policy type qos input test-mpls-in-policy
```

Note Classification based on Precedence is only supported and Marking is not supported on system level mpls-in-policy.

Configuring MPLS Egress LSR - Policing and Marking

To configure and apply a policy-map with policer config, use the **service-policy type qos input pmap1** command in interface configuration mode. To disable the setting, use the **no** form of this command.

Note Policing is not supported for SR L3 EVPN MPLS traffic

Procedure

	Command or Action	Purpose

Step 1	**configure terminal**
	Example:
	`switch# configure terminal`
	`switch(config)#`
		Enters global configuration mode.

Step 2	[**no**] **policy-map type qos** *class-map-name*
	Example:
	`switch(config)# policy-map type qos Class1`
	`switch(config-pmap-qos)#`
		Defines a class map, and enters class-map configuration mode.

Step 3	**policy** *policy-name* **Example:** `switch(config-pmap-qos)# class Class1`	Names the class-map.
Step 4	**set dscp** *dscp-value* **Example:** `switch(config-pmap-qos)# set dscp 4`	Identifies the dscp value.
Step 5	**set qos-group** *group-number* **Example:** `switch(config-pmap-qos)# set qos-group 1`	Identifies the qos-group number.
Step 6	[**no**] **police cir** *burst-in-msec* **bc** *conform-burst-in-msec* **conform-action** *conform-action* **violate-action** *violate-action* **Example:** `switch(config-pmap-qos)# police cir 100 mbps bc 200 ms conform transmit violate drop`	Defines a policer for classified traffic in policy-map class configuration mode.

	Command or Action	Purpose

Step 7	**interface** *type slot/port*
	Example:
	`switch(config)# interface ethernet 2/2`
	`switch(config-if)#`
		Enters the interface configuration mode for the specified interface.

Step 8	[**no**] **service-policy type qos input** *policy-map-name*
	Example:
	`switch(config-if)# service-policy type qos input pmap1`
	`switch(config-if)#`
		Attaches a policy map to an input interface, a virtual circuit (VC), an output interface, or a VC that will be used as the service policy for the interface or VC.

About Traffic Queuing

Traffic queuing is the ordering of packets and applies to both input and output of data. Device modules can support multiple queues, which you can use to control the sequencing of packets in different traffic classes. You can also set weighted random early detection (WRED) and taildrop thresholds. The device drops packets only when the configured thresholds are exceeded.

Configuring QoS Traffic Queuing

To set the output queue, use the **set qos-group** command in policy map configuration mode. To disable the setting, use the **no** form of this command.

Procedure

	Command or Action	Purpose

Step 1	**configure terminal**
	Example:
	`switch# configure terminal`
	`switch(config)#`
		Enters global configuration mode.

Step 2	[**no**] **policy-map type qos** *class-map-name*
	Example:
	`switch(config)# class-map type qos Class1`
	`switch(config-cmap-qos)#`
		Defines a class map, and enters class-map configuration mode.

Step 3	**class** *class-name* **Example:** `switch(config-cmap-qos)# class Class1`	Names the class-map.
Step 4	**set qos-group** *qos_group_number* **Example:** `switch(config-pmap-c-qos)# set qos-group`	Applies queueing parameters for the named QoS group in policy map. Value range from 0 to 7.

Verifying MPLS QoS

To display the MPLS QoS configuration, perform the following task:

Command	Description
show hardware internal forwarding table utilization	Displays information about the MAX label entries and Used label entries.
show class-map	Displays the interface class mapping statistics.
show policy-map system type qos input	Displays the cumulative statistics that show the packets matched for every class for all the interfaces (only for the EVPN tunnel case). For more information, see the sample output following this table.
show policy-map type qos interface interface	Displays the statistics that show the packets matched for every class on that interface in the given direction.
show policy-map type qos <pmap name>	Displays the service policy maps configured on the interfaces.
show queuing interface	Displays the queuing information of interfaces.

The following example displays the cumulative statistics that show the packets matched for every class for all the interfaces (only for the EVPN tunnel case).

```
switch# show policy-map system type qos input


  Service-policy (qos) input:   default-mpls-in-policy

    Class-map (qos):   c-dflt-mpls-qosgrp1 (match-any)

     Slot 3
        2775483 packets
     Aggregate forwarded :
        2775483 packets
      Match: precedence 1
      set qos-group 1

    Class-map (qos):   c-dflt-mpls-qosgrp2 (match-any)

     Slot 3
        2775549 packets
     Aggregate forwarded :
        2775549 packets
```

```
  Match: precedence 2
  set qos-group 2

Class-map (qos):   c-dflt-mpls-qosgrp3 (match-any)

 Slot 2
    2777189 packets
 Aggregate forwarded :
    2777189 packets
  Match: precedence 3
  set qos-group 3

Class-map (qos):   c-dflt-mpls-qosgrp4 (match-any)

 Slot 3
    2775688 packets
 Aggregate forwarded :
    2775688 packets
  Match: precedence 4
  set qos-group 4

Class-map (qos):   c-dflt-mpls-qosgrp5 (match-any)

 Slot 3
    2775756 packets
 Aggregate forwarded :
    2775756 packets
  Match: precedence 5
  set qos-group 5

Class-map (qos):   c-dflt-mpls-qosgrp6 (match-any)

 Slot 3
    2775824 packets
 Aggregate forwarded :
    2775824 packets
  Match: precedence 6
  set qos-group 6

Class-map (qos):   c-dflt-mpls-qosgrp7 (match-any)

 Slot 3
    2775892 packets
 Aggregate forwarded :
    2775892 packets
  Match: precedence 7
  set qos-group 7

Class-map (qos):   class-default (match-any)

 Slot 3
    2775962 packets
 Aggregate forwarded :
    2775962 packets
  set qos-group 0
```

CHAPTER 9

Configuring Segment Routing

This chapter contains information on how to configure segment routing.

About Segment Routing

Segment routing is a technique by which the path followed by a packet is encoded in the packet itself, similar to source routing. A node steers a packet through a controlled set of instructions, called segments, by prepending the packet with a segment routing header. Each segment is identified by a segment ID (SID) consisting of a flat unsigned 32-bit integer.

Border Gateway Protocol (BGP) segments, a subclass of segments, identify a BGP forwarding instruction. There are two groups of BGP segments: prefix segments and adjacency segments. Prefix segments steer packets along the shortest path to the destination, using all available equal-cost multi-path (ECMP) paths.

Adjacency segments steer packets onto a specific link to a neighbor.

The segment routing architecture is applied directly to the MPLS data plane.

Segment Routing Application Module

Segment Routing Application (SR-APP) module is used to configure the segment routing functionality. Segment Routing Application (SR-APP) is a separate internal process that handles all the CLIs related to segment routing. It is responsible for reserving the SRGB range and for notifying the clients about it. It is also responsible for maintaining the prefix to SID mappings. The SR-APP support is also available for the BGP, IS-IS, and OSPF protocols.

The SR-APP module maintains the following information:

- Segment routing operation state
- Segment routing global block label ranges
- Prefix SID mappings

For more information, see Configuring Segment Routing, on page 116.

NetFlow for MPLS

NetFlow identifies packet flows for ingress IP packets and provides statistics that are based on these packet flows. NetFlow does not require any change to either the packets themselves or to any networking device. You can export the data that NetFlow gathers for your flow by using a flow exporter and export this data to a remote NetFlow Collector, such as Cisco Stealthwatch. Cisco NX-OS exports flow as part of a NetFlow export User Datagram Protocol (UDP) datagram. You can export the data that NetFlow gathers for your flow by using a flow exporter and export this data to a remote NetFlow Collector, such as Cisco Stealthwatch. Cisco NX-OS exports a flow as part of a NetFlow export User Datagram Protocol (UDP) datagram.

Beginning with Cisco NX-OS Release 9.3(1), NetFlow Collector over segment routing is supported on Cisco Nexus 9300-EX, 9300-FX, 9300-FX2, 9500-EX, and 9500-FX platform switches.

Beginning with Cisco NX-OS Release 9.3(5), NetFlow Collector over segment routing is supported on Cisco Nexus 9300-FX3 platform switches.

NetFlow is not supported on Cisco Nexus 9300-GX platform switches..

NetFlow Collector supports both, single and double MPLS labels. Both, default and the non-default VRF in the exporter destination configurations is supported. NetFlow does not support an MPLS data path.

Since segment routing does not support a single label, you must configure the **address-family ipv4 labeled-unicast** command under BGP neighbor and the **allocate-label** command under the bgp configuration.

sFlow Collector

Sampled flow (sFlow) allows you to monitor real-time traffic in data networks that contain switches and routers. It uses the sampling mechanism in the sFlow agent software on switches and routers to monitor traffic and to forward the sample data to the central data collector.

Beginning with Cisco NX-OS Release 9.3(1), sFlow collector over segment routing is supported on Cisco Nexus 9300-EX, 9300-FX, 9300-FX2, 9500-EX, and 9500-FX platform switches.

Beginning Cisco NX-OS Release 9.3(5), sFlow collector over segment routing is supported on Cisco Nexus 9300-FX3 platform switches.

sFlow is not supported on Cisco Nexus 9364C-GX, Cisco Nexus 9316D-GX, and Cisco Nexus 93600CD-GX switches.

For information on configuring sFlow, see the *Configuring sFlow* section in the *Cisco Nexus 9000 Series NX-OS System Management Configuration Guide, Release 9.3(x)*.

Guidelines and Limitations for Segment Routing

Segment routing has the following guidelines and limitations:

- MPLS segment routing is not supported for FEX modules.
- Beginning with Cisco NX-OS Release 9.3(1), the **segment-routing mpls** command has changed to **segment-routing**.
- When you enable MPLS segment routing on Cisco Nexus 9504 and 9508 platform switches with a -R series line card, there can be instances of the BFD sessions going down and coming back. BGP peerings, if configured with BFD, also go down and come back up. When a BGP session goes down, it withdraws routes from the hardware. This results in packet loss until the BGP session is re-established and routes are re-installed. However, once the BFD comes up, no additional flaps occurs.
- You can run segment routing under IGP(like OSPF) or by AF labeled unicast in BGP.
- Segment Routing is supported on Cisco Nexus 9300-FX platform switches and the Cisco Nexus N9K-X9736C-FX line cards.
- Segment routing and SR-EVPN are supported on Cisco Nexus C31108PC-V, C31108TC-V, and C3132Q-V switches.
- Beginning with Cisco NX-OS Release 9.3(3), you can configure Layer 3 VPNs on Cisco Nexus 9300-GX platform switches.
- Beginning with Cisco NX-OS Release 9.3(3), segment routing and SR-EVPN is supported on Cisco Nexus 9364C-GX, Cisco Nexus 9316D-GX, and Cisco Nexus 93600CD-GX platform switches.
- Beginning with Cisco NX-OS Release 9.3(3), adjacency SIDs on OSPF are supported on Cisco Nexus 9364C-GX, Cisco Nexus 9316D-GX, and Cisco Nexus 93600CD-GX platform switches.
- Beginning with Cisco NX-OS Release 9.3(3), segment routing with OSPF, IS-IS underlay, and BGP labeled unicast is supported on Cisco Nexus 9364C-GX, Cisco Nexus 9316D-GX, and Cisco Nexus 93600CD-GX platform switches.
- BGP allocates the SRGB label for iBGP route-reflector clients only when next-hop-self is in effect (for example, the prefix is advertised with the next hop being one of the local IP/IPv6 addresses on RR). When you have configured next-hop-self on an RR, the next hop is changed for the routes that are being affected (subject to route-map filtering).
- A nondisruptive ISSU is not supported with MPLS features for Cisco Nexus 9300-EX and 9300-FX platform switches.
- Static MPLS, MPLS segment routing, and MPLS stripping cannot be enabled at the same time.
- Beginning with Cisco NX-OS Release 9.3(5), MPLS stripping is supported on Cisco Nexus 9300-GX platform switches and the following guidelines are applicable:
    - For the MPLS strip feature to work, both the **mpls strip** and the **hardware acl tap-agg** commands should be configured after the switches are reloaded.

- When the MPLS strip is enabled on the Cisco Nexus 9300-GX platform switches, the ACL log process is not displayed.
- MPLS strip with dot1q VLAN is not supported.
- For all double VLAN tags, the second VLAN range should be between 2-510.
- MPLS strip with dot1q is not supported.
- For PACL redirect support, you must use the **mode tap-aggregation** command on the ingress TAP interface.

- Because static MPLS, MPLS segment routing, and MPLS stripping are mutually exclusive, the only segment routing underlay for multi-hop BGP is single-hop BGP. iBGP multi-hop topologies with eBGP running as an overlay are not supported.
- MPLS pop followed by a forward to a specific interface is not supported. The penultimate hop pop (PHP) is avoided by installing the Explicit NULL label as the outlabel in the label FIB (LFIB) even when the control plane installs an IPv4 Implicit NULL label.
- BGP labeled unicast and BGP segment routing are not supported for IPv6 prefixes.
- BGP labeled unicast and BGP segment routing are not supported over tunnel interfaces (including GRE and VXLAN) or with vPC access interfaces.
- MTU path discovery (RFC 2923) is not supported over MPLS label switched paths (LSPs) or segment routed paths.
- For the Cisco Nexus 9200 Series switches, adjacency statistics are not maintained for Layer 3 or MPLS adjacencies.
- For the Cisco Nexus 9500 Series switches, MPLS LSPs and segment routed paths are not supported on subinterfaces (either port channels or normal Layer 3 ports).
- For the Cisco Nexus 9500 platform switches, segment routing is supported only in the nonhierarchical routing mode.
- The BGP configuration commands **neighbor-down fib-accelerate** and **suppress-fib-pending** are not supported for MPLS prefixes.
- The uniform model as defined in RFC 2973 and RFC 3270 is not supported. Therefore, the IP DSCP bits are not copied into the imposed MPLS header.
- Reconfiguration of the segment routing global block (SRGB) results in an automatic restart of the BGP process to update the existing URIB and ULIB entries. Traffic loss occurs for a few seconds, so you should not reconfigure the SRGB in production.
- If the segment routing global block (SRGB) is set to a range but the route-map label-index delta value is outside of the configured range, the allocated label is dynamically generated. For example, if the SRGB is set to range of 16000-23999 when a route-map label-index is set to 9000, the label is dynamically allocated.
- For network scalability, Cisco recommends using a hierarchical routing design with multi-hop BGP for advertising the attached prefixes from a top-of-rack (ToR) or border leaf switch.
- BGP sessions are not supported over MPLS LSPs or segment routed paths.
- The Layer 3 forwarding consistency checker is not supported for MPLS routes.

- You can configure segment routing traffic engineering with on-demand next hop on Cisco Nexus 9000 Series switches.
- Layer 3 VPN and Layer 3 EVPN stitching for segment routing is supported on Cisco Nexus 9000 Series switches.
- Beginning with Cisco NX-OS Release 9.3(3), Layer 3 VPN and Layer 3 EVPN stitching for segment routing is supported on 9300-GX platform switches.
- You can configure OSPFv2 as an IGP control plane for segment routing on Cisco Nexus 9000 Series switches.
- Layer 3 VPN and Layer 3 EVPN Stitching for segment routing is not supported on Cisco Nexus 9364C, 9200, 9300-EX, and 9500 platform switches with the -EX line cards.
- The OSPF segment routing command and segment-routing traffic engineering with on-demand next hop is not supported on Cisco Nexus 9364C switches.
- Segment Routing is supported on Cisco Nexus 9300-FX2 and 9300-FX3 platform switches.
- Layer 3 VPN and Layer 3 EVPN Stitching for Segment Routing, the OSPF segment routing command, and the segment-routing traffic engineering with on-demand next hop is supported on Cisco Nexus 9364C switches.
- Layer 3 VPN over Segment Routing is supported on Cisco Nexus 3100, 3200, 9200, 9300, 9300-EX/FX/FX2/FX3 platform switches and Cisco Nexus 9500 platform switches with -EX/FX and -R line cards.
- Deleting the segment routing configuration removes all the related segment routing configurations including the MPLS and the traffic engineering configurations.
- If you downgrade the Cisco Nexus device from Cisco NX-OS Release 9.3(1) to the previous NX-OS releases by setting the boot variables and reloading the switch, all earlier configurations of the segment-routing MPLS are lost.
- Before performing an ISSD from Cisco NX-OS Release 9.3(1), you must disable the segment routing configuration. Failure to do so will result in the loss of the existing segment routing configurations.
- Segment routing MPLS adjacency statistics are collected based on the out label stack and the next hop on the intermediate nodes. However, in the PHP mode, the statistics are shown on all adjacencies because the same stack is shared on all the FECs.
- If segment routing is enabled on a switch, Q-in-Q tagging on a dot1Q tagged MPLS packet is not supported, packets egress with only the outer tag.

  For example: Consider an ingress port in access dot1q tunnel mode, with VLAN 100. Incoming MPLS traffic has a dot1Q tag of 200. Typically, the traffic should egress with an outer tag of 100, and inner tag of 200 (same as the tag of the incoming packet). However, the packet egresses with an outer tag and loses the inner tag.
- When an incoming MPLS packet is untagged and the ingress port is in access VLAN mode, packets egress without any tag, if segment routing is enabled.
- We recommend that you do not configure segment routing using BGP, OSPF, and IS-IS underlay simultaneously.

Configuring Segment Routing

Configuring Segment Routing

Before you begin

Confirm that the following conditions are met before configuring segment routing.

- The **install feature-set mpls**, **feature-set mpls** and **feature mpls segment-routing** commands should be present before configuring the **segment-routing** command.
- If the global block is configured, the specified range is used. Otherwise, the default 16000 – 23999 range is used.
- BGP now uses both **set label-index** <*value*> configuration and the new **connected-prefix-sid-map** CLI. In case of a conflict, the configuration in SR-APP is preferred.

Procedure

	Command or Action	Purpose

Step 1	**configure terminal**
	Example:
	`switch# configure terminal`
	`switch(config)#`
		Enters global configuration mode.

Step 2	**segment-routing**
	Example:
	`switch(config)# segment-routing`
	`switch(config-sr)# mpls`
	`switch(config-sr-mpls)#`
		Enables the MPLS segment routing functionality. The **no** form of this command disables the MPLS segment routing feature.

Step 3	**connected-prefix-sid-map**
	Example:
	`switch(config-sr-mpls)#`
	`connected-prefix-sid-map`
	`switch(config-sr-mpls)#`
		Configures the connected prefix segment identifier mappings.

Step 4	**global-block** <*min*> <*max*>
	Example:
	`switch(config-sr-mpls)# global-block`
	`<min> <max>`
	`switch(config-sr-mpls)#`
		Specifies the global block range for the segment routing bindings.

Step 5	**connected-prefix-sid-map**
	Example:
	`switch(config-sr-mpls)#`
	`connected-prefix-sid-map`
	`switch(config-sr-mpls-conn-pfsid)#`
		Configures the connected prefix segment identifier mappings.

	Command or Action	Purpose
Step 6	**address-family ipv4** **Example:** `switch(config-sr-mpls-conn-pfsid)#address-family ipv4`	Configures the IPv4 address family.
Step 7	*<prefix>*/*<masklen>* [**index**|**absolute**] *<label>* **Example:** `switch(config-sr-mpls)# 2.1.1.5/32 absolute 201101`	The optional keywords **index** or **absolute** indicate whether the label value entered should be interpreted as an index into the SRGB or as an absolute value.

Example

See the following configuration examples of the show commands:

```
switch# show segment-routing mpls
Segment-Routing Global info

Service Name: segment-routing

State: Enabled

Process Id: 29123

Configured SRGB: 17000 - 24999

SRGB Allocation status: Alloc-Successful

Current SRGB: 17000 - 24999

Cleanup Interval: 60

Retry Interval: 180
```

The following CLI displays the clients that are registered with SR-APP. It lists the VRFs, for which the clients have registered interest.

```
switch# show segment-routing mpls clients
            Segment-Routing Mpls Client Info

Client: isis-1
    PIB index: 1    UUID: 0x41000118    PID: 29463    MTS SAP: 412
    TIBs registered:
        VRF: default Table: base

Client: bgp-1
    PIB index: 2    UUID: 0x11b    PID: 18546    MTS SAP: 62252
    TIBs registered:
        VRF: default Table: base

Total Clients: 2
```

In the **show segment-routing mpls ipv4 connected-prefix-sid-map** CLI command example, SRGB indicates whether the prefix SID is within the configured SRGB. The **Indx** field indicates that the configured label is an index into the global block. The **Abs** field indicates that the configured label is an absolute value.

If the SRGB field displays N, it means that the configured prefix SID is not within the SRGB range and it is not provided to the SR-APP clients. Only the prefix SIDs that fall into the SRGB range are given to the SR-APP clients.

```
switch# show segment-routing mpls ipv4 connected-prefix-sid-map
            Segment-Routing Prefix-SID Mappings
Prefix-SID mappings for VRF default Table base
Prefix             SID   Type Range SRGB
13.11.2.0/24       713   Indx 1     Y
30.7.7.7/32        730   Indx 1     Y
59.3.24.0/30       759   Indx 1     Y
150.101.1.0/24     801   Indx 1     Y
150.101.1.1/32     802   Indx 1     Y
150.101.2.0/24     803   Indx 1     Y
1.1.1.1/32         16013 Abs  1     Y
```

The following CLI displays the **show running-config segment-routing** output.

```
switch# show running-config segment-routing ?

> Redirect it to a file
>> Redirect it to a file in append mode
all Show running config with defaults
| Pipe command output to filter

switch# show running-config segment-routing
switch# show running-config segment-routing

!Command: show running-config segment-routing
!Running configuration last done at: Thu Dec 12 19:39:52 2019
!Time: Thu Dec 12 20:06:07 2019

version 9.3(3) Bios:version 05.39
segment-routing
    mpls
        connected-prefix-sid-map
            address-family ipv4
                2.1.1.1/32 absolute 100100

switch#
```

Enabling MPLS on an Interface

You can enable MPLS on an interface for use with segment routing.

Before you begin

You must install and enable the MPLS feature set using the **install feature-set mpls** and **feature-set mpls** commands.

Procedure

	Command or Action	Purpose

Step 1	**configure terminal**
	Example:
	`switch# configure terminal`
	`switch(config)#`
		Enters global configuration mode.

Step 2	**interface** *type slot/port*
	Example:
	`switch(config)# interface ethernet 2/2`
	`switch(config-if)#`
		Enters the interface configuration mode for the specified interface.

Step 3	[**no**] **mpls ip forwarding** **Example:** `switch(config-if)# mpls ip forwarding`	Enables MPLS on the specified interface. The **no** form of this command disables MPLS on the specified interface.
Step 4	(Optional) **copy running-config startup-config** **Example:** `switch(config-if)# copy running-config startup-config`	Copies the running configuration to the startup configuration.

Configuring the Segment Routing Global Block

You can configure the beginning and ending MPLS labels in the segment routing global block (SRGB).

Before you begin

You must install and enable the MPLS feature set using the **install feature-set mpls** and **feature-set mpls** commands.

You must enable the MPLS segment routing feature.

Procedure

	Command or Action	Purpose

Step 1	**configure terminal**
	Example:
	`switch# configure terminal`
	`switch(config)#`
		Enters global configuration mode.

Step 2	[**no**] **segment-routing**
	Example:
	`switch(config)# segment-routing`
	`switch(config-sr)# mpls`
		Enters the segment routing configuration mode and enables the default SRGB of 16000 to 23999. The **no** form of this command unallocates that block of labels.
		
		If the configured dynamic range cannot hold the default SRGB, an error message appears, and the default SRGB will not be allocated. If

	Command or Action	Purpose
		desired, you can configure a different SRGB in the next step.
Step 3	[**no**] **global-block** *beginning-label ending-label* **Example:** `switch(config-sr-mpls)# global-block 16000 471804`	Specifies the MPLS label range for the SRGB. Use this command if you want to change the default SRGB label range that is configured with the **segment-routing** command. The permissive values for the beginning MPLS label and the ending MPLS label are from 16000 to 471804. The **mpls label range** command permits 16 as the minimum label, but the SRGB can start only from 16000. **Note** The minimum value for the **global-block** command starts from 16000. If you upgrading from previous releases, you should modify the SRGB so that it falls within the supported range before triggering an upgrade.
Step 4	(Optional) **show mpls label range** **Example:** `switch(config-sr-mpls)# show mpls label range`	Displays the SRGB, only if the SRGB allocation is successful.
Step 5	**show segment-routing**	Displays the configured SRGB.
Step 6	**show segment-routing mpls** **Example:** `switch(config-sr-mpls)# show segment-routing mpls`	Displays the configured SRGB.
Step 7	(Optional) **copy running-config startup-config** **Example:** `switch(config-sr-mpls)# copy running-config startup-config`	Copies the running configuration to the startup configuration.

Configuring the Label Index

You can set the label index for routes that match the **network** command. Doing so causes the BGP prefix SID to be advertised for local prefixes that are configured with a route map that includes the **set label-index** command, provided the route map is specified in the **network** command that specifies the local prefix. (For more information on the **network** command, see the "Configuring Basic BGP" chapter in the Cisco Nexus 9000 Series NX-OS Unicast Routing Configuration Guide.)

Note Segment Routing Application (SR-APP) module is used to configure the segment routing functionality. BGP now uses both **set label-index** <*value*> configuration under route-map and the new **connected-prefix-sid-map** CLI for prefix SID configuration. In case of a conflict, the configuration in SR-APP is preferred.

Note Route-map label indexes are ignored when the route map is specified in a context other than the **network** command. Also, labels are allocated for prefixes with a route-map label index independent of whether the prefix has been configured by the **allocate-label route-map** *route-map-name* command.

Procedure

	Command or Action	Purpose

Step 1	**configure terminal**
	Example:
	`switch# configure terminal`
	`switch(config)#`
		Enters global configuration mode.

Step 2	**route-map** *map-name*
	Example:
	`switch(config)# route-map SRmap`
	`switch(config-route-map)#`
		Creates a route map or enters route-map configuration mode for an existing route map.

Step 3	[**no**] **set label-index** *index* **Example:** `switch(config-route-map)# set label-index 10`	Sets the label index for routes that match the **network** command. The range is from 0 to 471788. By default, a label index is not added to the route.

Step 4	**exit**
	Example:
	`switch(config-route-map)# exit`
	`switch(config)#`
		Exits route-map configuration mode.

Step 5	**router bgp** *autonomous-system-number*
	Example:
	`switch(config)# router bgp 64496`
	`switch(config-router)#`
		Enables BGP and assigns the AS number to the local BGP speaker. The AS number can be a 16-bit integer or a 32-bit integer in the form of a higher 16-bit decimal number and a lower 16-bit decimal number in xx.xx format.

Step 6	Required: **address-family ipv4 unicast**
	Example:
	`switch(config-router)# address-family ipv4 unicast`
	`switch(config-router-af)#`
		Enters global address family configuration mode for the IPv4 address family.

	Command or Action	Purpose
Step 7	**network** *ip-prefix* [**route-map** *map-name*] **Example:** `switch(config-router-af)# network 10.10.10.10/32 route-map SRmap`	Specifies a network as local to this autonomous system and adds it to the BGP routing table.
Step 8	(Optional) **show route-map** [*map-name*] **Example:** `switch(config-router-af)# show route-map`	Displays information about route maps, including the label index.
Step 9	(Optional) **copy running-config startup-config** **Example:** `switch(config-router-af)# copy running-config startup-config`	Copies the running configuration to the startup configuration.

Configuration Examples for Segment Routing

The examples in this section show a common BGP prefix SID configuration between two routers.

This example shows how to advertise a BGP speaker configuration of 10.10.10.10/32 and 20.20.20.20/32 with a label index of 10 and 20, respectively. It uses the default segment routing global block (SRGB) range of 16000 to 23999.

```
hostname s1
install feature-set mpls
feature-set mpls

feature telnet
feature bash-shell
feature scp-server
feature bgp
feature mpls segment-routing

segment-routing
  mpls
  vlan 1
segment-routing
  mpls
    connected-prefix-sid-map
    address-family ipv4
    2.1.1.1/32 absolute 100100

route-map label-index-10 permit 10
  set label-index 10
route-map label-index-20 permit 10
  set label-index 20

vrf context management
  ip route 0.0.0.0/0 10.30.108.1

interface Ethernet1/1
  no switchport
  ip address 10.1.1.1/24
  no shutdown

interface mgmt0
```

```
  ip address dhcp
  vrf member management

interface loopback1
  ip address 10.10.10.10/32

interface loopback2
  ip address 20.20.20.20/32

line console
line vty

router bgp 1
  address-family ipv4 unicast
    network 10.10.10.10/32 route-map label-index-10
    network 20.20.20.20/32 route-map label-index-20
    allocate-label all
  neighbor 10.1.1.2 remote-as 2
    address-family ipv4 labeled-unicast
```

This example shows how to receive the configuration from a BGP speaker.

```
hostname s2
install feature-set mpls
feature-set mpls

feature telnet
feature bash-shell
feature scp-server
feature bgp
feature mpls segment-routing

segment-routing mpls
vlan 1

vrf context management
  ip route 0.0.0.0/0 10.30.97.1
  ip route 0.0.0.0/0 10.30.108.1

interface Ethernet1/1
  no switchport
  ip address 10.1.1.2/24
  ipv6 address 10:1:1::2/64
  no shutdown

interface mgmt0
  ip address dhcp
  vrf member management

interface loopback1
  ip address 2.2.2.2/32
line console

line vty

router bgp 2
  address-family ipv4 unicast
    allocate-label all
  neighbor 10.1.1.1 remote-as 1
    address-family ipv4 labeled-unicast
```

This example shows how to display the configuration from a BGP speaker. The **show** command in this example displays the prefix 10.10.10.10 with label index 10 mapping to label 16010 in the SRGB range of 16000 to 23999.

```
switch# show bgp ipv4 labeled-unicast 10.10.10.10/32

BGP routing table information for VRF default, address family IPv4 Label Unicast
BGP routing table entry for 10.10.10.10/32, version 7
Paths: (1 available, best #1)
Flags: (0x20c001a) on xmit-list, is in urib, is best urib route, is in HW, , has label
  label af: version 8, (0x100002) on xmit-list
  local label: 16010

  Advertised path-id 1, Label AF advertised path-id 1
  Path type: external, path is valid, is best path, no labeled nexthop, in rib
  AS-Path: 1 , path sourced external to AS
    10.1.1.1 (metric 0) from 10.1.1.1 (10.10.10.10)
      Origin IGP, MED not set, localpref 100, weight 0
      Received label 0
      Prefix-SID Attribute: Length: 10
        Label Index TLV: Length 7, Flags 0x0 Label Index 10

  Path-id 1 not advertised to any peer
  Label AF advertisement
  Path-id 1 not advertised to any peer
```

This example shows how to configure egress peer engineering on a BGP speaker.

```
hostname epe-as-1
install feature-set mpls
feature-set mpls

feature telnet
feature bash-shell
feature scp-server
feature bgp
feature mpls segment-routing

segment-routing mpls
vlan 1

vrf context management
  ip route 0.0.0.0/0 10.30.97.1
  ip route 0.0.0.0/0 10.30.108.1

interface Ethernet1/1
  no switchport
  ip address 10.1.1.1/24
  no shutdown

interface Ethernet1/2
  no switchport
  ip address 11.1.1.1/24
  no shutdown

interface Ethernet1/3
  no switchport
  ip address 12.1.1.1/24
  no shutdown

interface Ethernet1/4
  no switchport
  ip address 13.1.1.1/24
```

```
  no shutdown

interface Ethernet1/5
  no switchport
  ip address 14.1.1.1/24
  no shutdown
```

The following is an example of show ip route vrf 2 command.

```
show ip route vrf 2
IP Route Table for VRF "2"
'*' denotes best ucast next-hop
'**' denotes best mcast next-hop
'[x/y]' denotes [preference/metric]
'%<string>' in via output denotes VRF <string>

41.11.2.0/24, ubest/mbest: 1/0
    *via 1.1.1.9%default, [20/0], 13:26:48, bgp-2, external, tag 11 (mpls-vpn)
42.11.2.0/24, ubest/mbest: 1/0, attached
    *via 42.11.2.1, Vlan2, [0/0], 13:40:52, direct
42.11.2.1/32, ubest/mbest: 1/0, attached
    *via 42.11.2.1, Vlan2, [0/0], 13:40:52, local
```

The following is an example of **show forwarding route vrf 2** command.

```
slot  1
=======

IPv4 routes for table 2/base

------------------+-----------------------------------------+----------------------+------------------+-----------------
Prefix            | Next-hop                                | Interface            | Labels
          | Partial Install
------------------+-----------------------------------------+----------------------+------------------+-----------------
0.0.0.0/32           Drop                                      Null0
127.0.0.0/8          Drop                                      Null0
255.255.255.255/32   Receive                                   sup-eth1
*41.11.2.0/24        27.1.31.4                                 Ethernet1/3            PUSH
  30002 492529
                     27.1.32.4                                 Ethernet1/21           PUSH
  30002 492529
                     27.1.33.4                                 port-channel23         PUSH
  30002 492529
                     27.11.31.4                                Ethernet1/3.11         PUSH
  30002 492529
                     27.11.33.4                                port-channel23.11      PUSH
  30002 492529
                     37.1.53.4                                 Ethernet1/53/1         PUSH
  29002 492529
                     37.1.54.4                                 Ethernet1/54/1         PUSH
  29002 492529
                     37.2.53.4                                 Ethernet1/53/2         PUSH
  29002 492529
                     37.2.54.4                                 Ethernet1/54/2         PUSH
  29002 492529
                     80.211.11.1                               Vlan801                PUSH
  30002 492529
```

The following is an example of **show bgp l2vpn evpn summary** command.

```
show bgp l2vpn evpn summary
BGP summary information for VRF default, address family L2VPN EVPN
BGP router identifier 2.2.2.3, local AS number 2
BGP table version is 17370542, L2VPN EVPN config peers 4, capable peers 1
1428 network entries and 1428 paths using 268464 bytes of memory
BGP attribute entries [476/76160], BGP AS path entries [1/6]
BGP community entries [0/0], BGP clusterlist entries [0/0]
476 received paths for inbound soft reconfiguration
476 identical, 0 modified, 0 filtered received paths using 0 bytes

Neighbor        V    AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down  State/PfxRcd
1.1.1.1         4    11       0       0        0    0    0 23:01:53 Shut (Admin)
1.1.1.9         4    11    4637    1836 17370542    0    0 23:01:40 476
1.1.1.10        4    11       0       0        0    0    0 23:01:53 Shut (Admin)
1.1.1.11        4    11       0       0        0    0    0 23:01:52 Shut (Admin)
```

The following is an example of **show bgp l2vpn evpn** command.

```
show bgp l2vpn evpn 41.11.2.0
BGP routing table information for VRF default, address family L2VPN EVPN
Route Distinguisher: 14.1.4.1:115
BGP routing table entry for [5]:[0]:[0]:[24]:[41.11.2.0]:[0.0.0.0]/224, version 17369591
Paths: (1 available, best #1)
Flags: (0x000002) on xmit-list, is not in l2rib/evpn, is not in HW

  Advertised path-id 1
  Path type: external, path is valid, received and used, is best path
             Imported to 2 destination(s)
  AS-Path: 11 , path sourced external to AS
    1.1.1.9 (metric 0) from 1.1.1.9 (14.1.4.1)
      Origin incomplete, MED 0, localpref 100, weight 0
      Received label 492529
      Extcommunity: RT:2:20

  Path-id 1 not advertised to any peer

Route Distinguisher: 2.2.2.3:113
BGP routing table entry for [5]:[0]:[0]:[24]:[41.11.2.0]:[0.0.0.0]/224, version 17369595
Paths: (1 available, best #1)
Flags: (0x000002) on xmit-list, is not in l2rib/evpn, is not in HW

  Advertised path-id 1
  Path type: external, path is valid, is best path
             Imported from 14.1.4.1:115:[5]:[0]:[0]:[24]:[41.11.2.0]:[0.0.0.0]/224
  AS-Path: 11 , path sourced external to AS
    1.1.1.9 (metric 0) from 1.1.1.9 (14.1.4.1)
```

Configuring Segment Routing with IS-IS Protocol

About IS-IS

IS-IS is an Interior Gateway Protocol (IGP) based on Standardization (ISO)/International Engineering Consortium (IEC) 10589 and RFC 1995. Cisco NX-OS supports Internet Protocol version 4 (IPv4) and IPv6. IS-IS is a dynamic link-state routing protocol that can detect changes in the network topology and calculate loop-free routes to other nodes in the network. Each router maintains a link-state database that describes the state of the network and sends packets on every configured link to discover neighbors. IS-IS floods the link-state information across the network to each neighbor. The router also sends advertisements and updates on the link-state database through all the existing neighbors

Segment routing on the IS-IS protocol supports the following:

- IPv4
- Level 1, level 2, and multi-level routing
- Prefix SIDs
- Multiple IS-IS instances on the same loopback interface for domain border nodes
- Adjacency SIDs for adjacencies

Configuring Segment Routing with IS-IS Protocol

You can configure segment routing with IS-IS protocol.

Before you begin

IS-IS segment routing is fully enabled when the following conditions are met:

- The **mpls segment-routing** feature is enabled.
- The IS-IS feature is enabled.
- Segment routing is enabled for at least one address family under IS-IS.

Procedure

	Command or Action	Purpose
Step 1	**configure terminal**	Enters global configuration mode.
Step 2	**router isis** *instance-tag*	Creates a new IS-IS instance with the configured instance tag.
Step 3	**net** *network-entity-title*	Configures the NET for this IS-IS instance.
Step 4	**address-family** *ipv4* **unicast**	Enters address family configuration mode.
Step 5	**segment-routing mpls**	Configures segment routing with IS-IS protocol.

	Command or Action	Purpose
		Note • The IS-IS command is supported only on the IPv4 address family. It is not supported on the IPv6 address family. • Redistribution is not supported from any other protocol to ISIS for the SR prefixes. You need to enable **ip router isis** command on all the prefix SID interfaces.

Configuring Segment Routing with OSPFv2 Protocol

About OSPF

Open Shortest Path First (OSPF) is an Interior Gateway Protocol (IGP) developed by the OSPF working group of the Internet Engineering Task Force (IETF). Designed expressly for IP networks, OSPF supports IP subnetting and tagging of externally derived routing information. OSPF also allows packet authentication and uses IP multicast when sending and receiving packets.

Segment routing configuration on the OSPF protocol can be applied at the process or the area level. If you configure segment routing at the process level, it is enabled for all the areas. However, you can enable ore disable it per area level.

Segment routing on the OSPF protocol supports the following:

- OSPFv2 control plane
- Multi-area
- IPv4 prefix SIDs for host prefixes on loopback interfaces
- Adjacency SIDs for adjacencies

Adjacency SID Advertisement

OSPF supports the advertisement of segment routing adjacency SID. An Adjacency Segment Identifier (Adj-SID) represents a router adjacency in Segment Routing.

A segment routing-capable router may allocate an Adj-SID for each of its adjacencies and an Adj-SID sub-TLV is defined to carry this SID in the Extended Opaque Link LSA.

OSPF allocates the adjacency SID for each OSPF neighbor if the OSPF adjacency which are in two way or in FULL state. OSPF allocates the adjacency SID only if the segment routing is enabled. The label for adjacency SID is dynamically allocated by the system. This eliminates the chances of misconfiguration, as this has got only the local significance.

Connected Prefix-SID

OSPFv2 supports the advertisement of prefix SID for address associated with the loopback interfaces. In order to achieve this, OSPF uses Extended Prefix Sub TLV in its opaque Extended prefix LSA. When OSPF receives this LSA from its neighbor, SR label is added to the RIB corresponding to received prefix based upon the information present in extended prefix sub TLV.

For configuration, segment-routing has to be enabled under OSPF and corresponding to loopback interface that is configured with OSPF, prefix-sid mapping is required under the segment routing module.

Note SID will only be advertised for loopback addresses and only for intra-area and inter-area prefix types. No SID value will be advertised for external or NSSA prefixes.

Prefix Propagation Between Areas

To provide segment routing support across the area boundary, OSPF is required to propagate SID values between areas. When OSPF advertises the prefix reachability between areas, it checks if the SID has been advertised for the prefix. In a typical case, the SID value come from the router, which contributes to the best path to the prefix in the source area. In this case, OSPF uses such SID and advertises it between the areas. If the SID value is not advertised by the router which contributes to the best path inside the area, OSPF will use the SID value coming from any other router inside the source area.

Segment Routing Global Range Changes

OSPF advertises it's segment routing capability in terms of advertising the SID/Label Range TLV. In OSPFv2, SID/Label Range TLV is a carried in Router Information LSA.

The segment routing global range configuration will be under the "segment-routing mpls" configuration. When the OSPF process comes, it will get the global range values from segment-routing and subsequent changes should be propagated to it.

When OSPF segment routing is configured, OSPF must request an interaction with the segment routing module before OSPF segment routing operational state can be enabled. If the SRGB range is not created, OSPF will not be enabled. When an SRGB change event occurs, OSPF makes the corresponding changes in it's sub-block entries.

Conflict Handling of SID Entries

In an ideal situation, each prefix should have unique SID entries assigned.

When there is a conflict between the SID entries and the associated prefix entries use any of the following methods to resolve the conflict:

- Multiple SIDs for a single prefix - If the same prefix is advertised by multiple sources with different SIDs, OSPF will install the unlabeled path for the prefix. The OSPF takes into consideration only those SIDs that are from reachable routers and ignores those from unreachable routers. When multiple SIDs are advertised for a prefix, which is considered as a conflict, no SID will be advertised to the attached-areas for the prefix. Similar logic will be used when propagating the inter-area prefixes between the backbone and the non-backbone areas.

- Out of Range SID - For SIDs that do not fit in our SID range, labels are not used while updating the RIB.

MPLS Forwarding on an Interface

MPLS forwarding must be enabled before segment routing can use an interface. OSPF is responsible for enabling MPLS forwarding on an interface.

When segment routing is enabled for a OSPF topology, or OSPF segment routing operational state is enabled, it enables MPLS for any interface on which the OSPF topology is active. Similarly, when segment routing is disabled for a OSPF topology, it disables the MPLS forwarding on all interfaces for that topology.

MPLS forwarding is not supported on an interface which terminates at the IPIP/GRE tunnel.

Configuring Segment Routing with OSPFv2

Configure segment routing with OSPFv2 protocol.

Before you begin

Confirm that the following conditions are met before configuring segment routing with OSPFv2:

- The OSPFv2 feature is enabled.
- The segment-routing feature is enabled.
- Segment routing is enabled under OSPF.

Procedure

	Command or Action	Purpose

Step 1	**configure terminal**
	Example:
	`switch# configure terminal`
	`switch(config)#`
		Enters global configuration mode.

Step 2	[**no**]**router ospf** *process* **Example:** `switch(config)# router ospf test`	Enables the OSPF mode.
Step 3	**segment-routing** **Example:** `switch(config-router)# segment-routing mpls`	Configures the segment routing functionality under OSPF.

Configuring Segment Routing on OSPF Network- Area Level

Before you begin

Before you configure segment routing on OSPF network, OSPF must be enabled on your network.

Procedure

	Command or Action	Purpose
Step 1	**router ospf process** **Example:** `switch(config)# router ospf test`	Enables the OSPF mode.
Step 2	**area** <*area id*> **segment-routing** [**mpls** | **disable**] **Example:** `switch(config-router)# area 1 segment-routing mpls`	Configures segment routing mpls mode in a specific area.
Step 3	[**no**]**area** <*area id*> **segment-routing** [**mpls** | **disable**] **Example:** `switch(config-router)#area 1 segment-routing disable`	Disables segment routing mpls mode for the specified area.
Step 4	**show ip ospf** *process* **segment-routing** **Example:** `switch(config-router)# show ip ospf test segment-routing`	Shows the output for configuring segment routing under OSPF.

Configuring Prefix-SID for OSPF

This task explains how to configure prefix segment identifier (SID) index under each interface.

Before you begin

Segment routing must be enabled on the corresponding address family.

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `switch# configure terminal`	Enters global configuration mode.
Step 2	[**no**]**router ospf** *process* **Example:** `switch(config)# router ospf test`	Configures OSPF.
Step 3	**segment-routing** **Example:**	Configures the segment routing functionality under OSPF.

	Command or Action	Purpose

	`switch(config-router)# segment-routing`
	`switch(config-sr)#mpls`
	`switch(config-sr-mpls)#`	

Step 4	**interface loopback** *interface_number* **Example:** `switch(config-sr-mpls)# Interface loopback 0`	Specifies the interface where OSPF is enabled.
Step 5	**ip address 1.1.1.1/32** **Example:** `switch(config-sr-mpls)# ip address 1.1.1.1/32`	Specifies the IP address configured on the ospf interface.
Step 6	**ip router ospf 1 area 0** **Example:** `switch(config-sr-mpls)# ip router ospf 1 area 0`	Specifies the OSPF enabled on the interface in area.

Step 7	**segment-routing**
	Example:
	`switch(config-router)#segment-routing`
	`(config-sr)#mpls`
		Configures prefix-sid mapping under SR module.

Step 8	**connected-prefix-sid-map**
	Example:
	`switch(config-sr-mpls)# connected-prefix-sid-map`
	`switch(config-sr-mpls-conn-pfxsid)#`
		Configures the prefix SID mapping under the segment routing module.

Step 9	**address-family ipv4**
	Example:
	`switch(config-sr-mpls-conn-pfxsid)# address-family ipv4`
	`switch(config-sr-mpls-conn-pfxsid-af)#`
		Specifies the IPv4 address family configured on the OSPF interface.

Step 10	**1.1.1.1/32 index 10** **Example:** `switch(config-sr-mpls-conn-af)# 1.1.1.1/32 index 10`	Associates SID 10 with the address 1.1.1.1/32.
Step 11	**exit** **Example:** `switch(config-sr-mpls-conn-af)# exit`	Exits segment routing mode and returns to the configuration terminal mode.

Configuring Prefix Attribute N-flag-clear

OSPF advertises prefix SIDs via Extended Prefix TLV in its opaque LSAs. It carries flags for the prefix and one of them is N flag (Node) indicating that any traffic sent along to the prefix is destined to the router originating the LSA. This flag typically marks host routes of router's loopback.

Procedure

	Command or Action	Purpose

Step 1	**configure terminal**
	Example:
	`switch# configure terminal`
	`switch(config)#`
		Enters global configuration mode.

Step 2	**interface loopback3** **Example:** `switch(config)# interface loopback3`	Specifies the interface loopback.
Step 3	**ip ospf prefix-attributes n-flag-clear** **Example:** `switch#(config-if)# ip ospf prefix-attributes n-flag-clear`	Clears the prefix N-flag.

Configuration Examples for Prefix SID for OSPF

This example shows the configuration for prefix SID for OSPF.

```
Router ospf 10
     Segment-routing mpls
Interface loop 0
    Ip address 1.1.1.1/32
    Ip router ospf 10 area 0
Segment-routing
   Mpls
      connected-prefix-sid-m
          address-family ipv4
              1.1.1.1/32 index 10
```

Configuring Segment Routing for Traffic Engineering

About Segment Routing for Traffic Engineering

Segment routing for traffic engineering (SR-TE) takes place through a tunnel between a source and destination pair. Segment routing for traffic engineering uses the concept of source routing, where the source calculates the path and encodes it in the packet header as a segment. A Traffic Engineered (TE) tunnel is a container of TE LSPs instantiated between the tunnel ingress and the tunnel destination. A TE tunnel can instantiate one or more SR-TE LSPs that are associated with the same tunnel.

With segment routing for traffic engineering (SR-TE), the network no longer needs to maintain a per-application and per-flow state. Instead, it simply obeys the forwarding instructions provided in the packet.

SR-TE utilizes network bandwidth more effectively than traditional MPLS-TE networks by using ECMP at every segment level. It uses a single intelligent source and relieves remaining routers from the task of calculating the required path through the network.

SR-TE Policies

Segment routing for traffic engineering (SR-TE) uses a "policy" to steer traffic through the network. A SR-TE policy is a container that includes sets of segments or labels. This list of segments can be provisioned by an operator, a stateful PCE. The head-end imposes the corresponding MPLS label stack on a traffic flow to be carried over the SR-TE policy. Each transit node along the SR-TE policy path uses the incoming top label to select the next-hop, pop or swap the label, and forward the packet to the next node with the remainder of the label stack, until the packet reaches the ultimate destination.

A SR-TE policy is uniquely identified by a tuple (color, end-point). A color is represented as a 32-bit number and an end-point is an IPv4 . Every SR-TE policy has a color value. Every policy between the same node pairs requires a unique color value. Multiple SR-TE policies can be created between the same two endpoints by choosing different colors for the policies.

Cisco Nexus 9000 Series switches support the following two types of SR-TE policies:

- Dynamic SR-TE Policy - When you configure dynamic path preference under the SR-TE policy configuration or an on-demand color configuration, the path computation engine (PCE) calculates the path to the destination address. Dynamic path calculation at PCE results in a list of segments/labels that gets applied to the head-end SR-TE policy, hence the traffic gets routed through the network by hitting the segments that the SR-TE policy holds.
- Explicit SR-TE Policy - An explicit path is a list of labels, each representing a node or link in the explicit path. This feature is enabled through the **explicit-path** command that allows you to create an explicit path and enter a configuration submode for specifying the path.

SR-TE Policy Paths

A SR-TE policy path is a list of segments that specifies the path, called a segment ID (SID) list. Every SR-TE policy consists of one or more candidate paths, which can be either a dynamic or an explicit path. The SR-TE policy instantiates a single path and the selected path is the preferred valid candidate path.

You can also add on-demand color with dynamic path option and explicit policy configuration with an explicit path option for the same color and endpoint. In this case, a single policy is created on the head-end and the path with the highest preference number configured is used for forwarding traffic.

The following two methods are used to compute the SR-TE policy path:.

- Dynamic Path - When you specify the dynamic PCEP option while configuring the path preference under an on-demand color configuration or a policy configuration, the path computation is delegated to a path computation engine(PCE).
- Explicit Path - This path is an explicitly specified SID-list or a set of SID-lists.

Affinity and Disjoint Constraints

Affinity Constraints - You can assign attributes to a link which gets advertised to path computation engine (PCE). SRTE process hosts the affinity-map and interface level configurations. Routing protocol(IGP) will register for interface updates and SRTE will notify IGP with interface updates. IGP tlvs will be passed to BGP to advertise it to external peers. There are three types of affinity constraints:

- exclude-any: specifies that links that have any of the specified affinity colors must not be traversed by the path.
- include-any: specifies that only links that have any of the specified affinity colors must be traversed by the path. Thus, links that do not have any of the specified affinity colors must not be used.
- include-all: specifies that only links that have all of the specified affinity colors must be traversed by the path. Thus, links that do not have all of the specified affinity colors must not be used.

Disjoint Constraints - You can assign disjoint constraints to the SR-TE policies which gets advertised to the PCE. The PCE then provides the disjoint path for the policies that share the same association group ID and the disjoint disjointness type.

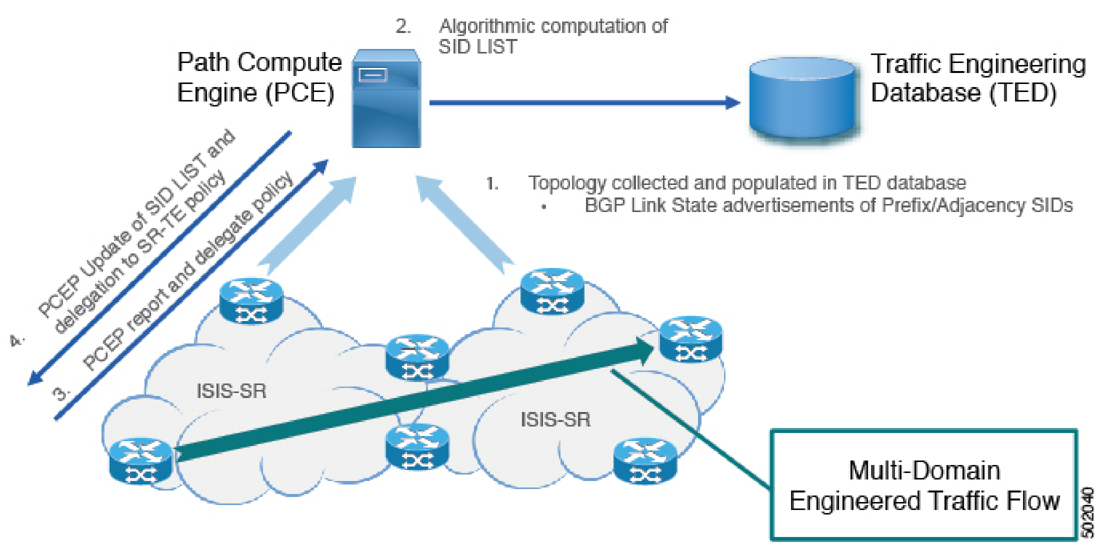
Cisco NX-OS Release 9.3(1) supports the following disjoint path levels :

- Link – The paths transit different links (but may transit same nodes).
- Node disjointness – The paths transit different links but may transit same node.

Segment Routing On Demand Next Hop

On-Demand Next hop (ODN) leverages upon BGP Dynamic SR-TE capabilities and adds the path computation (PCE) ability to find and download the end to end path based on the requirements. ODN triggers an SR-TE auto-tunnel based on the defined BGP policy. As shown in the following figure, an end-to-end path between ToR1 and AC1 can be established from both ends based on IGP Metric. The work-flow for ODN is summarized as follows:

Figure 8: ODN Operation



Guidelines and Limitations for SR-TE

SR-TE has the following guidelines and limitations:

- SR-TE ODN for both, IPv4 and IPv6 overlay is supported.
- SR-TE ODN is supported only with IS-IS underlay.
- Forwarding does not support routes with recursive next hops, where the recursive next hop resolves to a route with a binding SID.
- Forwarding does not support mixing paths with binding labels and paths without binding labels for the same route.
- The affinity and disjoint constraints are applicable only to those SR-TE policies that have a dynamic PCEP option.
- XTC supports only two policies with disjointness in the same group.
- When configuring the SR-TE affinity interfaces, the interface range is not supported.
- A preference cannot have both, the dynamic PCEP and the explicit segment lists configured together for the same preference.
- Only one preference can have a dynamic PCEP option per policy.
- For explicit policy, when configuring ECMP paths under same preference, if the first hop (NHLFE) is same for both the ECMP paths, ULIB will only install one path in switching. This occurs because both the ECMP paths create the same SRTE FEC as the NHLFE is same for both.
- In Cisco NX-OS Release 9.3(1), unprotected mode with affinity configuration is not supported by PCE (XTC).
- Beginning with Cisco NX-OS Release 9.3(3), SR-TE ODN, policies, policy paths, and the affinity and disjoint constraints are supported on Cisco Nexus 9364C-GX, Cisco Nexus 9316D-GX, and Cisco Nexus 93600CD-GX switches.

Configuring SR-TE

You can configure segment routing for traffic engineering.

Before you begin

You must ensure that the mpls segment routing feature is enabled.

Procedure

	Command or Action	Purpose
Step 1	**configure terminal**	Enters global configuration mode.
Step 2	**segment-routing**	Enters the segment-routing mode
Step 3	**traffic-engineering**	Enters the traffic engineering mode.
Step 4	**encapsulation mpls source ipv4** *tunnel_ip_address*	Configures the source address for the SR-TE Tunnel.
Step 5	**pcc**	Enters the PCC mode.

	Command or Action	Purpose
Step 6	**source-address ipv4** *pcc_source_address*	Configure source address for the PCC
Step 7	**pce-address ipv4 pce_source_address** *precedence* **num**	Configure IP address of the PCE. The lowest numbered PCE will take precedence, and the other(s) be used as a backup.
Step 8	**on-demand color** *color_num*	Enters the on-demand mode to configure the color.
Step 9	**candidate-paths**	Specifies the candidate paths of the policy.
Step 10	**preference** *preference_number*	Specifies the preference of the candidate path.
Step 11	**dynamic**	Specifies the path option.
Step 12	**pcep**	Specifies the path computation that needs to be done from the PCE.

Configuring Affinity Constraints

You can configure the affinity constraints to the SR-TE policy.

Before you begin

You must ensure that the mpls segment routing feature is enabled.

Procedure

	Command or Action	Purpose

Step 1	**configure terminal**
	Example:
	`switch# configure terminal`
	`switch(config)#`
		Enters global configuration mode.

Step 2	**segment-routing**
	Example:
	`switch(config)# segment-routing`
	`switch(config-sr)#`
		Enables the MPLS segment routing functionality.

Step 3	**traffic-engineering**
	Example:
	`switch(config-sr)# traffic-engineering`
	`switch(config-sr-te)#`
		Enters the traffic engineering mode.

Step 4	**pcc**	Enters the PCC mode.
Step 5	**source-address ipv4** *pcc_source_address*	Configure source address for the PCC

	Command or Action	Purpose
Step 6	**pce-address ipv4 pce_source_address** *precedence* **num**	Configure IP address of the PCE. The lowest numbered PCE takes precedence and the other(s) are used as a backup.

Step 7	**affinity-map**
	Example:
	`switch(config-sr-te)#affinity-map`
	`switch(config-sr-te-affmap)#`
		Configures the affinity-map configuration mode.

Step 8	**color** *name* **bit-position** *position*
	Example:
	`switch(config-sr-te-affmap)# color red bit-position 2`
	`switch(config-sr-te-affmap)#`
		Configures a mapping of the user-defined name to a specific bit position in the affinity bit-map.

Step 9	**interface** *interface-name*
	Example:
	`Enter SRTE interface config mode`
	`switch(config-sr-te-if)#interface eth1/1`
	`switch(config-sr-te-if)#`
		Specifies the name of the interface. This is the affinity mapping name which refers to the specific bit in the affinity bitmap.

Step 10	**affinity**
	Example:
	`switch(config-sr-te-if)# affinity`
	`switch(config-sr-te-if-aff)#`
	`switch(config-sr-te-if-aff)# color red`
	`switch(config-sr-te-if-aff)#`
		Adds the affinity color to the interface.

Step 11	**policy** *name* | **on-demand color** *color_num*
	Example:
	`switch(config-sr-te)# on-demand color 211`
	or
	`switch(config-sr-te-color)# policy test_policy`
		Configures the policy.

Step 12	**color** *color end-point address* **Example:** `switch(config-sr-te-pol)#color 200 endpoint 2.2.2.2`	Configures the color and the end point of the policy. This is required when you are configuring the policy using the "policy name" config mode.

Step 13	**candidate-path**
	Example:
	`switch(config-sr-te-color)# candidate-paths`
	`switch(cfg-cndpath)#`
		Specifies the candidate paths for the policy.

	Command or Action	Purpose

Step 14	**preference** *preference_number*
	Example:
	`switch(cfg-cndpath)# preference 100`
	`switch(cfg-pref)#`
		Specifies the preference of the candidate path.

Step 15	**dynamic**
	Example:
	`switch(cfg-pref)# dynamic`
	`switch(cfg-dyn)#`
		Specifies the path option.

Step 16	**pcep**
	Example:
	`switch(cfg-dyn)# pcep`
	`switch(cfg-dyn)#`
		Specifies that the headend uses PCEP to request the PCE to compute a path from itself to the segment routing's policy's end point.

Step 17	**constraints**
	Example:
	`switch(cfg-dyn)# constraints`
	`switch(cfg-constraints)#`
		Enters the candidate path preference constraint mode.

Step 18	**affinity**
	Example:
	`switch(cfg-constraints)# affinity`
	`switch(cfg-const-aff)#`
		Specifies the affinity constraints of the policy.

Step 19	**exclude-any |include-all | include-any**
	Example:
	`switch(cfg-const-aff)# include-any`
	`switch(cfg-aff-inclany)#`
		Specifies the affinity constraint type. The following affinity types are available:
		• exclude-any - specifies that links that have any of the specified affinity colors must not be traversed by the path.
		• include-any - specifies that only links that have any of the specified affinity colors must be traversed by the path.
		• include-all - specifies that only links that have all of the specified affinity colors must be traversed by the path.

Step 20	**color** *color_name*
	Example:
	`switch(cfg-aff-inclany)# color blue`
	`switch(cfg-aff-inclany)#`
		Specifies the affinity color definition.

Configuring Disjoint Paths

You can configure disjoint path constraints to the SR-TE policy.

Before you begin

You must ensure that the mpls segment routing feature is enabled.

Procedure

	Command or Action	Purpose

Step 1	**configure terminal**
	Example:
	`switch# configure terminal`
	`switch(config)#`
		Enters global configuration mode.

Step 2	**segment-routing**
	Example:
	`switch(config)# segment-routing`
	`switch(config-sr)#`
		Enables the MPLS segment routing functionality.

Step 3	**traffic-engineering**
	Example:
	`switch(config-sr)# traffic-engineering`
	`switch(config-sr-te)#`
		Enters the traffic engineering mode.

Step 4	**pcc**	Enters the PCC mode.
Step 5	**source-address ipv4** *pcc_source_address*	Configure source address for the PCC
Step 6	**pce-address ipv4 pce_source_address** *precedence* **num**	Configure IP address of the PCE. The lowest numbered PCE takes precedence and the other(s) are used as a backup.

Step 7	**policy** *name* | **on-demand color** *color_num*
	Example:
	`switch(config-sr-te)# on-demand color 211`
	or
	`switch(config-sr-te-color)# policy test_policy`
		Configures the policy.

Step 8	**color** *color end-point address* **Example:** `switch2(config-sr-te-pol)# color 200 endpoint 2.2.2.2`	Configures the color and the end point of the policy. This is required when you are configuring the policy using the “policy name” config mode.

Step 9	**candidate-path**
	Example:
	`switch(config-sr-te-color)# candidate-paths`
	`switch(cfg-cndpath)#`
		Specifies the candidate-paths for the policy

	Command or Action	Purpose

Step 10	**preference** *preference_number*
	Example:
	`switch(cfg-cndpath)# preference 100`
	`switch(cfg-pref)#`
		Specifies the preference of the candidate path.

Step 11	**dynamic**
	Example:
	`switch(cfg-pref)# dynamic`
	`switch(cfg-dyn)#`
		Specifies the path option.

Step 12	**pcep**
	Example:
	`switch(cfg-dyn)# pcep`
	`switch(cfg-dyn)#`
		Specifies that the headend uses PCEP to request the PCE to compute a path from itself to the segment routing's policy's end point.

Step 13	**constraints**
	Example:
	`switch(cfg-dyn)# constraints`
	`switch(cfg-constraints)#`
		Enters the candidate path preference constraint mode.

Step 14	**association-group**
	Example:
	`switch(cfg-constraints)# association-group`
	`switch(cfg-assoc)#`
		Specifies the association group type.

Step 15	**disjoint**
	Example:
	`switch(cfg-assoc)# disjoint`
	`switch(cfg-disj)#`
		Specifies the path that belongs to the disjointness association group.

Step 16	**type | link | node** **Example:** `switch(config-if)#type link`	Specifies the disjointness group type.
Step 17	**id** *number* **Example:** `switch(config-if)#id 1`	Specifies the identifier of the association-group.

Configuration Examples for SR-TE

The examples in this section show affinity and disjoint configurations.

This example shows the mappings of a user defined name to an administrative group.

```
segment-routing
 traffic-eng
  affinity-map
```

```
   color green bit-position 0
   color blue bit-position 2
   color red bit-position 3
```

This example shows the affinity link colors red and green for the adjacency on eth1/1 and affinity link color green for the adjacency on eth1/2.

```
segment-routing
 traffic-eng
  interface eth1/1
   affinity
    color red
    color green
  !
  interface eth1/2
   affinity
    color green
```

This examples shows the affinity constraints for the policy.

```
segment-routing
  traffic-engineering
    affinity-map
      color blue bit-position 0
      color red  bit-position 1
    on-demand color 10
      candidate-paths
        preference 100
          dynamic
            pcep
          constraints
            affinity
              [include-any|include-all|exclude-any]
                color <col_name>
                color <col_name>
    policy new_policy
      color 201 endpoint 2.2.2.0
      candidate-paths
        preference 200
          dynamic
            pcep
          constraints
            affinity
              include-all
                color red
```

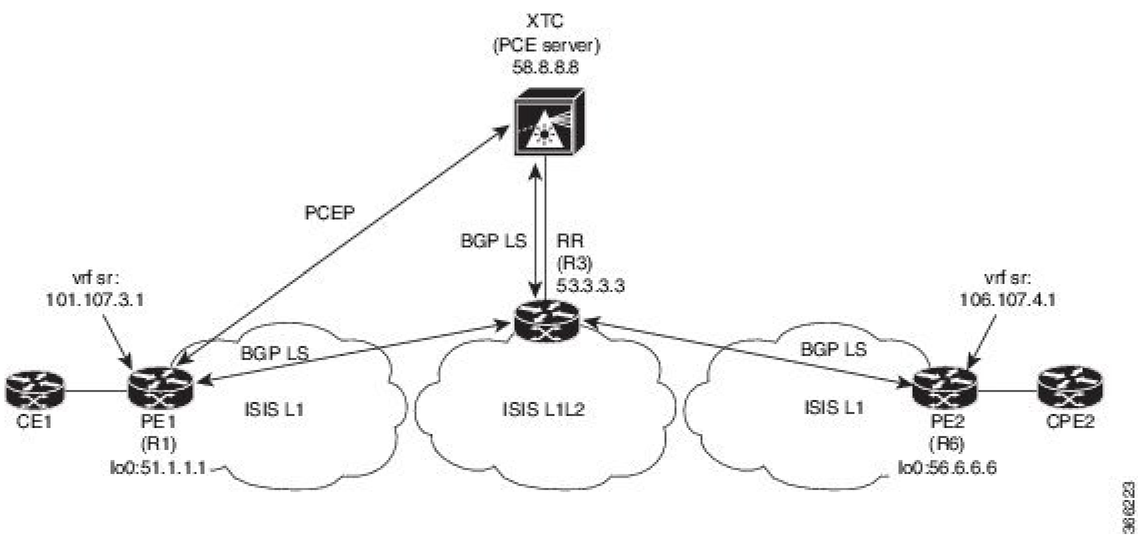
This examples shows the disjoint constraints for the policy.

```
segment-routing
 traffic-eng
  on-demand color 99
   candidate-paths
     preference 100
       dynamic
         pcep
       constraints
         association-group
           disjoint
             type link
             id 1
```

Configuration Example for an SR-TE ODN - Use Case

Perform the following steps to configure ODN for SR-TE. The following figure is used as a reference to explain the configuration steps.

Figure 9: Reference Topology



1. Configure all links with IS-IS point-to-point session from PE1 to PE2. Also, configure the domains as per the above topology.
2. Enable "distribute link-state" for IS-IS session on R1, R3, and R6.

```
router isis 1
  net 31.0000.0000.0000.712a.00
  log-adjacency-changes
  distribute link-state
  address-family ipv4 unicast
    bfd
    segment-routing mpls
    maximum-paths 32
    advertise interface loopback0
```

3. Configure the router R1 (headend) and R6 (tailend) with a VRF interface.

VRF configuration on R1:

```
interface Ethernet1/49.101
encapsulation dot1q 201
  vrf member sr
  ip address 101.10.1.1/24
  no shutdown

vrf context sr
  rd auto
  address-family ipv4 unicast
    route-target import 101:101
    route-target import 101:101 evpn
    route-target export 101:101
    route-target export 101:101 evpn
router bgp 6500
  vrf sr
    bestpath as-path multipath-relax
```

```
    address-family ipv4 unicast
      advertise l2vpn evpn
```

4. Tags VRF prefix with BGP community on R6 (tailend).

```
route-map color1001 permit 10
  set extcommunity color 1001
```

5. Enable BGP on R6 (tailend) and R1 (headend) to advertise and receive VRF SR prefix and match on community set on R6 (tailend).

   R6 < EVPN > R3 < EVPN > R1

   BGP Configuration R6:

```
router bgp 6500
  address-family ipv4 unicast
     allocate-label all
  neighbor 53.3.3.3
    remote-as 6500
    log-neighbor-changes
    update-source loopback0
    address-family l2vpn evpn
      send-community extended
     route-map Color1001 out
      encapsulation mpls
```

   BGP Configuration R1:

```
router bgp 6500
  address-family ipv4 unicast
     allocate-label all
  neighbor 53.3.3.3
    remote-as 6500
    log-neighbor-changes
    update-source loopback0
    address-family l2vpn evpn
      send-community extended
       encapsulation mpls
```

6. Enable BGP configuration on R3 and BGP LS with XTC on R1, R3.abd

   BGP Configuration R3:

```
router bgp 6500
  router-id 2.20.1.2
address-family ipv4 unicast
allocate-label all
address-family l2vpn evpn
retain route-target all
  neighbor 56.6.6.6
    remote-as 6500
    log-neighbor-changes
    update-source loopback0
    address-family l2vpn evpn
      send-community extended
       route-reflector-client
       route-map NH_UNCHANGED out
      encapsulation mpls
  neighbor 51.1.1.1
    remote-as 6500
    log-neighbor-changes
    update-source loopback0
    address-family l2vpn evpn
      send-community extended
```

```
      route-reflector-client
      route-map NH_UNCHANGED out
      encapsulation mpls
neighbor 58.8.8.8
    remote-as 6500
    log-neighbor-changes
    update-source loopback0
    address-family link-state

route-map NH_UNCHANGED permit 10
  set ip next-hop unchanged
```

BGP Configuration R1:

```
router bgp 6500
neighbor 58.8.8.8
              remote-as 6500
               log-neighbor-changes
               update-source loopback0
               address-family link-state
```

BGP Configuration R6:

```
outer bgp 6500
   neighbor 58.8.8.8
    remote-as 6500
       log-neighbor-changes
       update-source loopback0
       address-family link-state
```

7. Enable PCE and SR-TE tunnel configurations on R1.

```
segment-routing
  traffic-engineering
    pcc
      source-address ipv4 51.1.1.1
      pce-address ipv4 58.8.8.8
    on-demand color 1001
      metric-type igp
```

Configuring Egress Peer Engineering with Segment Routing

BGP Prefix SID

In order to support segment routing, BGP requires the ability to advertise a segment identifier (SID) for a BGP prefix. A BGP prefix SID is always global within the segment routing BGP domain and identifies an instruction to forward the packet over the ECMP-aware best path computed by BGP to the related prefix. The BGP prefix SID identifies the BGP prefix segment.

Adjacency SID

The adjacency segment Identifier (SID) is a local label that points to a specific interface and a next hop out of that interface. No specific configuration is required to enable adjacency SIDs. Once segment routing is enabled over BGP for an address family, for any interface that BGP runs over, the address family automatically allocates an adjacency SID toward every neighbor out of that interface.

High Availability for Segment Routing

In-service software upgrades (ISSUs) are minimally supported with BGP graceful restart. All states (including the segment routing state) must be relearned from the BGP router's peers. During the graceful restart period, the previously learned route and label state are retained.

Overview of BGP Egress Peer Engineering With Segment Routing

Cisco Nexus 9000 Series switches are often deployed in massive scale data centers (MSDCs). In such environments, there is a requirement to support BGP Egress Peer Engineering (EPE) with Segment Routing (SR).

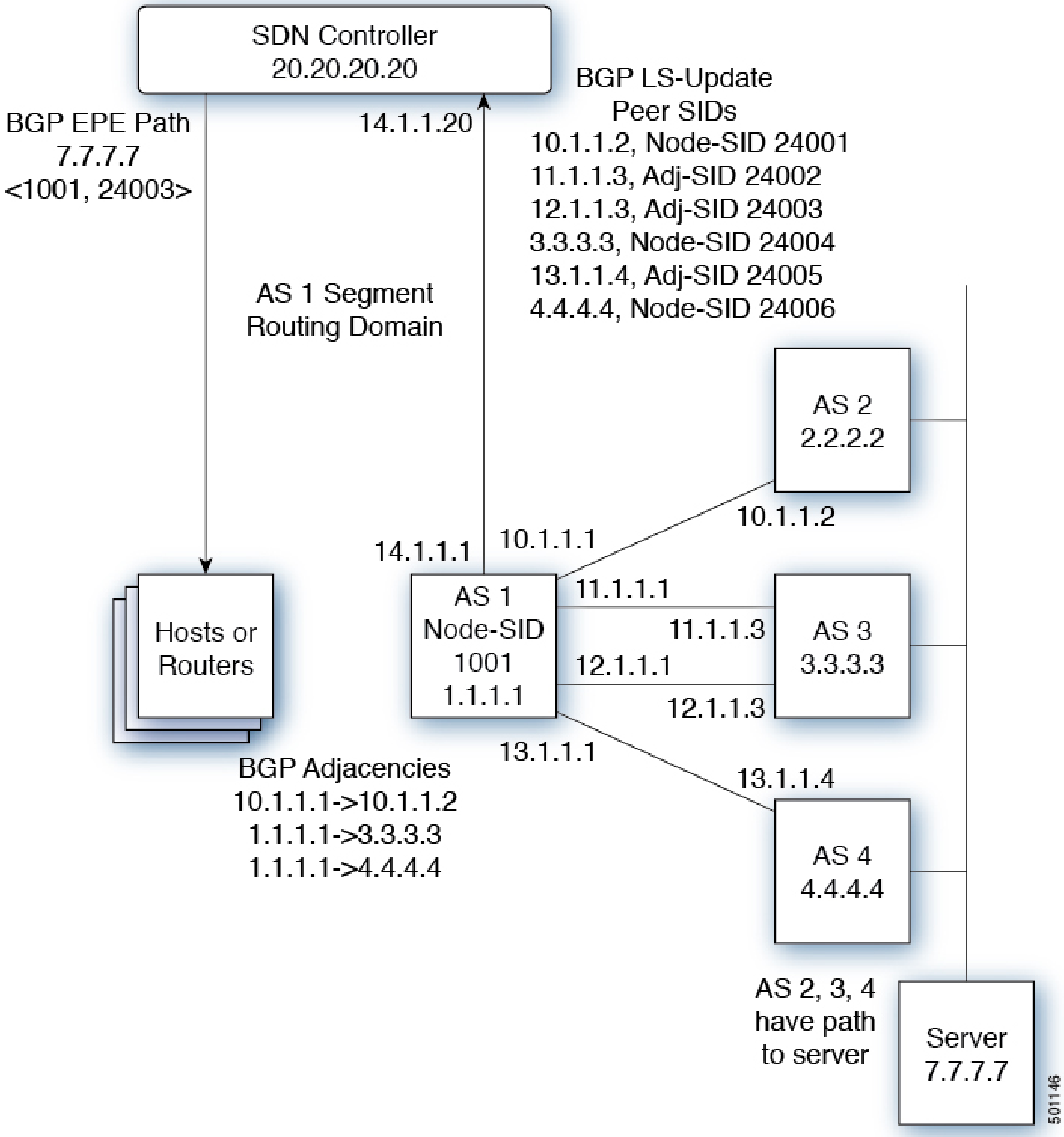
Segment Routing (SR) leverages source routing. A node steers a packet through a controlled set of instructions, known as segments, by prepending the packet with an SR header. A segment can represent any topological or service-based instruction. SR allows steering a flow through any topological path or any service chain while maintaining per-flow state only at the ingress node of the SR domain. For this feature, the Segment Routing architecture is applied directly to the MPLS data plane.

In order to support Segment Routing, BGP requires the ability to advertise a Segment Identifier (SID) for a BGP prefix. A BGP prefix is always global within the SR or BGP domain and it identifies an instruction to forward the packet over the ECMP-aware best-path that is computed by BGP to the related prefix. The BGP prefix is the identifier of the BGP prefix segment.

The SR-based Egress Peer Engineering (EPE) solution allows a centralized (SDN) controller to program any egress peer policy at ingress border routers or at hosts within the domain.

In the following example, all three routers run iBGP and they advertise NRLI to one another. The routers also advertise their loopback as the next-hop and it is recursively resolved. This provides an ECMP between the routers as displayed in the illustration.

Figure 10: Example of Egress Peer Engineering



The SDN controller receives the Segment IDs from the egress router 1.1.1.1 for each of its peers and adjacencies. It can then intelligently advertise the exit points to the other routers and the hosts within the controller's routing domain. As displayed in the illustration, the BGP Network Layer Reachability Information (NLRI) contains both the Node-SID to Router 1.1.1.1 and the Peer-Adjacency-SID 24003 indicating that the traffic to 7.7.7.7 should egress over the link 12.1.1.1->12.1.1.3.

Guidelines and Limitations for BGP Egress Peer Engineering

BGP Egress Peer Engineering has the following guidelines and limitations:

- BGP Egress Peer Engineering is only supported for IPv4 BGP peers. IPv6 BGP peers are not supported.
- BGP Egress Peer Engineering is only supported in the default VPN Routing and Forwarding (VRF) instance.
- Any number of Egress Peer Engineering (EPE) peers may be added to an EPE peer set. However, the installed resilient per-CE FEC is limited to 32 peers.
- A given BGP neighbor can only be a member of a single peer-set. Peer-sets are configured. Multiple peer-sets are not supported. An optional **peer-set** name may be specified to add neighbor to a peer-set. The corresponding RPC FEC load-balances the traffic across all the peers in the peer-set. The peer-set name is a string that is a maximum length of 63 characters (64 NULL terminated). This length is consistent with the NX-OS policy name lengths. A peer can only be a member of a single peer-set.
- Adjacencies for a given peer are not separately assignable to different peer-sets.
- Beginning with Cisco NX-OS Release 9.3(3), BGP Egress Peer Engineering is supported on Cisco Nexus 9300-GX platform switches.

Configuring Neighbor Egress Peer Engineering Using BGP

With the introduction of RFC 7752 and draft-ietf-idr-bgpls-segment-routing-epe, you can configure Egress Engineering. The feature is valid only for external BGP neighbors and it is not configured by default. Egress Engineering uses RFC 7752 encoding.

Before you begin

- You must enable BGP.
- After an upgrade from Release 7.0(3)I3(1) or Release 7.0(3)I4(1), configure the TCAM region before configuring Egress Peer Engineering (EPE) on Cisco Nexus 9000 Series switches using the following commands:
  1. switch# **hardware access-list tcam region vpc-convergence 0**
  2. switch# **hardware access-list tcam region racl 0**
  3. switch# **hardware access-list tcam region mpls 256 double-wide**
- Save the configuration and reload the switch.

For more information, see the Using Templates to Configure ACL TCAM Region Sizes and Configuring ACL TCAM Region Sizes sections in the *Cisco Nexus 9000 Series NX-OS Security Configuration Guide*.

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:**	Enters global configuration mode.

	Command or Action	Purpose
	`switch# configure terminal` `switch(config)#`	
Step 2	**router bgp** <*bgp autonomous number*>	Specifies the autonomous router BGP number.
Step 3	**neighbor** <*IP address*>	Configures the IP address for the neighbor.

Step 4	[**no**|**default**] **egress-engineering** [**peer-set** *peer-set-name*]
	Example:
	`switch(config)# router bgp 1`
	`switch(config-router)# neighbor 4.4.4.4`
	`switch(config-router)# egress-engineering peer-set NewPeer`
		Specifies whether a Peer-Node-SID is allocated for the neighbor and it is advertised in an instance of a BGP Link-State (BGP-LS) address family Link NLRI. If the neighbor is a multi-hop neighbor, a BGP-LS Link NLRI instance is also advertised for each Equal-Cost-MultiPath (ECMP) path to the neighbor and it includes a unique Peer-Adj-SID.
		Optionally, you can add the neighbor to a peer-set. The Peer-Set-SID is also advertised in the BGP-LS Link NLRI in the same instance as the Peer-Node-SID. BGP Link-State NLRI is advertised to all neighbors with the link-state address family configured.
		See RFC 7752 and draft-ietf-idr-bgpls-segment-routing-epe-05 for more information on EPE.

Configuration Example for Egress Peer Engineering

See the Egress Peer Engineering sample configuration for the BGP speaker 1.1.1.1. Note that the neighbor 20.20.20.20 is the SDN controller.

```
 hostname epe-as-1
install feature-set mpls
feature-set mpls

feature telnet
feature bash-shell
feature scp-server
feature bgp
feature mpls segment-routing

segment-routing mpls
vlan 1

vrf context management
  ip route 0.0.0.0/0 10.30.97.1
  ip route 0.0.0.0/0 10.30.108.1

interface Ethernet1/1
  no switchport
  ip address 10.1.1.1/24
  no shutdown

interface Ethernet1/2
  no switchport
```

```
  ip address 11.1.1.1/24
  no shutdown

interface Ethernet1/3
  no switchport
  ip address 12.1.1.1/24
  no shutdown

interface Ethernet1/4
  no switchport
  ip address 13.1.1.1/24
  no shutdown

interface Ethernet1/5
  no switchport
  ip address 14.1.1.1/24
  no shutdown

interface mgmt0
  ip address dhcp
  vrf member management


interface loopback1
  ip address 1.1.1.1/32
line console

line vty
ip route 2.2.2.2/32 10.1.1.2
ip route 3.3.3.3/32 11.1.1.3
ip route 3.3.3.3/32  12.1.1.3
ip route 4.4.4.4/32  13.1.1.4
ip route 20.20.20.20/32 14.1.1.20

router bgp 1
  address-family ipv4 unicast
  address-family link-state
 neighbor 10.1.1.2
    remote-as 2
    address-family ipv4
    egress-engineering
 neighbor 3.3.3.3
   remote-as 3
   address-family ipv4
   update-source loopback1
   ebgp-multihop 2
   egress-engineering
 neighbor 4.4.4.4
   remote-as 4
   address-family ipv4
   update-source loopback1
   ebgp-multihop 2
   egress-engineering
neighbor 20.20.20.20
   remote-as 1
   address-family link-state
   update-source loopback1
   ebgp-multihop 2
neighbor 124.11.50.5
    bfs
    remote-as 6
    update-source port-channel50.11
    egress-engineering peer-set pset2 <<<<<<<
    address-family ipv4 unicast
```

```
neighbor 124.11.101.2
    bfd
    remote-as 6
    update-source Vlan2401
    egress-engineering
    address-family ipv4 unicast
```

This example shows sample output for the **show bgp internal epe** command.

```
switch# show bgp internal epe
BGP Egress Peer Engineering (EPE) Information:
Link-State Server: Inactive
Link-State Client: Active
Configured EPE Peers: 26
Active EPE Peers: 3
EPE SID State:
RPC SID Peer or Set Assigned
ID Type Set Name ID Label Adj-Info, iod
1 Node 124.1.50.5 1 1600
2 Set pset1 2 1601
3 Node 6.6.6.6 3 1602
4 Node 124.11.50.5 4 1603
5 Set pset2 5 1604
6 Adj 6.6.6.6 6 1605 124.11.50.4->124.11.50.5/0x1600b031, 80
7 Adj 6.6.6.6 7 1606 124.1.50.4->124.1.50.5/0x16000031, 78
EPE Peer-Sets:
IPv4 Peer-Set: pset1, RPC-Set 2, Count 7, SID 1601
Peers: 124.11.116.2 124.11.111.2 124.11.106.2 124.11.101.2
124.11.49.5 124.1.50.5 124.1.49.5
IPv4 Peer-Set: pset2, RPC-Set 5, Count 5, SID 1604
Peers: 124.11.117.2 124.11.112.2 124.11.107.2 124.11.102.2
124.11.50.5
IPv4 Peer-Set: pset3, RPC-Set 0, Count 4, SID unspecified
Peers: 124.11.118.2 124.11.113.2 124.11.108.2 124.11.103.2
IPv4 Peer-Set: pset4, RPC-Set 0, Count 4, SID unspecified
Peers: 124.11.119.2 124.11.114.2 124.11.109.2 124.11.104.2
IPv4 Peer-Set: pset5, RPC-Set 0, Count 4, SID unspecified
Peers: 124.11.120.2 124.11.115.2 124.11.110.2 124.11.105.2
switch#
```

Configuring the BGP Link State Address Family

You can configure the BGP link state address family for a neighbor session with a controller to advertise the corresponding SIDs. You can configure this feature in global configuration mode and neighbor address family configuration mode.

Before you begin

You must enable BGP.

Procedure

	Command or Action	Purpose

Step 1	**configure terminal**
	Example:
	`switch# configure terminal`
	`switch(config)#`
		Enters global configuration mode.

	Command or Action	Purpose
Step 2	**router bgp** <*bgp autonomous number*>	Specifies the autonomous router BGP number.

Step 3	[**no**] **address-family link-state**
	Example:
	`switch(config)# router bgp 64497`
	`switch (config-router af)# address-family link-state`
		Enters address-family interface configuration mode.
		Note This command can also be configured in neighbor address-family configuration mode.

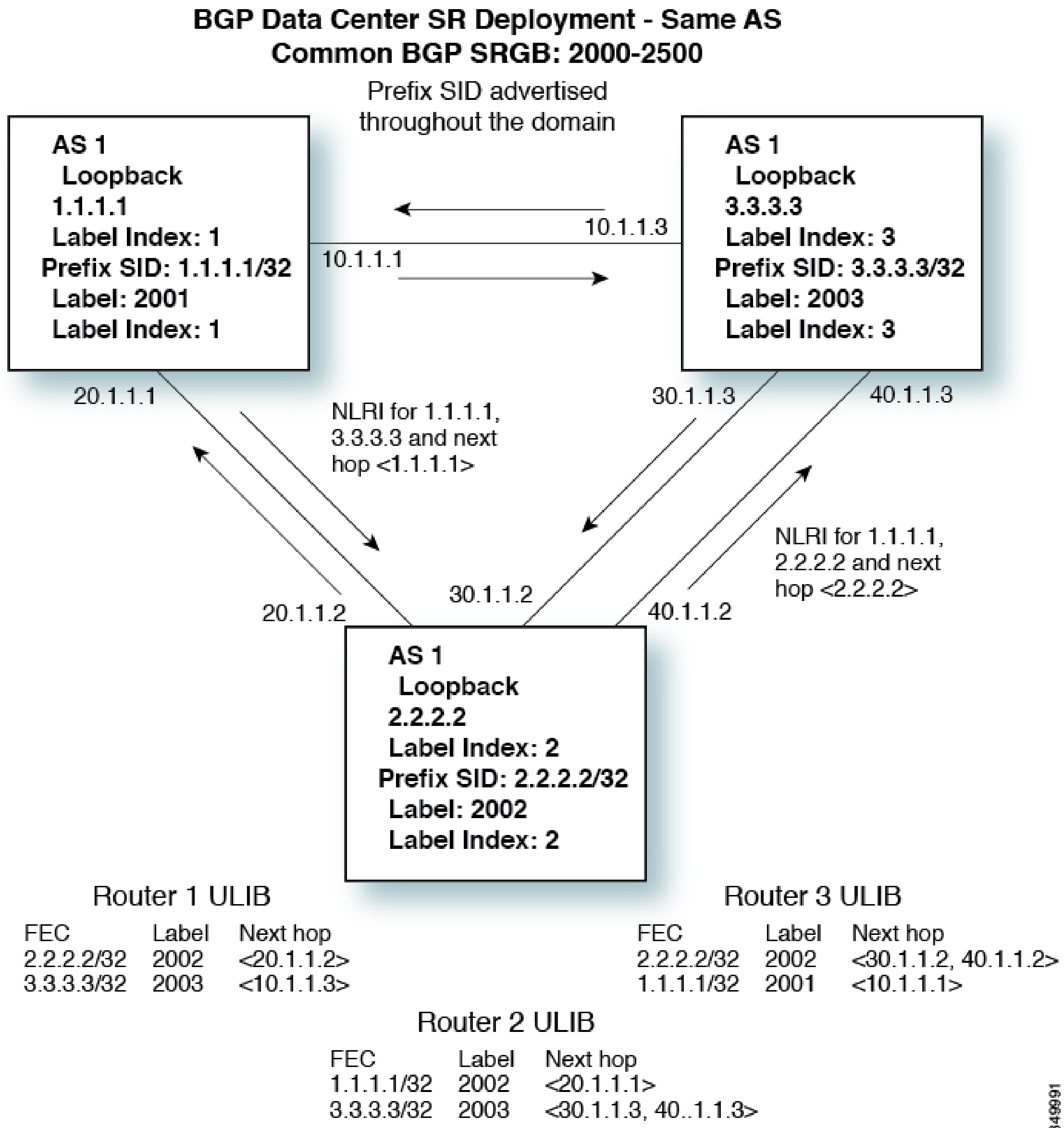
Step 4	**neighbor** <*IP address*>	Configures the IP address for the neighbor.

Step 5	[**no**] **address-family link-state**
	Example:
	`switch(config)#router bgp 1`
	`switch(config-router)#address-family link-state`
	`switch(config-router)#neighbor 20.20.20.20`
	`switch(config-router)#address-family link-state`
		Enters address-family interface configuration mode.
		Note This command can also be configured in neighbor address-family configuration mode.

BGP Prefix SID Deployment Example

In the simple example below, all three routers are running iBGP and advertising Network Layer Reachability Information (NRLI) to one another. The routers are also advertising their loopback interface as the next hop, which provides the ECMP between routers 2.2.2.2 and 3.3.3.3.

Figure 11: BGP Prefix SID Simple Example



Configuring Layer2 EVPN over Segment Routing MPLS

About Layer 2 EVPN

Ethernet VPN (EVPN) is a next generation solution that provides ethernet multipoint services over MPLS networks. EVPN operates in contrast to the existing Virtual Private LAN Service (VPLS) by enabling control-plane based MAC learning in the core. In EVPN, PEs participating in the EVPN instances learn customer MAC routes in control-plane using MP-BGP protocol. Control-plane MAC learning brings several benefits that allow EVPN to address the VPLS shortcomings, including support for multihoming with per-flow load balancing.

In a data center network, the EVPN control plane provides:

- Flexible workload placement that is not restricted with the physical topology of the data center network. Therefore, you can place virtual machines (VM) anywhere within the data center fabric.
- Optimal East-West traffic between servers within and across data centers. East-West traffic between servers, or virtual machines, is achieved by most specific routing at the first hop router. First hop routing is done at the access layer. Host routes must be exchanged to ensure most specific routing to and from servers or hosts. VM mobility is supported by detecting new endpoint attachment when a new MAC address or the IP address is directly connected to the local switch. When the local switch sees the new MAC or the IP address, it signals the new location to rest of the network.
- Segmentation of Layer 2 and Layer 3 traffic, where traffic segmentation is achieved using MPLS encapsulation and the labels (per-BD label and per-VRF labels) act as the segment identifier.

Guidelines and Limitations for Layer 2 EVPN over Segment Routing MPLS

Layer 2 EVPN over segment routing MPLS has the following guidelines and limitations:

- Segment routing Layer 2 EVPN flooding is based on the ingress replication mechanism. MPLS core does not support multicast.
- ARP suppression is not supported.
- Consistency checking on vPC is not supported.
- The same Layer 2 EVI and Layer 3 EVI cannot be configured together.
- Beginning with Cisco NX-OS Release 9.3(1), Layer 2 EVPN is supported on Cisco Nexus 9300-FX2 platform switches.
- Beginning with Cisco NX-OS Release 9.3(5), Layer 2 EVPN over segment routing MPLS is supported on Cisco Nexus 9300-GX and Cisco Nexus 9300-FX3 platform switches.

Configuring Layer 2 EVPN over Segment Routing MPLS

Before you begin

Do the following:

- You must install and enable the MPLS feature set using the **install feature-set mpls** and **feature-set mpls** commands.
- You must enable the MPLS segment routing feature.
- You must enable the nv overlay feature using the **nv overlay** command.
- You must enable EVPN control plane using the **nv overlay evpn** command.

Procedure

	Command or Action	Purpose

Step 1	**configure terminal**
	Example:
	`switch# configure terminal`
	`switch(config)#`
		Enters global configuration mode.

Step 2	**feature bgp** **Example:** `switch(config)#feature bgp`	Enables BGP feature and configurations.
Step 3	**install feature-set mpls** **Example:** `switch(config)#install feature-set mpls`	Enables MPLS configuration commands.
Step 4	**feature-set mpls** **Example:** `switch(config)#install feature-set mpls`	Enables MPLS configuration commands.
Step 5	**feature mpls segment-routing** **Example:** `switch(config)#feature mpls segment-routing`	Enables segment routing configuration commands.
Step 6	**feature mpls evpn** **Example:** `switch(config)#feature mpls evpn`	Enables EVPN over MPLS configuration commands. This command is mutually exclusive with the **feature-nv** CLI command.
Step 7	**feature nv overlay** **Example:** `switch(config)#feature nv overlay`	Enables the NVE feature that is used for the segment routing Layer 2 EVPN.
Step 8	**nv overlay evpn** **Example:** `switch(config)#nv overlay evpn`	Enables EVPN.
Step 9	**interface loopback** *Interface_Number* **Example:** `switch(config)#interface loopback 1`	Configures the loopback interface for NVE.
Step 10	**ip address** *address* **Example:** `switch(config-if)#ip address 192.168.15.1`	Configures the IP address.

	Command or Action	Purpose
Step 11	**exit** **Example:** `switch(config-if)#exit`	Exits global address family configuration mode.
Step 12	**evpn** **Example:** `switch(config)#evpn`	Enters the EVPN configuration mode.

Step 13	**evi** *number*
	Example:
	`switch(config-evpn)#evi 1000`
	`switch(config-evpn-sr)#`
		Configures Layer 2 EVI. If required, you can manually configure the RT based on the EVI that is generated automatically.

Step 14	**encapsulation mpls** **Example:** `switch(config-evpn)#encapsulation mpls`	Enables MPLS encapsulation and ingress-replication.
Step 15	**source-interface loopback** *Interface_Number* **Example:** `switch(config-evpn-nve-encap)#source-interface loopback 1`	Specifies the NVE source interface.
Step 16	**exit** **Example:** `switch(config-evpn-nve-encap)#exit`	Exits the configuration.
Step 17	**vrf context** *VRF_NAME* **Example:** `switch(config)#vrf context Tenant-A`	Configures the VRF.
Step 18	**evi** *EVI_ID* **Example:** `switch(config-vrf)#evi 30001`	Configures L3 EVI.
Step 19	**exit** **Example:** `switch(config-vrf)#exit`	Exits the configuration.
Step 20	**VLAN** *VLAN_ID* **Example:** `switch(config)#vlan 1001`	Configures VLAN.
Step 21	**evi auto** **Example:**	Configures L2 EVI.

	Command or Action	Purpose
	`switch(config-vlan)#evi auto`	
Step 22	**exit** **Example:** `switch(config-vlan)#exit`	
Step 23	**router bgp** *autonomous-system-number* **Example:** `switch(config)#router bgp 1`	Enters the BGP configuration mode.
Step 24	**address-family l2vpn evpn** **Example:** `switch(config-router)#address-family l2vpn evpn`	Enables EVPN address family globally.
Step 25	**neighbor address** *remote-as autonomous-system-number* **Example:** `switch(config-router)#neighbor 192.169.13.1 remote as 2`	Configures BGP neighbor.
Step 26	**address-family l2vpn evpn** **Example:** `switch(config-router-neighbor)#address-family l2vpn evpn`	Enables EVPN address family for neighbor.
Step 27	**encapsulation mpls** **Example:** `switch(config-router-neighbor)#encapsulation mpls`	Enables MPLS encapsulation.
Step 28	**send-community extended** **Example:** `switch(config-router-neighbor)#send-community extended`	Configures BGP to advertise extended community lists.
Step 29	**vrf** *VRF_NAME* **Example:** `switch(config-router)#vrf Tenant-A`	Configures BGP VRF.
Step 30	**exit** **Example:** `switch(config-router)#exit`	Exits the configuration.

Configuring VLAN for EVI

Procedure

	Command or Action	Purpose
Step 1	**vlan** *number*	Specifies the VLAN.
Step 2	**evi** *auto*	Creates a BD label for the VLAN. This label is used as an identifier for the VLAN across the segment routing Layer 2 EVPN.

Configuring the NVE Interface

Procedure

	Command or Action	Purpose

Step 1	**configure terminal**
	Example:
	`switch# configure terminal`
	`switch(config)#`
		Enters global configuration mode.

Step 2	**interface loopback** *loopback_number* **Example:** `switch(config)# interface loopback 1`	Associates the IP address with this loopback interface and uses this IP address for the segment routing configuration.
Step 3	**ip address** **Example:** `switch(config-if)#ip address 192.169.15.1/32`	Specifies the IPv4 address family and enters router address family configuration mode.
Step 4	**evpn** **Example:** `switch(config)#evpn`	Enters EVPN configuration mode.
Step 5	**encapsulation mpls** **Example:** `switch(config-evpn)# encapsulation mpls`	Enables MPLS encapsulation and ingress-replication.
Step 6	**source-interface** *loopback_number* **Example:** `switch(config-evpn-nve-encap)#source-interface loopback 1`	Specifies the NVE source interface.
Step 7	**exit** **Example:**	Exits segment routing mode and returns to the configuration terminal mode.

	Command or Action	Purpose
	`switch(config)# exit`	

Configuring EVI Under VRF

Procedure

	Command or Action	Purpose
Step 1	**vrf context** *tenant*	Create a VRF Tenant.
Step 2	**evi** *number*	Configure Layer 3 EVI under VRF.

Configuring Anycast Gateway

The fabric forwarding configuration is necessary only if the SVIs are configured in the anycast mode.

Procedure

	Command or Action	Purpose
Step 1	**fabric forwarding anycast-gateway-mac 0000.aabb.ccdd**	Configures the distributed gateway virtual MAC address.
Step 2	**fabric forwarding mode anycast-gateway**	Associates SVI with the Anycast Gateway under the interface configuration mode.

Advertising Labelled Path for the Loopback Interface

The loopback interface, advertised as Layer 2 EVPN endpoint should be mapped to a label index. Thereby BGP advertises MPLS labelled path for the same.

Procedure

	Command or Action	Purpose

Step 1	**configure terminal**
	Example:
	`switch# configure terminal`
	`switch(config)#`
		Enters global configuration mode.

Step 2	[**no**]**router ospf** *process* **Example:** `switch(config)# router ospf test`	Enables the OSPF mode.
Step 3	**segment-routing** **Example:**	Configures the segment routing functionality under OSPF.

	Command or Action	Purpose
	`switch(config-router)# segment-routing mpls`	
Step 4	**connected-prefix-sid-map** **Example:** `switch(config-sr-mpls)# connected-prefix-sid-map`	Enters a sub-mode where you can configure address-family specific mappings for local prefixes and SIDs.
Step 5	**address-family ipv4** **Example:** `switch(config-sr-mpls-conn)# address-family ipv4`	Specifies IPv4 address prefixes.
Step 6	**1.1.1.1/32 index 100** **Example:** `switch(config-sr-mpls-conn-af)# 1.1.1.1/32 100`	Associates SID 100 with the address 1.1.1.1/32.
Step 7	**exit-address-family** **Example:** `switch(config-sr-mpls-conn-af)# exit-address-family`	Exits the address family.

About SRv6 Static Per-Prefix TE

The SRv6 Static Per-Prefix TE feature allows you to map and advertise prefixes that at mapped to non-default VRFs. This feature allows you to advertise multiple prefixes in a single instance using the matching VRF route target and prevents the manual entry of each prefix.

In Cisco NX-OS Release 9.3(5), only one VNF can service a VM.

Configuring a SRv6 Static Per-Prefix TE

Before you begin

Do the following:

- You must install and enable the MPLS feature set using the **install feature-set mpls** and **feature-set mpls** commands.
- You must enable the MPLS segment routing feature.

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:**	Enters global configuration mode.

	Command or Action	Purpose
	`switch# configure terminal` `switch(config)#`	
Step 2	**vrf context** *VRF_Name* **Example:** `switch(config)# vrf context vrf_2_7_8`	Defines VRF and enters the VRF configuration mode.
Step 3	**rd** *rd_format* **Example:** `switch(config-vrf)# rd 2.2.2.0:2`	Assign the RD to VRF.
Step 4	**address-family** {**ipv4** | **ipv6** } **Example:** `switch(config-vrf)# address-family ipv4 unicast`	Specifies either the IPv4 or the IPv6 address family for the VRF instance and enters the address family configuration mode.
Step 5	**route-target import** *route-target-id* **Example:** `switch(config-vrf)# route-target import 1:2`	Configures the importing of routes to the VRF.
Step 6	**route-target import** *route-target-id* **evpn** **Example:** `switch(config-vrf)# route-target import 1:2 evpn`	Configures importing of routes that have a matching route target value from the Layer 3 EVPN to the VRF.
Step 7	**route-target export** *route-target-id* **Example:** `switch(config-vrf)# route-target export 1:2`	Configures the exporting of routes from the VRF.
Step 8	**route-target export** *route-target-id* **evpn** **Example:** `switch(config-vrf)# route-target export 1:2 evpn`	Configures exporting of routes that have a matching route target value from the VRF to the Layer 3 EVPN.
Step 9	**router bgp** *autonomous-system-number* **Example:** `switch(config)# router bgp 65000`	Enables BGP and assigns the AS number to the local BGP speaker.
Step 10	**router-id** *id* **Example:** `switch(config-router)# router-id 2.2.2.0`	Configures the router ID.
Step 11	**address-family l2vpn evpn** **Example:**	Enters global address family configuration mode for the Layer 2 VPN EVPN.

	Command or Action	Purpose
	`switch(config-router-af)# address-family l2vpn evpn`	

Step 12	**neighbor** *ipv4-address* **remote-as**
	Example:
	`switch(config-router)# neighbor 7.7.7.0 remote-as 65000`
	`switch(config-router-neighbor)#`
		Configures the IPv4 address and AS number for a remote BGP peer.

Step 13	**update-source loopback** *number* **Example:** `switch(config-router-neighbor)# update-source loopback0`	Specifies the loopback number.
Step 14	**address-family l2vpn evpn** **Example:** `switch(config-router-neighbor)#address-family l2vpn evpn`	Enables EVPN address family for a neighbor.
Step 15	**send-community extended** **Example:** `switch(config-router-neighbor)#send-community extended`	Configures BGP to advertise extended community lists.
Step 16	**encapsulation mpls** **Example:** `switch(config-router-neighbor)#encapsulation mpls`	Enables MPLS encapsulation.
Step 17	**exit** **Example:** `switch(config-router-neighbor)#exit`	Exits the configuration.

Example

The following example shows how to configure RPM configuration in order to define the VRF VT.

```
rf context vrf_2_7_8
          rd 2.2.2.0:2
          address-family ipv4 unicast
            route-target import 0.0.1.1:2
            route-target import 0.0.1.1:2 evpn
            route-target export 0.0.1.1:2
            route-target export 0.0.1.1:2 evpn
ip extcommunity-list standard vrf_2_7_8-test permit rt 0.0.1.1:2
            route-map Node-2 permit 4
            match extcommunity vrf_2_7_8-test
            set extcommunity color 204
```

About RD Auto

The auto-derived Route Distinguisher (rd auto) is based on the Type 1 encoding format as described in IETF RFC 4364 section 4.2 https://tools.ietf.org/html/rfc4364#section-4.2. The Type 1 encoding allows a 4-byte administrative field and a 2-byte numbering field. Within Cisco NX-OS, the auto derived RD is constructed with the IP address of the BGP Router ID as the 4-byte administrative field (RID) and the internal VRF identifier for the 2-byte numbering field (VRF ID).

The 2-byte numbering field is always derived from the VRF, but results in a different numbering scheme depending on its use for the IP-VRF or the MAC-VRF:

- The 2-byte numbering field for the IP-VRF uses the internal VRF ID starting at 1 and increments. VRF IDs 1 and 2 are reserved for the default VRF and the management VRF respectively. The first custom defined IP VRF uses VRF ID 3.
- The 2-byte numbering field for the MAC-VRF uses the VLAN ID + 32767, which results in 32768 for VLAN ID 1 and incrementing.

Example auto-derived Route Distinguisher (RD)

- IP-VRF with BGP Router ID 192.0.2.1 and VRF ID 6 - RD 192.0.2.1:6
- MAC-VRF with BGP Router ID 192.0.2.1 and VLAN 20 - RD 192.0.2.1:32787

About Route-Target Auto

The auto-derived Route-Target (route-target import/export/both auto) is based on the Type 0 encoding format as described in IETF RFC 4364 section 4.2 (https://tools.ietf.org/html/rfc4364#section-4.2). IETF RFC 4364 section 4.2 describes the Route Distinguisher format and IETF RFC 4364 section 4.3.1 refers that it is desirable to use a similar format for the Route-Targets. The Type 0 encoding allows a 2-byte administrative field and a 4-byte numbering field. Within Cisco NX-OS, the auto derived Route-Target is constructed with the Autonomous System Number (ASN) as the 2-byte administrative field and the Service Identifier (EVI) for the 4-byte numbering field.

2-byte ASN

The Type 0 encoding allows a 2-byte administrative field and a 4-byte numbering field. Within Cisco NX-OS, the auto-derived Route-Target is constructed with the Autonomous System Number (ASN) as the 2-byte administrative filed and the Service Identifier (EVI) for the 4-byte numbering field.

Examples of an auto derived Route-Target (RT):

- IP-VRF within ASN 65001 and L3EVI 50001 - Route-Target 65001:50001
- MAC-VRF within ASN 65001 and L2EVI 30001 - Route-Target 65001:30001

For Multi-AS environments, the Route-Targets must either be statically defined or rewritten to match the ASN portion of the Route-Targets.

Note Auto derived Route-Targets for a 4-byte ASN are not supported.

4-byte ASN

The Type 0 encoding allows a 2-byte administrative field and a 4-byte numbering field. Within Cisco NX-OS, the auto-derived Route-Target is constructed with the Autonomous System Number (ASN) as the 2-byte administrative filed and the Service Identifier (EVI) for the 4-byte numbering field. With the ASN demand of 4-byte length and the EVI requiring 24-bit (3-bytes), the Sub-Field length within the Extended Community is exhausted (2-byte Type and 6-byte Sub-Field). As a result of the length and format constraint and the importance of the Service Identifiers (EVI) uniqueness, the 4-byte ASN is represented in a 2-byte ASN named AS_TRANS, as described in IETF RFC 6793 section 9 (https://tools.ietf.org/html/rfc6793#section-9). The 2-byte ASN 23456 is registered by the IANA (https://www.iana.org/assignments/iana-as-numbers-special-registry/iana-as-numbers-special-registry.xhtml) as AS_TRANS, a special purpose AS number that aliases 4-byte ASNs.

Example auto derived Route-Target (RT) with 4-byte ASN (AS_TRANS):

- IP-VRF within ASN 65656 and L3EVI 50001 - Route-Target 23456:50001
- MAC-VRF within ASN 65656 and L2EVI 30001 - Route-Target 23456:30001

Configuring RD and Route Targets for BD

The Bridge Domain (BD) RD and Route Targets are automatically generated when you configure **evi auto** under the VLAN. To configure the BD RD and Route Targets manually, perform these steps:

Procedure

	Command or Action	Purpose

Step 1	**configure terminal**
	Example:
	`switch# configure terminal`
	`switch(config)#`
		Enters global configuration mode.

Step 2	**evpn** **Example:** `switch(config)# evpn`	Enters EVPN configuration mode.
Step 3	**evi** *VLAN_ID* **Example:** `switch(config-evpn)# evi 1001`	Specifies L2 EVI to configure RD/Route Target.
Step 4	**rd** *rd_format* **Example:** `switch(config-evpn-evi-sr)# rd 192.1.1.1:33768`	Configures RD.
Step 5	**route-target both** *rt_format* **Example:** `switch(config-evpn-evi-sr)# route-target both 1:20001`	Configures Route Target.

Configuring RD and Route Targets for VRF

The VRF RD and Route Targets are automatically generated when you configure the **evi** *evi_ID* under the VRF. To configure the VRF RD and Route Targets manually, perform these steps:

Procedure

	Command or Action	Purpose

Step 1	**configure terminal**
	Example:
	`switch# configure terminal`
	`switch(config)#`
		Enters global configuration mode.

Step 2	**vrf context** *VRF_NAME* **Example:** `switch(config)# vrf context A`	Configures the VRF.
Step 3	**rd auto** or **rd_format** **Example:** `switch(config-vrf)# rd auto`	Configures RD.
Step 4	**address-family ipv4 unicast** **Example:** `switch(config-vrf)# address-family ipv4 unicast`	Enables IPv4 address family.
Step 5	**route-target both** *rt_format* **evpn** **Example:** `switch(config-vrf-af-ipv4)# route-target both 1:30001 evpn`	Configures Route Target.

Configuration Examples for Layer 2 EVPN over Segment Routing MPLS

The following examples show the configuration for Layer 2 EVPN over Segment Routing MPLS:

```
install feature-set mpls
feature-set mpls
nv overlay evpn
feature bgp
feature mpls segment-routing
feature mpls evpn
feature interface-vlan
feature nv overlay

fabric forwarding anycast-gateway-mac 0000.1111.2222

vlan 1001
  evi auto

vrf context Tenant-A
  evi 30001
```

```
interface loopback 1
  ip address 192.168.15.1/32

interface vlan 1001
  no shutdown
  vrf member Tenant-A
  ip address 111.1.0.1/16
  fabric forwarding mode anycast-gateway

router bgp 1
  address-family l2vpn evpn
    neighbor 192.169.13.1
      remote-as 2
      address-family l2vpn evpn
        send-community extended
        encapsulation mpls
    vrf Tenant-A

evpn
  encapsulation mpls
    source-interface loopback 1
```

Configuring Proportional Multipath for VNF for Segment Routing

About Proportional Multipath for VNF for Segment Routing

In Network Function Virtualization Infrastructures (NFVi), service networks (Portable IPs) are routed by Virtual Network Functions (VNFs). The VNFs, also referred to as portable IP-Gateway (PIP-GW) routes the data packets to and from the VMs in the VNF. The Proportional Multipath for VNF for Segment Routing feature enables advertising the VNF of a service network (PIP) in the EVPN address-family. The IP address of the VNF is encoded in the “Gateway-IP Address” field of the EVPN IP Prefix Route NLRI advertisement of a service network.

By advertising the IP address of the VNFs, ingress nodes in the EVPN fabric recursively resolve the VNF IP address to the leaf attached to the VNF, which could be the same node that advertises the service network (PIP).

Route-injectors are BGP protocols that inject routes in the IPv4 or IPv6 AF. In this case, the route-injector injects routes to the VMs whose next hop is set as VNFs.

Unlike a route-injector, VNFs can participate in a routing protocol to advertise the VM reachability. The supported protocols are eBGP, IS-IS, and OSPF.

Enabling Proportional Multipath for VNF for Segment Routing

You can enable the Proportional Multipath for VNF for Segment Routing feature to redistribute routes for IGP or static routes by preserving the next-hop paths. You can then export and advertise the gateway-IP for the reoriginated EVPN type-5 routes.

In Cisco NX-OS Release 9.3(5), only one VNF can service a VM.

Before you begin

Do the following:

- Install and enable the MPLS feature set using the **install feature-set mpls** and **feature-set mpls** commands.
- Enable the MPLS segment routing feature.

Procedure

	Command or Action	Purpose

Step 1	**configure terminal**
	Example:
	`switch# configure terminal`
	`switch(config)#`
		Enter global configuration mode.

Step 2	**route-map** *export-l2evpn-rtmap* **permit** *10* **Example:** `switch(config)# route-map export-l2evpn-rtmap permit 10`	<<need description>>
Step 3	**match ip address prefix-list** *pip-pfx-list* **Example:** `switch(config-route-map)# match ip prefix-list vm-pfx-list`	Defines the prefixes that must be advertised with PIP-GW as the gateway.
Step 4	**set evpn gateway-ip use-nexthop** **Example:** `switch(config-route-map)# set evpn gateway-ip use-nexthop`	Defines specific routes to advertise the gateway-ip.

Step 5	**vrf context** *VRF_Name*
	Example:
	`switch(config-route-map)# vrf context vrf`
	`switch(config-route-map)# address-family ipv4 unicast`
	`switch(config-route-map)# export map export-l2evpn-rtmap`
		Applies the route map to the vrf context.

Step 6	**address-family ipv4 unicast**
	Example:
	`switch(config-route-map)# address-family ipv4 unicast`
	`switch(config-route-map)# export map export-l2evpn-rtmap`
		Applies the route map to the vrf context.

Step 7	**export map** *export-l2evpn-rtmap* **Example:** `switch(config-route-map)# export map export-l2evpn-rtmap`	Applies the route map to the vrf context.

	Command or Action	Purpose
Step 8	**router bgp** *number* **Example:** `switch(config)# router bgp 100`	Configure BGP.
Step 9	**vrf** *VRF_Name* **Example:** `switch(config-route-map)# vrf vrf3`	Applies the route map to the vrf context.
Step 10	**address-family ipv4 unicast** **Example:** `switch(config-router)# address-family ipv4 unicast`	Configure address family for IPv4.
Step 11	**export-gateway-ip** **Example:** `switch(config-route-map)# export-gateway-ip`	Exports and advertises the gateway-ip to reconnect the EVPN type-5 routes. **Note** The export gateway-ip and set the EVPN gateway configurations can be performed simultaneously. If you configure them simultaneously, all prefixes are exported with the gateway-ip.

vPC Multihoming

About Multihoming

Cisco Nexus platform switches support vPC-based multihoming, where a pair of switches act as a single device for redundancy and both switches function in active mode. With Cisco Nexus platform switches in an EVPN environment, there are two solutions that support Layer 2 multihoming; these solutions are based on the traditional vPC (emulated or virtual IP address), where the MCT link is required and the BGP EVPN techniques.

While using the BGP EVPN control plane, each vPC pair uses a common virtual IP (VIP) to provide active/active redundancy. BGP EVPN based multihoming further provides fast convergence during certain failure scenarios, that otherwise cannot be achieved without a control protocol (data plane flood and learn).

Per-BD label on vPC Peers

To ensure that the vPC peers have the same per-BD label, you must specify the per-BD label to have the following value:

```
Label value = Label_base + VLAN_ID
```

The label base is configured on the same vPC peers. Currently, the VLAN configuration is identical on both the vPC peers, which ensures that both vPC peers have the same label.

In Cisco NX-OS Release 9.3(1), configuring the per-BD label is not supported. This release supports only evi auto.

Per-VRF label on vPC Peers

To ensure that the vPC peers have the same per-VRF label, you must specify the per-VRF label to have the following value:

```
Label value = Label_base + vrf_allocate_index
```

To configure the allocate-index for the vPC peers, do the following:

```
Router bgp 1
  vrf Tenant_A
    allocate-index 11
```

Configuring Backup Link

The backup link needs to be configured between the vPC peers. This link can be any Layer 3 link which is parallel to MCT.

Example

```
interface vlan 100
    ip add 10.1.1.1/24
        mpls ip forwarding

< enable underlay protocol >
```

Guidelines and Limitations for vPC Multihoming

vPC multihoming has the following guidelines and limitations:

- ESI-based multihoming is not supported.
- The physical and virtual secondary IP addresses should be both advertised via the MPLS labeled path.
- vPC consistency checking is not supported for the per-BD label configuration.

Configuration Examples for vPC Multihoming

This example shows the configuration for vPC multihoming:

- vPC Primary

```
interface loopback1
  ip address 192.169.15.1/32
  ip address 192.169.15.15/32 secondary

evpn
  encapsulation mpls
    source-interface loopback1

vlan 101
  evi auto

vrf context A
  evi 301
```

```
router bgp 1
  vrf A
    allocate-index 1001
```

- vPC Secondary

```
interface loopback1
  ip address 192.169.15.2/32
  ip address 192.169.15.15/32 secondary

evpn
  encapsulation mpls
    source-interface loopback1

vlan 101
  evi auto

vrf context A
  evi 301

router bgp 1
  vrf A
    allocate-index 1001
```

Configuring Layer 3 EVPN and Layer 3 VPN over Segment Routing MPLS

This section describes tasks to configure the Layer 3 EVPN and stitching of L3 EVPN and L3VPN router. Perform the following tasks to complete the configuration:

Configuring the Features to Enable Layer3 EVPN and Layer3 VPN

Before you begin

Install the VPN Fabric license.

Make sure that the **feature interface-vlan** command is enabled.

Procedure

	Command or Action	Purpose
Step 1	**feature bgp**	Enables BGP feature and configurations.
Step 2	**install feature-set mpls**	Enables MPLS configuration commands.
Step 3	**feature-set mpls**	Enables MPLS configuration commands.
Step 4	**feature mpls segment-routing**	Enables segment routing configuration commands.
Step 5	**feature mpls evpn**	Enables EVPN over MPLS configuration commands. This command is mutually exclusive with the **feature-nv** CLI command.

	Command or Action	Purpose
Step 6	**feature mpls l3vpn**	Enables EVPN over MPLS configuration commands. This command is mutually exclusive with the **feature-nv** CLI command.

Configuring VRF and Route Targets for Import and Export Rules

Procedure

	Command or Action	Purpose
Step 1	**configure terminal**	Enters global configuration mode.
Step 2	**vrf** *vrf-name*	Defines a VPN routing and forwarding (VRF) instance and enters the VRF configuration mode.
Step 3	**rd auto**	Automatically assigns a unique route distinguisher (RD) to VRF.
Step 4	**address-family** { **ipv4** | **ipv6** } **unicast**	Specifies either the IPv4 or IPv6 address family for the VRF instance and enters address family configuration submode.
Step 5	**route-target import** *route-target-id*	Configures importing of routes to the VRF from the L3VPN BGP NLRIs that have the matching route-target value.
Step 6	**route-target export** *route-target-id*	Configures exporting of routes from the VRF to the L3VPN BGP NLRIs and assigns the specified route-target identifiers to the L3VPN BGP NLRIs.
Step 7	**route-target import** *route-target-id* **evpn**	Configures importing of routes from the L3 EVPN BGP NLRI that have the matching route-target value.
Step 8	**route-target export** *route-target-id* **evpn**	Configures exporting of routes from the VRF to the L3 EVPN BGP NLRIs and assigns the specified route-target identifiers to the BGP EVPN NLRIs.

Configuring BGP EVPN and Label Allocation Mode

You can use MPLS tunnel encapsulation using the **encapsulation mpls** command. You can configure the label allocation mode for the EVPN address family. The default tunnel encapsulation in EVPN for IP Route type in NX-OS is VXLAN.

Advertisement of (IP or Label) bindings from a Cisco Nexus 9000 Series switch via BGP EVPN enables a remote switch to send the routed traffic to that IP using the label for that IP to the switch that advertised the IP over MPLS.

The IP prefix route (Type-5) is:

- Type-5 route with MPLS encapsulation

```
RT-5 Route – IP Prefix

RD:  L3 RD
IP Length:  prefix length
IP address:  IP (4 bytes)
Label1:  BGP MPLS Label
Route Target
RT for IP-VRF
```

The default label allocation mode is per-VRF for Layer 3 EVPN over MPLS.

Complete the following steps to configure BGP EVPN and label allocation mode:

Before you begin

You must install and enable the MPLS feature set using the **install feature-set mpls** and **feature-set mpls** commands.

You must enable the MPLS segment routing feature.

Procedure

	Command or Action	Purpose
Step 1	**configure terminal**	Enters global configuration mode.

Step 2	[**no**] **router bgp** *autonomous-system-number*
	Example:
	`switch(config)# router bgp 64496`
	`switch(config-router)#`
		Enables BGP and assigns the AS number to the local BGP speaker. The AS number can be a 16-bit integer or a 32-bit integer in the form of a higher 16-bit decimal number and a lower 16-bit decimal number in xx.xx format.
		Use the **no** option with this command to remove the BGP process and the associated configuration.

Step 3	Required: **address-family l2vpn evpn**
	Example:
	`switch(config-router)# address-family l2vpn evpn`
	`switch(config-router-af)#`
		Enters global address family configuration mode for the Layer 2 VPN EVPN.

Step 4	Required: **exit**
	Example:
	`switch(config-router-af)# exit`
	`switch(config-router)#`
		Exits global address family configuration mode.

	Command or Action	Purpose

Step 5	**neighbor** *ipv4-address* **remote-as** *autonomous-system-number*
	Example:
	`switch(config-router)# neighbor 10.1.1.1 remote-as 64497`
	`switch(config-router-neighbor)#`
		Configures the IPv4 address and AS number for a remote BGP peer.

Step 6	**address-family l2vpn evpn**
	Example:
	`switch(config-router-neighbor)# address-family l2vpn evpn`
	`switch(config-router-neighbor-af)#`
		Advertises the labeled Layer 2 VPN EVPN.

Step 7	**encapsulation mpls**
	Example:
	`router bgp 100`
	`address-family l2vpn evpn`
	`neighbor NVE2 remote-as 100`
	`address-family l2vpn evpn`
	`send-community extended`
	`encapsulation mpls`
	`vrf foo`
	`address-family ipv4 unicast`
	`advertise l2vpn evpn`
	BGP segment routing configuration:
	`router bgp 100`
	`address-family ipv4 unicast`
	`network 200.0.0.1/32 route-map label_index_pol_100`
	`network 192.168.5.1/32 route-map label_index_pol_101`
	`network 101.0.0.0/24 route-map label_index_pol_103`
	`allocate-label all`
	`neighbor 192.168.5.6 remote-as 20`
	`address-family ipv4 labeled-unicast`
	`send-community extended`
		Enables BGP EVPN address family and sends EVPN type-5 route update to the neighbors.
		Note The default tunnel encapsulation in EVPN for the IP route type in NX-OS is VXLAN. To override that, a new CLI is introduced to indicate MPLS tunnel encapsulation.

Step 8	**vrf** <*customer_name*>	Configures the VRF.
Step 9	**address-family ipv4 unicast**	Enters global address family configuration mode for the IPv4 address family.
Step 10	**advertise l2vpn evpn**	Advertises Layer 2 VPN EVPN.
Step 11	**redistribute direct route-map DIRECT_TO_BGP**	Redistributes the directly connected routes into BGP-EVPN.
Step 12	**label-allocation-mode per-vrf**	Sets the label allocation mode to per-VRF. If you want to configure the per-prefix label

	Command or Action	Purpose
		mode, use the **no label-allocation-mode per-vrf** CLI command. For the EVPN address family, the default label allocation is per-vrf, compared to per-prefix mode for the other address-families where the label allocation CLI is supported. No form of CLI is displayed in the running configuration.

Example

See the following example for configuring per-prefix label allocation:

```
router bgp 65000
    [address-family l2vpn evpn]
    neighbor 10.1.1.1
        remote-as 100
        address-family l2vpn evpn
        send-community extended
    neighbor 20.1.1.1
        remote-as 65000
        address-family l2vpn evpn
        encapsulation mpls
        send-community extended
    vrf customer1
        address-family ipv4 unicast
            advertise l2vpn evpn
            redistribute direct route-map DIRECT_TO_BGP
            no label-allocation-mode per-vrf
```

Configuring BGP Layer 3 EVPN and Layer 3 VPN Stitching

In order to configure the stitching on the same router, configure the layer 3 VPN neighbor relationship and router advertisement.

Procedure

	Command or Action	Purpose

Step 1	**configure terminal**
	Example:
	`switch# configure terminal`
	`switch(config)#`
		Enters global configuration mode.

Step 2	[**no**] **router bgp** *autonomous-system-number*
	Example:
	`switch# configure terminal`
	`switch(config)# router bgp 64496`
	`switch(config-router)#`
		Enables BGP and assigns the AS number to the local BGP speaker. The AS number can be a 16-bit integer or a 32-bit integer in the form of a higher 16-bit decimal number and a lower 16-bit decimal number in xx.xx format.

	Command or Action	Purpose
		Use the **no** option with this command to remove the BGP process and the associated configuration.

Step 3	**address-family** {**vpnv4** | **vpnv6**} **unicast**
	Example:
	`switch(config-router)# address-family vpnv4 unicast`
	`switch(config-router-af)# address-family vpnv6 unicast`
	`switch(config-router-af)#`
		Enters global address family configuration mode for the Layer 3 VPNv4 or VPNv6.

Step 4	**exit**
	Example:
	`switch(config-router-af)# exit`
	`switch(config-router)#`
		Exits global address family configuration mode.

Step 5	**neighbor** *ipv4-address* **remote-as** *autonomous-system-number* **Example:** `switch(config-router)# neighbor 20.1.1.1 remote-as 64498`	Configures the IPv4 address and AS number for a remote BGP L3VPN peer.

Step 6	**address-family** {**vpnv4** | **vpnv6**} **unicast**
	Example:
	`switch(config-router)# address-family vpnv4 unicast`
	`switch(config-router-af)# address-family vpnv6 unicast`
	`switch(config-router-af)#`
		Configure the neighbor address-family for VPNv4 or VPNv6.

Step 7	**send-community extended**	Enables BGP VPN address family
Step 8	**import l2vpn evpn reoriginate**	Configures import of routing information from the Layer 3 VPN BGP NLRIs that has route target identifier matching the normal route target identifier and exports this routing information after re-origination that assigns it with stitching route target identifier, to the BGP EVPN neighbor.

Step 9	**neighbor** *ipv4-address* **remote-as** *autonomous-system-number*
	Example:
	`switch(config-router)# neighbor 10.1.1.1 remote-as 64497`
	`switch(config-router-neighbor)#`
		Configures the IPv4 address and AS number for a remote Layer 3 EVPN BGP peer.

Step 10	**address-family** {**l2vpn** | **evpn** **Example:**	Configure the neighbor address-family for Layer 3 EVPN.

	Command or Action	Purpose
	`switch(config-router-neighbor)# address-family l2vpn evpn switch(config-router-neighbor-af)#`	
Step 11	**import vpn unicast reoriginate**	Enables import of routing information from BGP EVPN NLRIs that has route target identifier matching the stitching route target identifier and exports this routing information after re-origination to the Layer 3 VPN BGP neighbor.
Step 12	**vrf** <*customer_name*>	Configures the VRF.
Step 13	**address-family ipv4 unicast**	Enters global address family configuration mode for the IPv4 address family.
Step 14	**advertise l2vpn evpn**	Advertises Layer 2 VPN EVPN.

Example

```
vrf context Customer1
    rd auto
    address-family ipv4 unicast
        route-target import 100:100
        route-target export  100:100
        route-target import 100:100 evpn
        route-target export 100:100 evpn

segment-routing
    mpls
      global-block 11000 20000
      connected-prefix-sid
         address-family ipv4 unicast
         200.0.0.1 index 101
!
int lo1
  ip address 200.0.0.1/32
!
interface e1/13
  description "MPLS interface towards Core"
  ip address 192.168.5.1/24
  mpls ip forwarding
  no shut

router bgp 100
address-family ipv4 unicast
allocate-label all
address-family ipv6 unicast
address-family l2vpn evpn
address-family vpnv4 unicast
address-family vpnv6 unicast
 neighbor 10.0.0.1 remote-as 200
    update-source loopback1
    address-family vpnv4 unicast
      send-community extended
      import l2vpn evpn reoriginate
    address-family vpnv6 unicast
      import l2vpn evpn reoriginate
```

```
    send-community extended
neighbor 20.0.0.1 remote-as 300
  address-family l2vpn evpn
    send-community extended
    import vpn unicast reoriginate
    encapsulation mpls
neighbor 192.168.5.6 remote-as 300
    address-family ipv4 labeled-unicast
vrf Customer1
  address-family ipv4 unicast
    advertise l2vpn evpn
  address-family ipv6 unicast
    advertise l2vpn evpn
```

Configuring BGP L3 VPN over Segment Routing

Before you begin

You must install and enable the MPLS feature set using the **install feature-set mpls** and **feature-set mpls** commands.

You must enable the MPLS segment routing feature.

You must enable the MPLS L3 VPN feature using the **feature mpls l3vpn** command.

Procedure

	Command or Action	Purpose

Step 1	**configure terminal**
	Example:
	`switch# configure terminal`
	`switch(config)#`
		Enters global configuration mode.

Step 2	[**no**] **router bgp** *autonomous-system-number*
	Example:
	`switch(config)# router bgp 64496`
	`switch(config-router)#`
		Enables BGP and assigns the AS number to the local BGP speaker. The AS number can be a 16-bit integer or a 32-bit integer in the form of a higher 16-bit decimal number and a lower 16-bit decimal number in xx.xx format.
		Use the **no** option with this command to remove the BGP process and the associated configuration.

Step 3	**address-family** {**vpnv4** | **vpnv6**} **unicast**
	Example:
	`switch(config-router)# address-family vpnv4 unicast`
	`switch(config-router-af)# address-family vpnv6 unicast`
	`switch(config-router-af)#`
		Enters global address family configuration mode for the Layer 3 VPNv4 or VPNv6.

Step 4	[**no**] **allocate-label option-b**	Disables the inter-AS option-b

	Command or Action	Purpose

Step 5	Required: **exit**
	Example:
	`switch(config-router-af)# exit`
	`switch(config-router)#`
		Exits global address family configuration mode.

Step 6	**neighbor** *ipv4-address* **remote-as** *autonomous-system-number*
	Example:
	`switch(config-router)# neighbor 20.1.1.1 remote-as 64498`
	`switch(config-router-neighbor)#`
		Configures the IPv4 address and AS number for a remote BGP L3VPN peer.

Step 7	**address-family** {**vpnv4** | **vpnv6** } **unicast**
	Example:
	`switch(config-router-neighbor)# address-family vpnv4 unicast`
	`switch(config-router-neighbor-af)#`
		Configure the neighbor address-family for VPNv4 or VPNv6.

Step 8	**send-community extended**	Enables BGP VPN address family.
Step 9	**vrf** <*customer_name*>	Configures the VRF.
Step 10	**allocate-index x**	Configure the allocate-index.
Step 11	**address-family ipv4 unicast**	Enters global address family configuration mode for the IPv4 address family.
Step 12	**redistribute direct route-map DIRECT_TO_BGP**	Redistributes the directly connected routes into BGP-L3VPN.

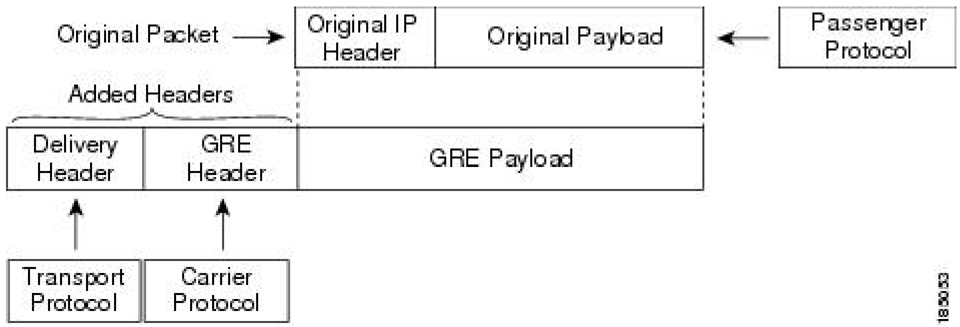
Configuring Segment Routing MPLS and GRE Tunnels

GRE Tunnels

You can use generic routing encapsulation (GRE) as the carrier protocol for a variety of passenger protocols.

The following figure shows the IP tunnel components for a GRE tunnel. The original passenger protocol packet becomes the GRE payload and the device adds a GRE header to the packet. The device then adds the transport protocol header to the packet and transmits it.

Figure 12: GRE PDU



Segment Routing MPLS and GRE

Beginning Cisco NX-OS Release 9.3(1), you can configure both, segment routing MPLS and generic routing encapsulation (GRE) on a Cisco Nexus device. Both these technologies operate seamlessly. All MPLS traffic can be forwarded to the GRE tunnel after the MPLS tunnel termination. Similarly, you can forward all traffic from the GRE tunnel to the MPLS cloud after the GRE termination.

All PE routers can initiate, forward, or terminate the GRE traffic from or to another GRE cloud. Similarly, all tunnel transit or tunnel end nodes can configure MPLS tunnel encapsulation.

When both, the tunnel and segment routing is enabled on the Cisco Nexus 9000 switches, the following is the TTL behavior is for the respective flows:

- Incoming IP traffic, egresses with GRE header, the TTL value in the GRE header is one less than the TTL value of the incoming IP packet.
- Incoming IP traffic, egresses with MPLS header, the TTL value in the MPLS header is one less than the TTL value of the incoming IP packet.
- Incoming GRE traffic, egresses with MPLS header, the TTL value in the MPLS header is default (255).
- Incoming MPLS traffic, egresses with GRE header, the TTL value in the GRE header is default (255).

Guidelines and Limitations for Segment Routing MPLS and GRE

Segment routing MPLS and GRE have the following guidelines and limitations:

- Ingress stats are not supported for tunnel packets.
- Only template-mpls-heavy template is supported.
- MPLS segment routing is not supported on the tunnel interfaces.
- Due to a hardware limitation on the modular switches, the tunnel Tx traffic is not supported if the egress interface for the tunnel destination IP address is over the Cisco Nexus 9300-FX/FX2 platform switches.
- Maximum four GRE tunnels are supported.
- Beginning with Cisco NX-OS Release 9.3(3), you can configure both, segment routing MPLS and GRE on Cisco Nexus 9300-GX platform switches.
- Tunnel Rx packet counters do not work when both segment routing MPLS and GRE coexist.

Configuring Segment Routing MPLS and GRE

You can enable MPLS segment routing as long as mutually-exclusive MPLS features such as static MPLS are not enabled.

Before you begin

You must install and enable the MPLS feature set using the **install feature-set mpls** and **feature-set mpls** commands.

You must enable the tunneling feature using the **feature tunnel** command.

Procedure

	Command or Action	Purpose

Step 1	**configure terminal**
	Example:
	`switch# configure terminal`
	`switch(config)#`
		Enters global configuration mode.

Step 2	[**no**] **feature segment-routing** **Example:** `switch(config)# feature segment-routing`	Enables the MPLS segment routing feature. The **no** form of this command disables the MPLS segment routing feature.
Step 3	(Optional) **show running-config | inc 'feature segment-routing'** **Example:** `switch(config)# show running-config | inc 'feature segment-routing'`	Displays the status of the MPLS segment routing feature.
Step 4	(Optional) **copy running-config startup-config** **Example:** `switch(config)# copy running-config startup-config`	Copies the running configuration to the startup configuration.

Step 5	**configure terminal**
	Example:
	`switch# configure terminal`
	`switch(config)#`
		Enters global configuration mode.

Step 6	**feature tunnel**
	Example:
	`switch(config)# feature tunnel`
	`switch(config-if)#`
		Allows the creation of a new tunnel interface.
		To disable the tunnel interface feature, use the **no** form of this command.

Step 7	switch(config)# **interface tunnel** *number*	Enters a tunnel interface configuration mode.
Step 8	switch(config-if)# **tunnel mode** {**gre ip** }	Sets this tunnel mode to GRE.

	Command or Action	Purpose
		The **gre** and **ip** keywords specify that GRE encapsulation over IP will be used.
Step 9	**tunnel source** {*ip-address* |*interface-name*} **Example:** `switch(config-if)# tunnel source ethernet 1/2`	Configures the source address for this IP tunnel. The source can be specified by IP address or logical interface name.
Step 10	**tunnel destination** {*ip-address* |*host-name*} **Example:** `switch(config-if)# tunnel destination 192.0.2.1`	Configures the destination address for this IP tunnel. The destination can be specified by IP address or logical host name.
Step 11	**tunnel use-vrf** *vrf-name* **Example:** `switch(config-if)# tunnel use-vrf blue`	
Step 12	**ipv6 address** *IPv6 address*	`switch(config-if)# 10.1.1.1` Configures the IPv6 address. **Note** The tunnel source and the destination addresses are still the same (IPv4 address.)
Step 13	(Optional) switch(config-if)# **show interface tunnel** *number*	Displays the tunnel interface statistics.
Step 14	switch(config-if)# **mtu** *value*	Sets the maximum transmission unit (MTU) of IP packets sent on an interface.
Step 15	(Optional) switch(config-if)# **copy running-config startup-config**	Saves the change persistently through reboots and restarts by copying the running configuration to the startup configuration.

Verifying the Segment Routing MPLS and GRE Configuration

To display the segment routing MPLS and GRE configuration, perform one of the following tasks:

Command	Purpose
show segment-routing mpls	Displays segment routing mpls information

Verifying SR-TE for Layer 3 EVPN

The ODN verifications are based on L3VPN VRF prefixes.

1. Verify that the PCEP session between R1 (headend and PCE server) is established.

```
R1# show srte pce ipv4 peer

PCC's peer database:
--------------------
Remote PCEP conn IPv4 addr: 58.8.8.8
Local PCEP conn IPv4 addr: 51.1.1.1
Precedence: 0
State: up
```

2. Verify BGP LS and BGP EVPN session on R1, R3, and R6 using the following commands:
   - Show bgp l2vpn evpn summary
   - Show bgp link-state summary

3. Verify that the R1 (headend) has no visibility to the R6 loopback address.

```
R1# show ip route 56.6.6.6
IP Route Table for VRF "default"
'*' denotes best ucast next-hop
'**' denotes best mcast next-hop
'[x/y]' denotes [preference/metric]
'%<string>' in via output denotes VRF <string>

56.6.6.6/32, ubest/mbest: 1/0
    *via Null0, [1/0], 1d02h, static
```

4. Verify that the VRF prefix is injected via MP-BGP in a R1 VRF SR routing table.

```
R1# show ip route vrf sr
106.107.4.1/32, ubest/mbest: 1/0
    *via binding label 100534%default, [20/0], 1d01h, bgp-6503, external, tag 6500
(mpls-vpn)
```

5. Verify the SR-TE Tunnel.

```
R1# show srte policy
Policy name: 51.1.1.1|1001
    Source: 51.1.1.1
    End-point: 56.6.6.6
    Created by: bgp
    State: UP
    Color: 1001
    Insert: FALSE
    Re-opt timer: 0
    Binding-sid Label: 100534
    Policy-Id: 2
    Flags:
    Path type = MPLS           Path options count: 1
     Path-option Preference:100 ECMP path count: 1
      1.      PCE         Weighted: No
        Delegated PCE: 58.8.8.8
                Index: 1                 Label: 101104
                Index: 2                 Label: 201102
                Index: 3                 Label: 201103
```

Verifying the Segment Routing Configuration

To display the segment routing configuration, perform one of the following tasks:

Command	Purpose
show bgp ipv4 labeled-unicast *prefix*	Displays the advertised label index and the selected local label for the specified IPv4 prefix.
show bgp paths	Displays the BGP path information, including the advertised label index.
show mpls label range	Displays the configured SRGB range of labels.
show route-map [*map-name*]	Displays information about a route map, including the label index.
show running-config | inc 'feature segment-routing'	Displays the status of the MPLS segment routing feature.
show ip ospf neighbors detail	Displays the list of OSPFv2 neighbors and the adjacency SID allocated, along with the corresponding flags.
show ip ospf database opaque-area	Displays the LSAs for the adjacency SID.
show ip ospf segment-routing adj-sid-database	Displays all locally allocated adjacency SIDs.
show running-config segment-routing	Displays the status of the segment routing feature.
show srte policy	Displays the list of policies available in the SR-TE.
show srte policy fh	Displays the set of first hops.
show segment-routing mpls clients	Displays the clients registered with the SR-APP.
show segment-routing mpls details	Displays detailed information.
show segment-routing ipv4	Displays the information for the IPv4 address family.
show segment-routing mpls	Displays segment routing mpls information
show segment-routing ipv4 connected-prefix-sid	Displays the MPLS label range for the SRGB. **Note** This command is only available in Cisco NX-OS Release 9.3(1) .
show ip ospf *process*	Displays the OSPF mode.
show ip ospf *process* **segment-routing sid-database**	Displays the segment routing database details.
show ip ospf *process* **segment-routing global block**	Displays the segment routing global block information.
show nve evi	Displays the status of the EVIs.
show nve peer mpls	Displays the status of the segment routing peers.
show nve adjacency mpls	Displays the status of the peer adjacencies.

Additional References

Related Documents

Related Topic	Document Title
BGP	*Cisco Nexus 9000 Series Unicast Routing Configuration Guide*

CHAPTER 10

Configuring MVPNs

This chapter contains information on how to configure multicast virtual private networks (MVPNs)

About MVPNs

The multicast virtual private networks (MVPNs) feature allows you to support multicast connectivity over Layer 3 VPN. IP multicast is used to stream video, voice, and data to an VPN network core.

Historically, point-to-point tunnels were the only way to connect through an enterprise or service provider network. Although such tunneled networks had scalability issues, they were the only means of passing IP multicast traffic through a virtual private network (VPN). Because Layer 3 VPNs support only unicast traffic connectivity, deploying with a Layer 3 VPN allows operators to offer both unicast and multicast connectivity to Layer 3 VPN customers

MVPNs allows you to configure and support multicast traffic in an MVPN environment. MVPNs support routing and forwarding of multicast packets for each individual virtual routing and forwarding (VRF) instance, and it also provides a mechanism to transport VPN multicast packets across the enterprise or service provider backbone. IP multicast is used to stream video, voice, and data to a VPN network core.

A VPN allows network connectivity across a shared infrastructure, such as an Internet Service Provider (ISP). Its function is to provide the same policies and performance as a private network at a reduced cost of ownership.

MVPNs allow an enterprise to transparently interconnect its private network across the network backbone. Using MVPNs to interconnect an enterprise network does not change the way that an enterprise network is administered and it does not change general enterprise connectivity.

MVPN Routing and Forwarding and Multicast Domains

MVPNs introduce multicast routing information to the VPN routing and forwarding table. When a provider edge (PE) router receives multicast data or control packets from a customer edge (CE) router, the router forwards the data or control packets according to the information in the MVPN routing and forwarding (MVRF).

A set of MVRFs that can send multicast traffic to each other constitutes a multicast domain. For example, the multicast domain for a customer that wanted to send certain types of multicast traffic to all global employees would consist of all CE routers that are associated with that enterprise.

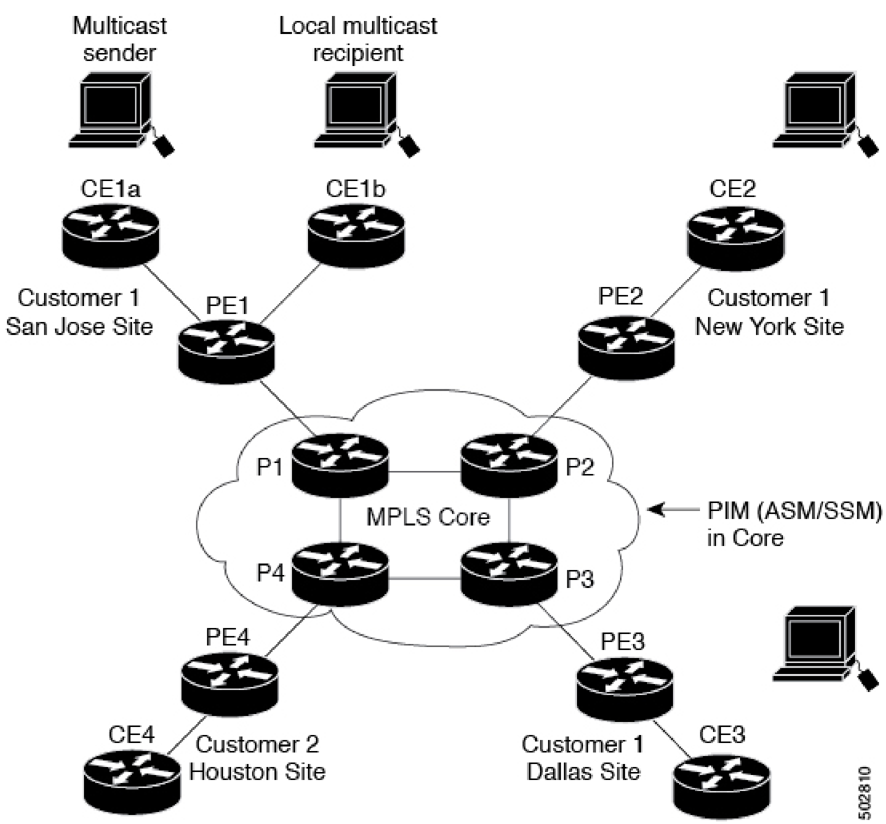
Multicast Distribution Trees

MVPNs establish a static default multicast distribution tree (MDT) for each multicast domain. The default MDT defines the path used by PE routers to send multicast data and control messages to every other PE router in the multicast domain.

MVPNs also support the dynamic creation of MDTs for high-bandwidth transmission. Data MDTs are intended for high-bandwidth sources such as full-motion video inside the VPN to ensure optimal traffic forwarding in the VPN core.

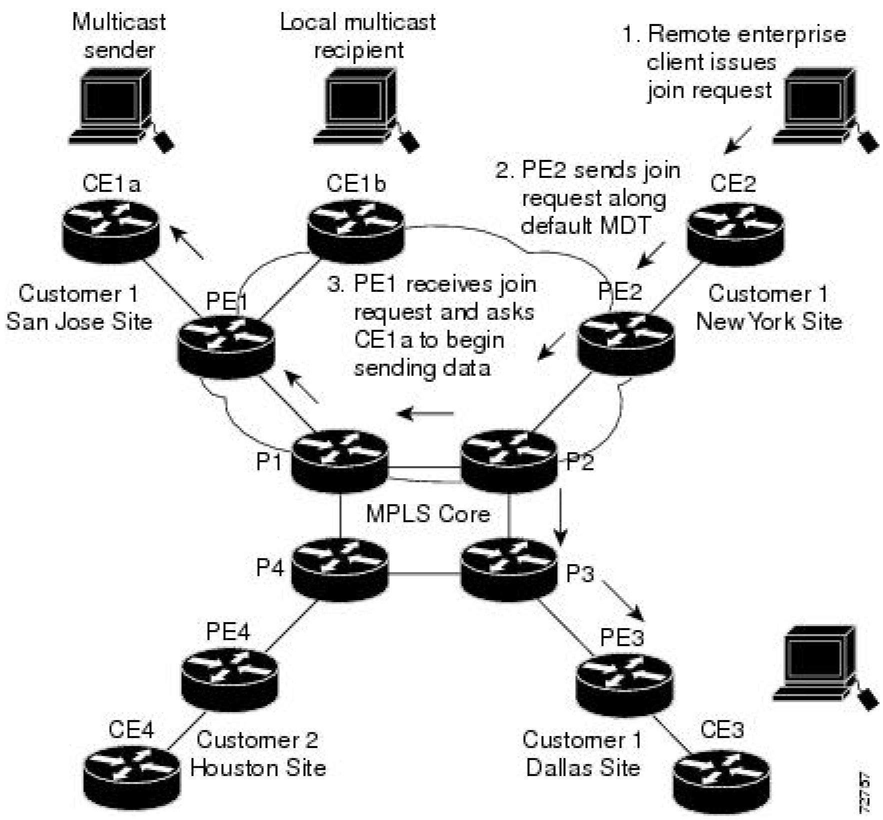
In the following example, a service provider has a multicast customer with offices in San Jose, New York, and Dallas. A one-way multicast presentation is occurring in San Jose. The service provider network supports all three sites that are associated with this customer, in addition to the Houston site of a different enterprise customer. The default MDT for the enterprise customer consists of provider routers P1, P2, and P3 and their associated PE routers. PE4 is not part of the default MDT, because it is associated with a different customer. The following figure shows that no data flows along the default MDT, because no one outside of San Jose has joined the multicast.

Figure 13: Default Multicast Distribution Tree Overview



An employee in New York joins the multicast session. The PE router that is associated with the New York site sends a join request that flows across the default MDT for the multicast domain of the customer. PE1, the PE router that is associated with the multicast session source, receives the request. The following figure depicts that the PE router forwards the request to the CE router that is associated with the multicast source (CE1a).

Figure 14: Initializing the Data MDT



The CE router (CE1a) begins to send the multicast data to the associated PE router (PE1), which sends the multicast data along the default MDT. PE1 creates a data MDT, sends a message to all routers using the default MDT that contains information about the data MDT, and, three seconds later, begins sending the multicast data for that particular stream using the data MDT. Only PE2 has interested receivers for this source, so only PE2 joins the data MDT and receives traffic on it. (If the data MDT had not been configured and only the default MDT had been configured, all the customer sites would have received the traffic even though they were not interested in it.) PE routers maintain a PIM relationship with other PE routers over the default MDT and a PIM relationship with its directly attached P routers.

Multicast Tunnel Interface

An MVPN routing and forwarding (MVRF), which is created per multicast domain, requires the router to create a tunnel interface from which all MVRF traffic is sourced. A multicast tunnel interface is an interface that the MVRF uses to access the multicast domain. The interface is a conduit that connects an MVRF and the global MVRF. One tunnel interface is created per MVRF.

Benefits of MVPNs

The benefits of MVPNs are as follows:

- Provides a scalable method to dynamically send information to multiple locations.
- Provides high-speed information delivery.
- Provides connectivity through a shared infrastructure.

BGP Advertisement Method - MVPN Support

When you configure the default MDT in a PIM Source Specific Multicast (PIM-SSM) environment rather than a PIM-SM environment, the receiver PE needs information about the source PE and the default MDT. This information is used to send (S, G) joins toward the source PE to build a distribution tree from the source PE without the need for a rendezvous point (RP). The source provider edge (PE) address and default MDT address are sent using the Border Gateway Protocol (BGP).

BGP MDT SAFI

BGP MDT SAFI is the BGP advertisement method that is used for MVPNs. In the current release, only IPv4 is supported. MDT SAFI has the following settings:

- AFI = 1
- SAFI = 66

In Cisco NX-OS, the source PE address and the MDT address are passed to PIM using BGP MDT SAFI updates. The Route Descriptor (RD) type has changed to RD type 0 and BGP determines the best path for the MDT updates before passing the information to PIM.

You must configure the MDT SAFI address family for BGP neighbors by using the **address-family ipv4 mdt** command. You must still enable neighbors that do not support the MDT SAFI for the MDT SAFI in the local BGP configuration. Prior to the MDT SAFI, additional BGP configuration from the VPNv4 unicast configuration was not needed to support MVPNs.

Prerequisites for MVPNs

MVPNs configuration has the following prerequisites:

- Ensure that you have configured MPLS and Label Distribution Protocol (LDP) in your network. All routers in the core, including the PE routers, must be able to support MPLS forwarding. VPNv4 routes are not installed by BGP if labeled paths do not exist for PE source addresses.
- Ensure that you have installed the correct license for MPLS and any other features you will be using with MPLS.

Guidelines and Limitations for MVPNs

Configuring MVPNs has the following guidelines and limitations:

- MVPNs are supported beginning with Cisco NX-OS Release 9.3(3).
- MVPNs are supported only for Cisco Nexus 9500 platform switches with -R/-RX line cards (except the N9K-X96136YC-R line card).
- Bidirectional Forwarding Detection (BFD) is not supported on the Multicast Tunnel Interface (MTI).
- By default, the BGP update source is used as the source of the MVPN tunnel. However, you can use the mdt source to override the BGP update source and provide a different source to the multicast tunnel.
- MVPN supports a maximum of 16 MDT source interfaces.
- You must configure the MDT SAFI on all routers that participate in the MVPN operations.
- Extended communities are needed for VPNv4 interior BGP (iBGP) sessions to carry the connector attribute.
- MDT MTU configuration is not supported. The maximum customer multicast packet size that can be sent over MVPN is limited by the MTU of the core interfaces. For example:
    - MTU 1500 – Customer IP packet size = 1476
    - MTU 9216 – Customer IP packet size = 9192
- Some of the MVPN multicast control packets are classified into the copp-system-p-class-l2-default CoPP policy. We recommend modifying the CoPP policy to increase the policer rate under this class if the violated count increases.
- MDT bidir-enable is not supported.
- vPCs are not supported for MVPN.
- Data MDT entries are not cached when the transit PE router does not have receivers and is connected to a CE which is a RP. The data MDT entries are cached only when a local receiver is attached to this PE router. However, there is a delay in the switchover because the entries are not pre-downloaded.
- For Date MDT, only 'immediate-switch' mode is supported. Threshold based switching is not supported.
- Sub-interface and SVI support between PE and P /PE devices is not available.
- MVPN Consistency-checker is not supported in Cisco Nexus Release 9.3(3).
- Statistics for MTI interfaces are not supported in Cisco Nexus Release 9.3(3).
- Maximum 40G multicast traffic per ASIC is supported in Cisco Nexus Release 9.3(3).
- You are allowed to configure a non-default MTU on a VRF only after you remove the MDT MTU configuration from the VRF. This occurs when the MTI is down in a switch in which the VRF with the non-default MDT MTU is available.
- Due to a hardware limitation, the MTI TX packet counts are not supported. However, all MTI RX packet and byte counts are supported.

Default Settings for MVPNs

Table 8: Default MVPN Parameters

Parameters	Default
mdt default *address*	No default
mdt enforce-bgp-mdt-safi	Enabled
mdt source	No default
mdt ip pim hello-interval *interval*	30000 ms
mdt ip pim jp-interval *interval*	60000 ms
mdt default asm-use-shared-tree	Disabled

Configuring MVPNs

This chapter describes how to configure multicast virtual private networks (MVPNs) on Cisco NX-OS devices.

Note For MVPN, a new TCAM region "ing-mvpn" is used (with default size of 10). This region is carved automatically hence you need not carve it. To verify if this TCAM region is carved or not, you can use the following commands:

```
switch# show hardware access-list tcam region | i ing-mvpn
Ingress mVPN [ing-mvpn] size = 10
switch#
```

If the region is not carved due to any reason (size shows is 0), you can use the following command to carve the TCAM region to size 10 and reload the device. The TCAM is expected to be carved to size 10.

```
switch (config)# hardware access-list tcam region ing-mvpn 10
WARNING: On module 2,
WARNING: On module 4,
Warning: Please reload all linecards for the configuration to take effect
switch (config)#
```

Enabling MVPNs

Beginning with Cisco NX-OS Release 9.3(3), you can configure MVPNs on Cisco Nexus 9500-R switches.

Before you begin

You must install and enable the MPLS feature set using the **install feature-set mpls** and **feature-set mpls** commands.

Procedure

	Command or Action	Purpose

Step 1	**configure terminal**
	Example:
	`switch#configure terminal`
	`switch(config)#`
		Enters global configuration mode.

Step 2	**feature bgp** **Example:** `switch(config)#feature bgp`	Enables BGP feature and configurations.
Step 3	**feature pim** **Example:** `switch(config)#feature pim`	Enables the PIM feature.
Step 4	**feature mvpn** **Example:** `switch(config)#feature mvpn`	Enables the MVPN feature.
Step 5	**feature mpls l3vpn** **Example:** `switch(config)#feature mpls l3vpn`	Enables the MPLS Layer 3 VPN feature. This determines the unicast routes across sites.
Step 6	**feature mpls ldp** **Example:** `switch(config)#feature mpls ldp`	Enables the MPLS Label Distribution Protocol (LDP).

Enabling PIM on Interfaces

You can configure Protocol Independent Multicast (PIM) on all interfaces that are used for IP multicast. We recommend that you configure PIM sparse mode on all physical interfaces of provider edge (PE) routers that connect to the backbone. We also recommend that you configure PIM sparse mode on all loopback interfaces if they are used for BGP peering or if their IP address is used as an RP address for PIM.

Procedure

	Command or Action	Purpose

Step 1	**configure terminal**
	Example:
	`switch#configure terminal`
	`switch(config)#`
		Enters global configuration mode.

Step 2	**ip pim sparse-mode** **Example:** `switch(config)#ip pim sparse-mode`	Enables PIM sparse mode on the interface.

Configuring a Default MDT for a VRF

You can configure a default MDT for a VRF.

Before you begin

The default MDT must be the same that is configured on all routers that belong to the same VPN. The source IP address is the address that you use to source the BGP sessions.

Procedure

	Command or Action	Purpose

Step 1	**configure terminal**
	Example:
	`switch#configure terminal`
	`switch(config)#`
		Enters global configuration mode.

Step 2	**vrf context** *VRF_NAME* **Example:** `switch(config)#vrf context vrf1`	Configures the VRF.
Step 3	**mdt default** *address* **Example:** `switch(config)#mdt default 232.0.0.1`	Configures the multicast address range for data MDTs for a VRF as follows: • A tunnel interface is created as a result of this command. • By default, the destination address of the tunnel header is the address argument.

Configuring MDT SAFI for a VRF

By default, MDT subsequent address family identifiers (SAFI) for a VRF are enforced. If desired, you can configure MDT to interoperate with peers that do not support MDT SAFI.

Procedure

	Command or Action	Purpose

Step 1	**configure terminal**
	Example:
	`switch#configure terminal`
	`switch(config)#`
		Enters global configuration mode.

Step 2	**vrf context** *VRF_NAME*
	Example:
	`switch(config)#vrf context vrf1`
	`switch(config-vrf)#`
		Configures the VRF.

	Command or Action	Purpose
Step 3	**no mdt enforce-bgp-mdt-safi** **Example:** `switch(config-vrf)#no mdt enforce-bgp-mdt-safi`	Enables MDT to interoperate with peers that do not support MDT SAFI. Initially only the (*,G) entry for the default MDT group is populated if it falls within the Any Source Multicast (ASM) range. Then later, based on traffic, the (S,G) entries are learned like regular ASM routes. Removing the **no** option from the command enforces the use of MDT SAFI for the specified VRF.

Configuring the MDT Address Family in BGP for MVPNs

You can configure an MDT address family session on PE routers to establish MDT peering sessions for MVPNs.

Use the **address-family ipv4 mdt** command under neighbor mode to configure an MDT address-family session. MDT address-family sessions are used to pass the source PE address and MDT address to PIM using BGP MDT Subaddress Family Identifier (SAFI) updates.

Before you begin

Before MVPN peering can be established through an MDT address family, you must configure MPLS in the BGP network and multiprotocol BGP on PE routers that provide VPN services to CE routers.

Procedure

	Command or Action	Purpose

Step 1	**configure terminal**
	Example:
	`switch#configure terminal`
	`switch(config)#`
		Enters global configuration mode.

Step 2	**feature bgp** *as-number* **Example:** `switch(config)#feature bgp 65635`	Enters switch configuration mode and creates a BGP routing process.

Step 3	**vrf context** *VRF_NAME*
	Example:
	`switch(config)#vrf context vpn1`
	`switch(config-vrf)#`
		Defines a VPN routing instance identified by vrf-name and enters VRF configuration mode. The vrf-name argument is any case-sensitive, alphanumeric string up to 32 characters.

Step 4	**rd** *route-distinguisher* **Example:** `switch(config-vrf)#rd 1.2.1`	Assigns a route distinguisher to the VRF vrf-name. The route-distinguisher argument adds an 8-byte value to an IPv4 prefix to create a VPN IPv4 prefix. You can enter an RD in either of these formats:

	Command or Action	Purpose
		• 16-bit or 32-bit AS number: your 32-bit number, for example, 1.2:3 • 32-bit IP address: your 16-bit number, for example, 192.0.2.1:1

Step 5	**address-family ipv4 unicast**
	Example:
	`switch(config-vrf)#address-family ipv4unicast`
	`switch(config-vrf-af)#`
		Specifies the IPv4 address family type and enters address family configuration mode

Step 6	**route-target import** *route-target-ext-community* **Example:** `switch(config-vrf-af)# route-target import 1.0.1`	Specifies a route-target extended community for a VRF. The **import** keyword imports routing information from the target VPN extended community. The *route-target-ext-community* argument adds the route-target extended community attributes to the VRF list of import route-target extended communities. You can enter the *route-target-ext-community* argument in either of these formats: • 16-bit or 32-bit AS number: your 32-bit number, for example, 1.2:3 • 32-bit IP address: your 16-bit number, for example, for example, 192.0.2.1:1
Step 7	**route-target export** *route-target-ext-community* **Example:** `switch(config-vrf-af)# route-target export 1.0.1`	Specifies a route-target extended community for a VRF. The **export** keyword imports routing information from the target VPN extended community. The *route-target-ext-community* argument adds the route-target extended community attributes to the VRF list of import route-target extended communities. You can enter the *route-target-ext-community* argument in either of these formats: • 16-bit or 32-bit AS number: your 32-bit number, for example, 1.2:3 • 32-bit IP address: your 16-bit number, for example, for example, 192.0.2.1:1
Step 8	**router bgp** *as-number* **Example:**	Configures a BGP routing process and enters router configuration mode. The as-number argument indicates the number of an

	Command or Action	Purpose
	`switch(config)#router bgp 1.1` `switch(config-router)#`	autonomous system that identifies the router to other BGP routers and tags the routing information passed along. The AS number can be a 16-bit integer or a 32-bit integer in the form of a higher 16-bit decimal number and a lower 16-bit decimal number in xx.xx format.
Step 9	**address-family ipv4 mdt** **Example:** `switch(config-router)#address-family ipv4 mdt`	Enters IPv4 MDT address family configuration mode.

Step 10	**address-family** {**vpn4**} [**unicast**]
	Example:
	`switch(config-router-af)# address-family vpnv4`
	`switch(config-router-af)#`
		Enters address family configuration mode for configuring routing sessions, such as BGP, that use standard VPNv4 or VPNv6 address prefixes. The optional **unicast** keyword specifies VPNv4 or VPNv6 unicast address prefixes.

Step 11	**address-family** {**ipv4**} **unicast**
	Example:
	`switch(config-router-af)# address-family ipv4 unicast`
	`switch(config-router-af)#`
		Enters address family configuration mode for configuring routing sessions that use standard IPv4 or IPv6 address prefixes.

Step 12	**neighbor** *neighbor-address* **Example:** `switch(config-switch-af)# neighbor 192.168.1.1`	Enters neighbor configuration mode.
Step 13	**update source** *interface* **Example:** `switch(config-switch-neighbor)# update-source loopback 1`	Sets the update source as loopback1.
Step 14	**address-family ipv4 mdt** **Example:** `switch(config-router-neighbor)# address-family ipv4 mdt`	Enters address family configuration mode to create an IP MDT address family session.
Step 15	**send-community extended** **Example:** `switch(config-router-neighbor-af)#send-community extended`	Specifies that extended communities attribute should be sent to a BGP neighbor.
Step 16	**show bgp** {**ipv4**} **unicast neighbors vrf***VRF_NAME* **Example:**	Displays information about BGP neighbors. The vrf-name argument is any case-sensitive, alphanumeric string up to 32 characters.

	Command or Action	Purpose
	`switch(config-router-neighbor-af)#show bgp ipv4 unicast neighbors vrf vpn1`	
Step 17	**copy running-config startup-config** **Example:** `switch(config-router-neighbor-af)#copy running-config startup-config`	Copies the running configuration to the startup configuration.

Configuring a Data MDT

You can configure a data MDT. Multicast groups that are used to create the data MDT are dynamically chosen from a pool of configured IP addresses. If the number of streams is greater than the maximum number of data MDTs per VRF per PE, multiple streams share the same data MDT.

Before you begin

Before configuring a data MDT, you must configure the default MDT on the VRF.

Procedure

	Command or Action	Purpose

Step 1	**configure terminal**
	Example:
	`switch#configure terminal`
	`switch(config)#`
		Enters global configuration mode.

Step 2	**vrf context** *VRF_NAME* **Example:** `switch#ip vrf vrf1`	Enters VRF configuration mode and defines the VPN routing instance by assigning a VRF name.

Step 3	**mdt data** *prefix* [**immediate-switch**] [**route-map** *policy-name*]
	Example:
	`switch(config-vrf)# mdt data 225.1.1.1/32 immediate-switch route-map test`
	Example:
	`switch(config-vrf)# mdt data 225.1.1.1/32 route-map test`
		Specifies a range of values as follows:
		• The *prefix* specifies the range of addresses to be used in the data MDT pool.
		• The *policy-name* defines a policy file that defines which customer data streams should be considered for switching onto the data MDT.
		Note Entering this command with or without the immediate-switch option has the same effect.

Step 4	**exit** **Example:** `switch(config)#exit`	Returns to global configuration mode.

Verifying the MVPN Configuration

To display the MVPN configuration, perform one of the following tasks:

Table 9: Verifying the MVPN Configuration

Command	Purpose
show interface	Displays details of an interface.
show ip mroute vrf	Displays multicast routes.
show ip pim event-history mvpn	Displays the details of the MVPN event history logs.
show ip pim mdt	Displays the details of MTI tunnels created by MVPN.
show ip pim mdt receive vrf vrf-name	Displays the mapping of the customer source, the customer group to data MDT source, and the data MDT group on the receiving side.
show ip pim mdt send vrf vrf-name	Displays the mapping of the customer source, the customer group to data MDT source, and the data MDT group on the sending side.
show ip pim neighbor	Displays details of established PIM neighbors.
show ip route detail	Displays the details of the unicast routing tables.
show mvpn bgp mdt-safi	Displays the BGP MDT SAFI database in MVPN.
show mvpn mdt encap vrf *vrf*	Displays the encapsulation table in MVPN. This table indicates how MVPN packets are encapsulated when sent out on the default vrf.
show mvpn mdt route	Displays details of the default and MDT routes. This data determines how customer data and control traffic is sent on the default VRF.
show routing [**ip**] **multicast mdt encap**	Displays the encapsulation table in the MRIB. This table indicates how MVPN packets are encapsulated when sent out on the default vrf.

Configuration Examples for MVPN

The following example shows how to configure an MVPN with two contexts:

```
vrf context vpn1
  ip pim rp-address 10.10.1.2 -list 224.0.0.0/8
  ip pim ssm range 232.0.0.0/8
  rd auto
  mdt default 232.1.1.1
  mdt source loopback1
  mdt data 225.122.111.0/24 immediate-switch
```

```
vrf context vpn4
  ip pim rp-address 10.10.4.2 -list 224.0.0.0/8
  ip pim ssm range 232.0.0.0/8
  mdt default 235.1.1.1
  mdt asm-use-shared-tree
ip pim rp-address 10.11.0.2 -list 224.0.0.0/8
ip pim rp-address 10.11.0.4 -list 235.0.0.0/8
ip pim ssm range 232.0.0.0/8
```

The following example shows how to assign to the VPN routing instance a VRF named blue. The MDT default for a VPN VRF is 10.1.1.1, and the multicast address range for MDTs is 10.1.2.0 with wildcard bits of 0.0.0.3:

```
Vrf context blue
mdt data 225.122.111.0/24 immediate-switch
```

CHAPTER 11

Configuring MPLS Segment Routing OAM

This chapter describes the Multiprotocol Label Switching (MPLS) segment routing OAM functionality.

About MPLS Segment Routing OAM

MPLS segment routing (SR) has been deployed on the Cisco Nexus 9000 Series switches. As MPLS segment routing (SR) is deployed, a few diagnostic tools are required to help resolve the misconfigurations or failures in the segment routing network. Segment Routing Operations, Administration, and Maintenance (OAM) helps service providers monitor label-switched paths (LSPs) and quickly isolate forwarding problems to assist with fault detection and troubleshooting in the network.

MPLS SR OAM provides two main functions for diagnostics purposes:

1. MPLS ping
2. MPLS traceroute

The segment routing OAM feature provides support for the following FEC types:

- Ping and traceroute to SR-IGP IS-IS IPv4 prefixes. This allows validation of prefix SIDs distributed in an IS-IS SR underlay.
- Ping and traceroute to BGP IPv4 prefixes. This allows validation of prefix SIDs distributed in a BGP SR underlay.
- Ping and traceroute to Generic IPv4 prefixes. This allows validation of prefix SIDs distributed in an SR underlay agnostic to the protocol that performed the distribution. The validation is performed by checking the Unicast Routing Information Base (URIB) and Unicast Label Information Base (ULIB).
- Ping and traceroute to Nil FEC prefixes. This allows a less comprehensive data-plane-only validation for any MPLS SR prefix, with finer-grained control over the path the ping or traceroute takes. The path may be specified using an SR-TE policy name or SR-TE policy color and endpoint.

To enable MPLS OAM on Cisco Nexus 9000 Series switches, use the **feature mpls oam** CLI command. Use the **no feature mpls oam** CLI command to disable MPLS OAM on Cisco Nexus 9000 Series switches.

Segment Routing Ping

Similar to how an IP ping validates connectivity to an IP host, MPLS ping is used to validate unidirectional continuity along an MPLS Label-Switched Path (LSP). By providing a FEC representing the LSP to be validated, MPLS ping performs the following:

- Confirms that the echo requests for the FEC reach an endpoint for the LSP. Except for the Nil FEC, for all other FEC types it confirms that the endpoint is the correct egress for that FEC.
- Measures coarse round trip time.
- Measures coarse round trip delay.

The MPLS LSP ping feature is used to check the connectivity between ingress Label Switch Routers (LSRs) and egress LSRs along an LSP. MPLS LSP ping uses MPLS echo request and reply messages, similar to Internet Control Message Protocol (ICMP) echo request and reply messages, to validate an LSP. The destination IP address of the MPLS echo request packet is different from the address used to select the label stack. The destination IP address is defined as a 127.x.y.z/8 address and it prevents the IP packet from being IP switched to its destination, if the LSP is broken.

Segment Routing Traceroute

MPLS traceroute verifies forwarding and control plane at each hop of the LSP to isolate faults. Traceroute sends MPLS echo requests with monotonically increasing time-to-live (TTL), starting with TTL of 1. Upon TTL expiry, transit node processes the request in software and verifies if it has an LSP to the target FEC and intended transit node. The transit node sends echo reply containing return code specifying the result of above verification and label stack to reach the next-hop, as well as ID of the next-hop towards destination, if verification is successful. Originator processes echo reply to build the next echo request containing TTL+1. This process is repeated until the destination replies that it is the egress for the FEC.

The MPLS LSP traceroute feature is used to isolate the failure point of an LSP. It is used for hop-by-hop fault localization and path tracing. The MPLS LSP Traceroute feature relies on the expiration of the Time to Live (TTL) value of the packet that carries the echo request. When the MPLS echo request message hits a transit node, it checks the TTL value and if it is expired, the packet is passed to the control plane, else the message is forwarded. If the echo message is passed to the control plane, a reply message is generated based on the contents of the request message

Guidelines and Limitations for MPLS SR OAM

MPLS OAM Nil FEC has the following guidelines and limitations:

- MPLS OAM Nil FEC is supported on the Cisco Nexus 9300-FX platform switches.
- MPLS OAM Nil FEC is not supported on Cisco Nexus 9500 platform switches with -R line cards.
- For all new FEC types supported in Cisco NX-OS Release 9.3(1), only a one-label stack is supported. FEC-Stack change TLV support and the associated validations are not supported. This limitation is not applicable to Nil FEC.

- In Cisco NX-OS Release 9.3(1), the SR-IGP "any" prefix type and the adjacency SIDs described in RFC 8287 are not supported.
- OSPF ping and traceroute is not supported in Cisco NX-OS Release 9.3(1).
- Beginning with Cisco NX-OS Release 9.3(3), MPLS OAM Nil FEC is supported on Cisco Nexus 9300-GX platform switches.
- A maximum of 4 labels can be specified in the **ping mpls nil-fec** and **traceroute mpls nil-fec** commands. This value is enforced by querying the platform and currently Cisco Nexus 9000 Series switches limit the label stack to 5. It means that for a Nil FEC echo request, you can specify a maximum of 4 labels because internally an extra explicit-null is added.
- The nexthop specified in the ping and traceroute commands must be a connected nexthop on the originator and it should not be a recursive nexthop.
- There is no support for treetrace.
- Nil FEC does not carry any information to identify the intended target. The packet may mis-forward at an incorrect node but the validation may return success if the packet ends up at a node after popping the non-null labels.
- Nil FEC operates on forwarding the information alone. It cannot detect the inconsistencies between the control plane and the forwarding plane by definition.
- Nil FEC ping and traceroute is not supported for deaggregator (per-VRF) labels. This includes the BGP EVPN-Layer 3 deaggregator labels.
- On Cisco Nexus 9000 Series switches that use Broadcom chipsets, there is no support to allow the software to send a query to determine which ECMP a packet takes. It means that for MPLS traceroutes that traverse one of these switches may display an error at the next hop if there is more than one ECMP as displayed in the following example:

  ```
  D 2 6.0.0.2 MRU 1496 [Labels: 2003/explicit-null Exp: 0/0] 4 ms
  ```

- When you use OAM to test a BGP EPE LSP (for example, the last label in the ping/traceroute label stack is an EPE label), OAM only returns success if the final router has OAM enabled and MPLS is enabled on the incoming interface.

  For example, if you have a setup as A---B---C, A and B are in the SR network, and B acts like a PE and C acts like a CE, B is configured with C as a BGP EPE peer (using egress-engineering on B), then C must have OAM and MPLS forwarding enabled on the incoming interface.

MPLS Ping and Traceroute for Nil FEC

The Nil FEC LSP ping and traceroute operations are extensions of regular MPLS ping and traceroute. The Nil FEC LSP ping and traceroute functionality supports segment routing and MPLS Static. It also acts as an additional diagnostic tool for all other LSP types.

Unlike the other FEC types, Nil FEC does not provide control plane validation. Nil FEC ping or traceroute probes can reach any switch on which the MPLS OAM functionality is enabled.

This feature allows operators to provide the ability to freely test any label stack by allowing them to specify the following:

- Label stack

- Outgoing interface
- Nexthop address

In case of segment routing, each segment nodal label and adjacent label along the routing path is put into the label stack of an echo request message from the initiator Label Switch Router (LSR); MPLS data plane forwards this packet to the label stack target, and the label stack target sends the echo message back.

Use the [**ping**|**traceroute**] **mpls nil-fec labels** *comma-separated-labels* [**output** {**interface** *tx-interface*} [**nexthop** *nexthop-ip-addr*]] CLI command to execute a ping or a traceroute.

If you have configured an SR-TE policy name or the color and the endpoint, you can use the following CLI command to execute a ping or a traceroute to use the existing SR-TE policy information.:

[**ping**|**traceroute**] **mpls nil-fec** [**policy name** *name*] [**endpoint** *nexthop-ip-addr*] [**on-demand color** *color-num*] CLI command to execute a ping or a traceroute.

MPLS Ping and Traceroute for BGP and IGP Prefix SID

MPLS ping and traceroute operations for Prefix SID are supported for the following BGP and IGP scenarios:

- Within an IS-IS level
- Across IS-IS levels
- BGP SR underlay

These FEC types perform an additional control plane check to ensure that the packets are not mis-routed. This validation ensures that the pinged FEC type is connected to the switch and is distributed to the other nodes. Nil FEC does not provide this validation.

MPLS echo request packets carry Target FEC Stack sub-TLVs. The Target FEC sub-TLVs are used by the responder for FEC validation. The IGP/BGP IPv4 prefix sub-TLV has been added to the Target FEC Stack sub-TLV. The IGP/BGP IPv4 prefix sub-TLV contains the prefix SID, the prefix length, and the protocol (IS-IS).

Use the **ping|traceroute sr-mpls** A.B.C.D/LEN fec-type [*bgp* | *igp {isis}* | *generic*] CLI command to execute a traceroute.

Verifying Segment Routing OAM

This section provides information on the CLI commands that can be used to verify the segment routing OAM features.

- Verifying Segment Routing OAM IS-IS, on page 202

Verifying Segment Routing OAM IS-IS

The following ping commands are used to display SR OAM when the underlying network is IS-IS:

```
switch# ping sr-mpls 11.1.1.3/32 fec-type igp isis

Sending 5, 100-byte MPLS Echos to IGP Prefix SID(IS-IS) FEC 11.1.1.3/32,
```

```
     timeout is 2 seconds, send interval is 0 msec:

Codes: '!' - success, 'Q' - request not sent, '.' - timeout,
  'L' - labeled output interface, 'B' - unlabeled output interface,
  'D' - DS Map mismatch, 'F' - no FEC mapping, 'f' - FEC mismatch,
  'M' - malformed request, 'm' - unsupported tlvs, 'N' - no label entry,
  'P' - no rx intf label prot, 'p' - premature termination of LSP,
  'R' - transit router, 'I' - unknown upstream index,
  'X' - unknown return code, 'x' - return code 0

Type Ctrl-C to abort.
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 2/2/3 ms
 Total Time Elapsed 18 ms
```

```
switch# traceroute sr-mpls 11.1.1.3/32 fec-type igp isis


Codes: '!' - success, 'Q' - request not sent, '.' - timeout,
  'L' - labeled output interface, 'B' - unlabeled output interface,
  'D' - DS Map mismatch, 'F' - no FEC mapping, 'f' - FEC mismatch,
  'M' - malformed request, 'm' - unsupported tlvs, 'N' - no label entry,
  'P' - no rx intf label prot, 'p' - premature termination of LSP,
  'R' - transit router, 'I' - unknown upstream index,
  'X' - unknown return code, 'x' - return code 0

Type Ctrl-C to abort.
  0 172.18.1.2 MRU 1500 [Labels: 16103 Exp: 0]
L 1 172.18.1.1 MRU 1504 [Labels: implicit-null Exp: 0] 4 ms
! 2 172.18.1.10 3 ms
```

```
switch# ping sr-mpls 11.1.1.3/32 fec-type igp isis verbose

Sending 5, 100-byte MPLS Echos to IGP Prefix SID(IS-IS) FEC 11.1.1.3/32,
     timeout is 2 seconds, send interval is 0 msec:

Codes: '!' - success, 'Q' - request not sent, '.' - timeout,
  'L' - labeled output interface, 'B' - unlabeled output interface,
  'D' - DS Map mismatch, 'F' - no FEC mapping, 'f' - FEC mismatch,
  'M' - malformed request, 'm' - unsupported tlvs, 'N' - no label entry,
  'P' - no rx intf label prot, 'p' - premature termination of LSP,
  'R' - transit router, 'I' - unknown upstream index,
  'X' - unknown return code, 'x' - return code 0

Type Ctrl-C to abort.
!    size 100, reply addr 172.18.1.10, return code 3
!    size 100, reply addr 172.18.1.10, return code 3
!    size 100, reply addr 172.18.1.10, return code 3
!    size 100, reply addr 172.18.1.10, return code 3
!    size 100, reply addr 172.18.1.10, return code 3

Success rate is 100 percent (5/5), round-trip min/avg/max = 2/2/3 ms
 Total Time Elapsed 17 ms
```

```
switch# ping sr-mpls 11.1.1.3/32 fec-type igp isis destination 127.0.0.1 127.0.0.2 repeat
1 verbose

Sending 1, 100-byte MPLS Echos to IGP Prefix SID(IS-IS) FEC 11.1.1.3/32,
     timeout is 2 seconds, send interval is 0 msec:

Codes: '!' - success, 'Q' - request not sent, '.' - timeout,
  'L' - labeled output interface, 'B' - unlabeled output interface,
  'D' - DS Map mismatch, 'F' - no FEC mapping, 'f' - FEC mismatch,
  'M' - malformed request, 'm' - unsupported tlvs, 'N' - no label entry,
  'P' - no rx intf label prot, 'p' - premature termination of LSP,
```

```
  'R' - transit router, 'I' - unknown upstream index,
  'X' - unknown return code, 'x' - return code 0

Type Ctrl-C to abort.
Destination address 127.0.0.1
!    size 100, reply addr 172.18.1.10, return code 3

Destination address 127.0.0.2
!    size 100, reply addr 172.18.1.22, return code 3

Success rate is 100 percent (2/2), round-trip min/avg/max = 3/3/3 ms
 Total Time Elapsed 8 ms
```

Examples for using Ping and Traceroute CLI commands

Examples for IGP or BGP SR Ping and Traceroute

Using CLI to Execute a Ping with Explicit Outgoing Information

Use the **ping sr-mpls** *fec* **fec-type igp isis** CLI command to execute an IS-IS SR ping and the **ping sr-mpls** *fec* **fec-type bgp** CLI command to execute a BGP ping.

```
switch# ping sr-mpls 11.1.1.3/32 fec-type igp isis

Sending 5, 100-byte MPLS Echos to IGP Prefix SID(IS-IS) FEC 11.1.1.3/32,
     timeout is 2 seconds, send interval is 0 msec:

Codes: '!' - success, 'Q' - request not sent, '.' - timeout,
  'L' - labeled output interface, 'B' - unlabeled output interface,
  'D' - DS Map mismatch, 'F' - no FEC mapping, 'f' - FEC mismatch,
  'M' - malformed request, 'm' - unsupported tlvs, 'N' - no label entry,
  'P' - no rx intf label prot, 'p' - premature termination of LSP,
  'R' - transit router, 'I' - unknown upstream index,
  'X' - unknown return code, 'x' - return code 0

Type Ctrl-C to abort.
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 2/2/3 ms
 Total Time Elapsed 18 ms
```

```
switch# ping sr-mpls 11.1.1.3/32 fec-type igp isis verbose

Sending 5, 100-byte MPLS Echos to IGP Prefix SID(IS-IS) FEC 11.1.1.3/32,
     timeout is 2 seconds, send interval is 0 msec:

Codes: '!' - success, 'Q' - request not sent, '.' - timeout,
  'L' - labeled output interface, 'B' - unlabeled output interface,
  'D' - DS Map mismatch, 'F' - no FEC mapping, 'f' - FEC mismatch,
  'M' - malformed request, 'm' - unsupported tlvs, 'N' - no label entry,
  'P' - no rx intf label prot, 'p' - premature termination of LSP,
  'R' - transit router, 'I' - unknown upstream index,
  'X' - unknown return code, 'x' - return code 0

Type Ctrl-C to abort.
!    size 100, reply addr 172.18.1.10, return code 3
!    size 100, reply addr 172.18.1.10, return code 3
!    size 100, reply addr 172.18.1.10, return code 3
!    size 100, reply addr 172.18.1.10, return code 3
!    size 100, reply addr 172.18.1.10, return code 3
```

```
Success rate is 100 percent (5/5), round-trip min/avg/max = 2/2/3 ms
 Total Time Elapsed 17 ms
```

Examples for Nil FEC Ping and Traceroute

Using CLI to Execute a Ping with Explicit Outgoing Information

Use the **ping sr-mpls nil-fec labels** *comma-separated-labels* [**output** {**interface** *tx-interface*} [**nexthop** *nexthop-ip-addr*]] CLI command to execute a ping.

For example, the following command sends an MPLS packet with the outermost two labels in the label stack being 2001 and 2000 out the interface Ethernet 1/1 with a nexthop IP address of 4.0.0.2:

```
switch# ping mpls nil-fec labels 2001,2000 output interface e1/1 nexthop 4.0.0.2
```

It is mandatory that the nexthop is a connected nexthop; it is not recursively resolved.

The above CLI format is a simplified version. The [**output** {**interface** *tx-interface*} [**nexthop** *nexthop-ip-addr*]] is mandatory to be present in the VSH server. For example:

```
switch# ping mpls nil-fec labels 1,2 ?
output Output options
switch# ping mpls nil-fec labels1,2
^
% Invalid command at '^' marker.
```

Using CLI to Execute a Ping with Outgoing Information from an SRTE Policy

Use the following CLI command to execute a ping:

```
switch# ping mpls nil-fec policy name policy1
switch# ping mpls nil-fec policy endpoint 2.0.0.1 color 16
```

Using CLI to Execute a Traceroute with Explicit Outgoing Information

Use the following CLI command to execute a traceroute:

```
switch# ping mpls nil-fec labels 2001,2000 output interface e1/1 nexthop 4.0.0.2
```

Using CLI to Execute a Traceroute with Outgoing Information from an SRTE Policy

Use the following CLI command to execute a traceroute:

```
switch# traceroute mpls nil-fec policy name policy1
switch# traceroute mpls nil-fec policy endpoint 2.0.0.1 color 16
```

Displaying Show Statistics

Use the following command to display the statistics about the echo requests sent by the local MPLS OAM service:

```
show mpls oam echo statistics
```

CHAPTER 12

InterAS Option B

This chapter explains the different InterAS option B configuration options. The available options are InterAS option B, InterAS option B (with RFC 3107), and InterAS option B lite. The InterAS option B (with RFC 3107) implementation ensures complete IGP isolation between the data centers and WAN. When BGP advertises a particular route to ASBR, it also distributes the label which is mapped to that route.

Information About InterAS

An autonomous system (AS) is a single network or group of networks that is controlled by a common system administration group and using a single, clearly defined protocol. In many cases, virtual private networks (VPNs) extend to different ASes in different geographical areas. Some VPNs must extend across multiple service providers; these VPNs are called overlapping VPNs. The connection between ASes must be seamless to the customer, regardless of the complexity or location of the VPNs.

InterAS and ASBR

Separate ASes from different service providers can communicate by exchanging information in the form of VPN IP addresses. The ASBRs use EBGP to exchange that information. The IBGP distributes the network layer information for IP prefixes throughout each VPN and each AS. The following protocols are used for sharing routing information:

- Within an AS, routing information is shared using IBGP.
- Between ASes, routing information is shared using EBGP. EBGP allows service providers to set up an interdomain routing system that guarantees loop-free exchange of routing information between separate ASes.

The primary function of EBGP is to exchange network reachability information between ASes, including information about the list of AS routes. The ASes use EBGP border edge routers to distribute the routes, which includes label-switching information. Each border edge router rewrites the next-hop and MPLS labels.

InterAS configuration supported in this MPLS VPN can include an interprovider VPN, which is MPLS VPNs that include two or more ASes, connected by separate border edge routers. The ASes exchange routes use EBGP, and no IBGP or routing information is exchanged between the ASes.

Exchanging VPN Routing Information

ASes exchange VPN routing information (routes and labels) to establish connections. To control connections between ASes, the PE routers and EBGP border edge routers maintain a label forwarding information base (LFIB). The LFIB manages the labels and routes that the PE routers and EBGP border edge routers receive during the exchange of VPN information.

The ASes use the following guidelines to exchange VPN routing information:

- Routing information includes:
  - The destination network.
  - The next-hop field associated with the distributing router.
  - A local MPLS label
- A route distinguisher (RD1) is part of a destination network address. It makes the VPN IP route globally unique in the VPN service provider environment.

The ASBRs are configured to change the next-hop when sending VPN NLRIs to the IBGP neighbors. Therefore, the ASBRs must allocate a new label when they forward the NLRI to the IBGP neighbors.

InterAS Options

Nexus 9508 series switches support the following InterAS options:

- **InterAS option A** - In an interAS option A network, autonomous system border router (ASBR) peers are connected by multiple subinterfaces with at least one interface VPN that spans the two ASes. These ASBRs associate each subinterface with a VPN routing and forwarding (VRF) instance and a BGP session to signal unlabeled IP prefixes. As a result, traffic between the back-to-back VRFs is IP. In this scenario, the VPNs are isolated from each other and, because the traffic is IP Quality of Service (QoS) mechanisms that operate on the IP traffic can be maintained. The downside of this configuration is that one BGP session is required for each subinterface (and at least one subinterface is required for each VPN), which causes scalability concerns as the network grows.
- **InterAS option B** - In an interAS option B network, ASBR ports are connected by one or more subinterfaces that are enabled to receive MPLS traffic. A Multiprotocol Border Gateway Router (MP-BGP) session distributes labeled VPN prefixes between the ASBRs. As a result, the traffic that flows between the ASBRs is labeled. The downside of this configuration is that, because the traffic is MPLS, QoS mechanisms that are applied only to IP traffic cannot be carried and the VRFs cannot be isolated. InterAS option B provides better scalability than option A because it requires only one BGP session to exchange all VPN prefixes between the ASBRs. Also, this feature provides nonstop forwarding (NSF) and Graceful Restart. The ASBRs must be directly connected in this option.

Some functions of option B are noted below:

- You can have an IBGP VPNv4/v6 session between Nexus 9508 series switches within an AS and you can have an EBGP VPNv4/v6 session between data center edge routers and WAN routers.

- There is no requirement for a per VRF IBGP session between data center edge routers, like in the lite version.
- – LDP distributes IGP labels between ASBRs.

- **InterAS option B (with BGP-3107 or RFC 3107 implementation)**
- You can have an IBGP VPNv4/v6 implementation between Nexus 9508 switches within an AS and you can have an EBGP VPNv4/v6 session between data center edge routers and WAN routers.
- BGP-3107 enables BGP packets to carry label information without using LDP between ASBRs.
- The label mapping information for a particular route is piggybacked in the same BGP update message that is used to distribute the route itself.
- When BGP is used to distribute a particular route, it also distributes an MPLS label which is mapped to that route. Many ISPs prefer this method of configuration since it ensures complete IGP isolation between the data centers.

- **InterAS option B lite** – Support for the InterAS option B feature is restricted in the Cisco NX-OS 6.2(2) release. Details are noted in the Configuring InterAS Option B (lite version) section.

Guidelines and Limitations for Configuring InterAS Option B

InterAS Option B has the following guidelines and limitations:

- InterAS option B is not supported with BGP confederation AS.
- InterAS option B is supported on Cisco Nexus 9500 platform switches with -R line cards.

Configuring BGP for InterAS Option B

Configure DC Edge switches with IBGP & EBGP VPNv4/v6 with the following steps:

Before you begin

To configure BGP for InterAS option B, you need to enable this configuration on both the IBGP and EBGP sides. Refer to Figure 1 for reference.

Procedure

	Command or Action	Purpose

Step 1	**configure terminal**
	Example:
	`switch# configure terminal`
	`switch(config)#`
		Enters global configuration mode.

	Command or Action	Purpose
Step 2	**router bgp** *as-number* **Example:** `switch(config)# router bgp 100`	Enters the router BGP configuration mode and assigns an autonomous system (AS) number to the local BGP speaker device.
Step 3	**neighbor** *ip-address* **Example:** `switch(config-router)# neighbor 10.0.0.2`	Adds an entry to the BGP or multiprotocol BGP neighbor table, and enters router BGP neighbor configuration mode.
Step 4	**remote-as** *as-number* **Example:** `switch(config-router-neighbor)# remote-as 200`	The as-number argument specifies the autonomous system to which the neighbor belongs.
Step 5	**address-family {vpnv4 | vpnv6} unicast** **Example:** `switch(config-router-neighbor)# address-family vpnv4 unicast`	Enters address family configuration mode for configuring IP VPN sessions.
Step 6	**send-community {both | extended}** **Example:** `switch(config-router-neighbor-af)# send-community both`	Specifies that a communities attribute should be sent to both BGP neighbors.
Step 7	**retain route-target all** **Example:** `switch(config-router-neighbor-af)# retain route-target all`	(Optional). Retains VPNv4/v6 address configuration on the ASBR without VRF configuration. **Note** If you have a VRF configuration on the ASBR, this command is not required.
Step 8	**vrf** *vrf-name* **Example:** `switch(config-router-neighbor-af)# vrf VPN1`	Associates the BGP process with a VRF.
Step 9	**address-family {ipv4 | ipv6} unicast** **Example:** `switch(config-router-vrf)# address-family ipv4 unicast`	Specifies the IPv4 or IPv6 address family and enters address family configuration mode.
Step 10	**exit** **Example:** `switch(config-vrf-af)# exit`	Exits IPv4 address family.

	Command or Action	Purpose
Step 11	**copy running-config startup-config** **Example:** `switch(config-router-vrf)# copy running-config startup-config`	(Optional) Copies the running configuration to the startup configuration.

Configuring BGP for InterAS Option B (with RFC 3107 implementation)

Configure DC Edge switches with IBGP & EBGP VPNv4/v6 along with BGP labeled unicast family with following steps:

Procedure

	Command or Action	Purpose

Step 1	**configure terminal**
	Example:
	`switch# configure terminal`
	`switch(config)#`
		Enters global configuration mode.

Step 2	**router bgp** *as-number* **Example:** `switch(config)# router bgp 100`	Enters the router BGP configuration mode and assigns an autonomous system (AS) number to the local BGP speaker device.
Step 3	**address-family {vpnv4 | vpnv6} unicast** **Example:** `switch(config-router-neighbor)# address-family vpnv4 unicast`	Enters address family configuration mode for configuring IP VPN sessions.
Step 4	**redistribute direct route-map** *tag* **Example:** `switch(config-router-af)# redistribute direct route-map loopback`	Redistributes directly connected routes using the Border Gateway Protocol.
Step 5	**allocate-label all** **Example:** `switch(config-router-af)# allocate-label all`	Configures ASBRs with the BGP labeled unicast address family to advertise labels for the connected interface.
Step 6	**exit** **Example:** `switch(config-router-af)# exit`	Exits address family router configuration mode and enters router BGP configuration mode.

	Command or Action	Purpose
Step 7	**neighbor** *ip-address* **Example:** `switch(config-router)# neighbor 10.1.1.1`	Configures the BGP neighbor's IP address, and enters router BGP neighbor configuration mode.
Step 8	**remote-as** *as-number* **Example:** `switch(config-router-neighbor)# remote-as 100`	Specifies the BGP neighbor's AS number.
Step 9	**address-family {ipv4|ipv6} labeled-unicast** **Example:** `switch(config-router-neighbor)# address-family ipv4 labeled-unicast`	Configures the ASBR with the BGP labeled unicast address family to advertise labels for the connected interface. **Note** This is the command that implements RFC 3107.
Step 10	**retain route-target all** **Example:** `switch(config-router-neighbor-af)# retain route-target all`	(Optional). Retains VPNv4/v6 address configuration on the ASBR without VRF configuration. **Note** If you have a VRF configuration on the ASBR, this command is not required.
Step 11	**exit** **Example:** `Switch(config-router-neighbor-af)# exit`	Exits router BGP neighbor address family configuration mode and returns to router BGP configuration mode.
Step 12	**neighbor** *ip-address* **Example:** `switch(config-router)# neighbor 10.1.1.1`	Configures a loopback IP address, and enters router BGP neighbor configuration mode.
Step 13	**remote-as** *as-number* **Example:** `switch(config-router-neighbor)# remote-as 100`	Specifies the BGP neighbor's AS number.
Step 14	**address-family {vpnv4|vpnv6} unicast** **Example:** `switch(config-router-vrf)# address-family ipv4 unicast`	Configures the ASBR with the BGP VPNv4 unicast address family.
Step 15	**exit** **Example:** `switch(config-vrf-af)# exit`	Exits IPv4 address family.

	Command or Action	Purpose
Step 16	**address-family {vpnv4|vpnv6} unicast** **Example:** `switch(config-router-vrf)# address-family ipv4 unicast`	Configures the ASBR with the BGP VPNv4 unicast address family.
Step 17	**Repeat the process with ASBR2**	Configures ASBR2 with option B (RFC 3107) settings and implements complete IGP isolation between the two data centers DC1 and DC2.
Step 18	**copy running-config startup-config** **Example:** `switch(config-router-vrf)# copy running-config startup-config`	(Optional) Copies the running configuration to the startup configuration.

CHAPTER 13

IETF RFCs Supported for Label Switching

This appendix lists the IETF RFCs supported for label switching on the device.

IETF RFCs Supported for Label Switching

This table lists the IETF RFCs supported for label switching on the device.

RFCs	Title
RFC 3107	*Carrying Label Information in BGP-4*
RFC 7752	*North-Bound Distribution of Link-State and Traffic Engin Information Using BGP*
RFC 8029	*Detecting Multiprotocol Label Switched (MPLS) Data-Pl*
RFC 8287	*Label Switched Path (LSP) Ping/Traceroute for Segment I IGP-Prefix and IGP-Adjacency Segment Identifiers (SIDs Data Planes.*
Draft-ietf-idr-bgpls-segment-routing-epe-05	*Segment Routing BGP Egress Peer Engineering BGP-LS draft-ietf-idr-bgpls-segment-routing-epe-05*

IETF RFCs Supported for Label Switching

INDEX