12
IPv4 Multicast VPN Support

Note

- For complete syntax and usage information for the commands used in this chapter, see these publications:

  http://www.cisco.com/en/US/products/ps11846/prod_command_reference_list.html
- Cisco IOS Release 15.4SY supports only Ethernet interfaces. Cisco IOS Release 15.4SY does not support any WAN features or commands.

Tip

For additional information about Cisco Catalyst 6500 Series Switches (including configuration examples and troubleshooting information), see the documents listed on this page:

http://www.cisco.com/en/US/products/hw/switches/ps708/tsd_products_support_series_home.html

Participate in the Technical Documentation Ideas forum

Prerequisites for mVPNs

None.

Restrictions for mVPNs

General Restrictions

- All PE routers in the multicast domain need to be running a Cisco IOS software image that supports the mVPN feature. There is no requirement for mVPN support on the P and CE routers.
- Support for IPv4 multicast traffic must be enabled on all backbone routers.
- The Border Gateway Protocol (BGP) routing protocol must be configured and operational on all routers supporting multicast traffic. In addition, BGP extended communities must be enabled (using the **neighbor send-community both** or **neighbor send-community extended** command) to support the use of MDTs in the network.
- When the switch is acting as a PE, and receives a multicast packet from a customer router with a time-to-live (TTL) value of 2, it drops the packet instead of encapsulating it and forwarding it across the mVPN link. Because such packets would normally be dropped by the PE at the other end of the mVPN link, this does not affect traffic flow.
- If the core multicast routing uses SSM, then the data and default multicast distribution tree (MDT) groups must be configured within the SSM range of IPv4 addresses.
- The update source interface for the BGP peerings must be the same for all BGP peerings configured on the router in order for the default MDT to be configured properly. If you use a loopback address for BGP peering, then PIM sparse mode must be enabled on the loopback address.
- The **ip mroute-cache** command must be enabled on the loopback interface used as the BGP peering interface in order for distributed multicast switching to function on the platforms that support it. The **no ip mroute-cache** command must *not* be present on these interfaces.
- Data MDTs are not created for VRF PIM dense mode multicast streams because of the flood and prune nature of dense mode multicast flows and the resulting periodic bring-up and tear-down of such data MDTs.
- Data MDTs are not created for VRF PIM bidirectional mode because source information is not available.
- mVPN does not support multiple BGP peering update sources, and configuring them can break mVPN RPF checking. The source IPv4 address of the mVPN tunnels is determined by the highest IPv4 address used for the BGP peering update source. If this IPv4 address is not the IPv4 address used as the BGP peering address with the remote PE router, mVPN will not function properly.
- MDT tunnels do not carry unicast traffic.
- If mVPN uses the infrastructure of an MPLS VPN network, you cannot apply MPLS tags or labels to multicast traffic over the VPNs.
- Each mVRF that is configured with a default MDT uses three hidden VLANs (one each for encapsulation, decapsulation, and interface), in addition to external, user-visible VLANs. This means that an absolute maximum of 1,000 mVRFs are supported on each router. (mVRFs without a configured MDT still use one internal VLAN, so unused mVRFs should be deleted to conserve VLAN allocation.)
- If your MPLS VPN network already contains a network of VRFs, you do not need to delete them or recreate them to be able to support mVRF traffic. Instead, configure the **mdt default** and **mdt data** commands, as listed in the following procedure, to enable multicast traffic over the VRF.
- The same mVRF must be configured on each PE router that is to support a particular VPN connection.
- Each PE router that supports a particular mVRF must be configured with the same **mdt default** command.

mVPN with L3VPN over mGRE Restrictions

- In releases earlier than Release 15.1(1)SY, With mVPN with L3VPN over mGRE configured, do not configure IPv4 routing on the supervisor engine ports or on ports on switching modules that have a CFC. (CSCtr05033)
- Ensure that the unicast path to the RP does not use any supervisor engine ports; additionally, in VSS mode, ensure that the unicast path to the RP does not use any ports on switching modules that have a CFC. (CSCts43614)
- When a GRE tunnel has the same destination address and source address as the mGRE tunnel, the GRE tunnel is route-cache switched.
- Packets that require fragmentation are route cache-switched.
- When an L3VPN profile is removed and added back, you should clear the Border Gateway Protocol (BGP) using the **clear ip bgp** *neighbor_ip_address* **soft** command.
- When an mGRE tunnel is created, a dummy tunnel is also created.
- The loopback or IP address used in the update source of the BGP configuration should be the same as that of the transport source of the L3VPN profile.
- mGRE is not stateful switchover (SSO) compliant. However, both mGRE and SSO coexist.
- Not all GRE options are supported in hardware (for example, the GRE extended header and GRE key).
- Checking identical VLANs (Internet Control Message Protocol [ICMP] redirect) is not supported on the tunnels.
- Features such as unicast reverse path forwarding (uRPF) and BGP policy accounting are not supported on the tunnels.

Information About mVPN

mVPN Overview

mVPN is a standards-based feature that transmits IPv4 multicast traffic across across a virtualized provider network (for example, MPLS or mGRE tunnels). mVPN uses the IPv4 multicast traffic PFC hardware support to forward multicast traffic over VPNs at wire speeds. mVPN adds support for IPv4 multicast traffic over Layer 3 IPv4 VPNs to the existing IPv4 unicast support.

mVPN routes and forwards multicast packets for each individual VPN routing and forwarding (VRF) instance, as well as transmitting the multicast packets through VPN tunnels across the service provider backbone.

mVPN is an alternative to full-mesh point-to-point GRE tunnels, which is not a readily scalable solution and are limited in the granularity they provide to customers.

Multicast Routing and Forwarding and Multicast Domains

mVPN adds 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, forwarding is performed according to the information in the multicast VRF (mVRF).

Each mVRF maintains the routing and forwarding information that is needed for its particular VRF instance. An mVRF is created and configured in the same way as existing VRFs, except multicast routing is also enabled on each mVRF.

A multicast domain constitutes the set of hosts that can send multicast traffic to each other within the MPLS network. 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 associated with that enterprise.

Multicast Distribution Trees

The mVPN feature establishes at least one multicast distribution tree (MDT) for each multicast domain. The MDT provides the information needed to interconnect the same mVRFs that exist on the different PE routers.

mVPN supports two MDT types:

- Default MDT—The default MDT is a permanent channel for PIM control messages and low-bandwidth streams between all PE routers in a particular multicast domain. All multicast traffic in the default MDT is replicated to every other PE router in the domain. Each PE router is logically seen as a PIM neighbor (one hop away) from every other PE router in the domain.
- Data MDT—Data MDTs are optional. If enabled, they are dynamically created to provide optimal paths for high-bandwidth transmissions, such as full-motion video, that do not need to be sent to every PE router. This allows for on-demand forwarding of high-bandwidth traffic between PE routers, so as to avoid flooding every PE router with every high-bandwidth stream that might be created.

To create data MDTs, each PE router that is forwarding multicast streams to the backbone periodically examines the traffic being sent in each default MDT as follows:

1. Each PE router periodically samples the multicast traffic (approximately every 10 seconds for software switching, and 90 seconds for hardware switching) to determine whether a multicast stream has exceeded the configured threshold. (Depending on when the stream is sampled, this means that in a worst-case scenario, it could take up to 180 seconds before a high-bandwidth stream is detected.)

   Note Data MDTs are created only for (S, G) multicast route entries within the VRF multicast routing table. They are not created for (*, G) entries.

2. If a particular multicast stream exceeds the defined threshold, the sending PE router dynamically creates a data MDT for that particular multicast traffic.

3. The sending PE router then transmits a DATA-MDT JOIN request (which is a User Datagram Protocol (UDP) message to port 3232) to the other PE routers, informing them of the new data MDT.
4. Receiving PE routers examine their VRF routing tables to determine if they have any customers interested in receiving this data stream. If so, they use the PIM protocol to transmit a PIM JOIN message for this particular data MDT group (in the global table PIM instance) to accept the stream. Routers that do not currently have any customers for this stream still cache the information, in case any customers request it later on.
5. Three seconds after sending the DATA-MDT JOIN message, the sending PE router removes the high-bandwidth multicast stream from the default MDT and begins transmitting it over the new data MDT.
6. The sending PE router continues to send a DATA-MDT JOIN message every 60 seconds, as long as the multicast stream continues to exceed the defined threshold. If the stream falls below the threshold for more than 60 seconds, the sending PE router stops sending the DATA-MDT JOIN messages, and moves the stream back to the default MDT.
7. Receiving routers age out the cache information for the default MDT when they do not receive a DATA-MDT JOIN message for more than three minutes.

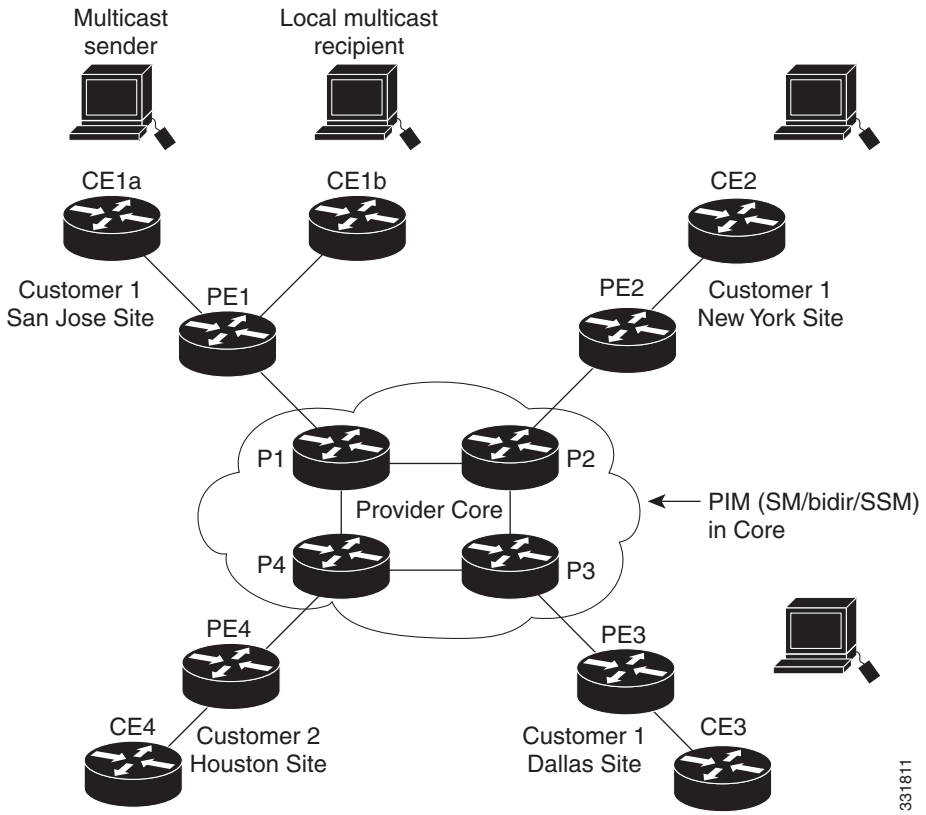
Data MDTs allow for high-bandwidth sources inside the VPN while still ensuring optimal traffic forwarding in the MPLS VPN core.

In the following example, a service provider has a multicast customer with offices in San Jose, New York, and Dallas. The San Jose site is transmitting a one-way multicast presentation. The service provider network supports all three sites 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. Although PE4 is interconnected to these other routers in the MPLS core, PE4 is associated with a different customer and is therefore not part of the default MDT.

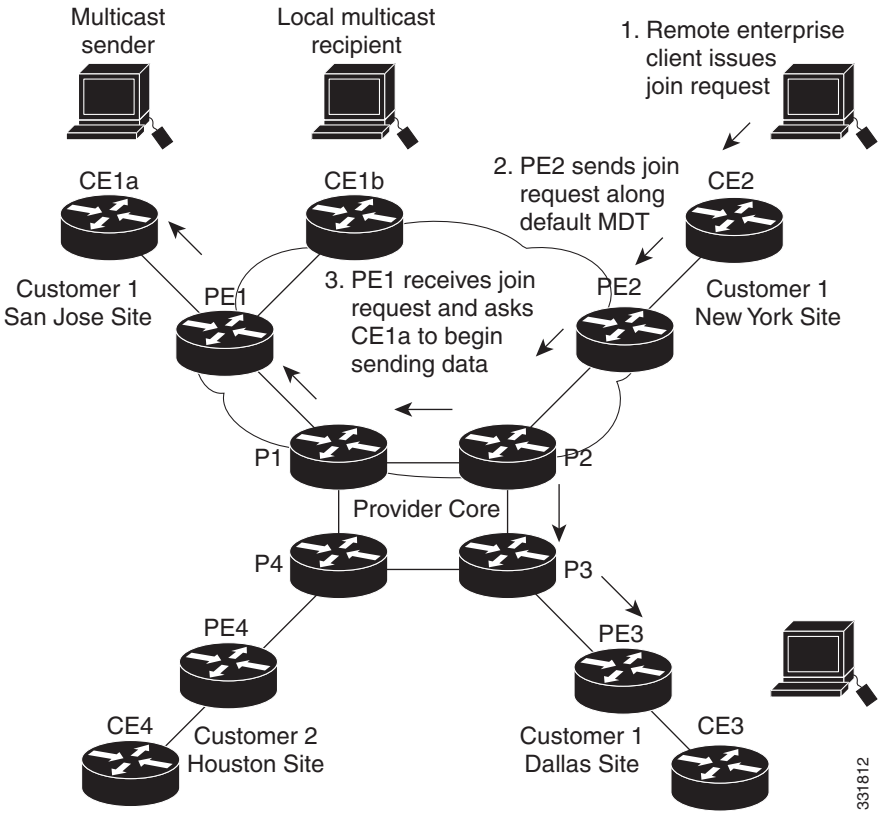
Figure 12-1 shows the situation in this network when no one outside of San Jose has joined the multicast broadcast, which means that no data is flowing along the default MDT. Each PE router maintains a PIM relationship with the other PE routers over the default MDT, as well as a PIM relationship with its directly attached PE routers.

Figure 12-1 Default Multicast Distribution Tree Overview



If an employee in New York joins the multicast session, the PE router associated for the New York site sends a join request that flows across the default MDT for the multicast domain. The PE router associated with the multicast session source (PE1) receives the request. Figure 12-2 shows how the PE router forwards the request to the CE router associated with the multicast source (CE1a).

Figure 12-2 Initializing the Data MDT



The CE router (CE1a) starts sending the multicast data to the associated PE router (PE1), which recognizes that the multicast data exceeds the bandwidth threshold at which a data MDT should be created. PE1 then creates a data MDT and sends a message to all routers using the default MDT that contains information about the data MDT.

Approximately three seconds later, PE1 begins sending the multicast data for that particular stream using the data MDT. Because only PE2 has receivers who are interested in this source, only PE2 joins the data MDT and receives traffic on it.

Multicast Tunnel Interfaces

The PE router creates a multicast tunnel interface (MTI) for each multicast VRF (mVRF) in the multicast domain. The mVRF uses the tunnel interface to access the multicast domain to provide a conduit that connects an mVRF and the global mVRF.

On the router, the MTI is a tunnel interface (created with the **interface tunnel** command) with a class D multicast address. All PE routers that are configured with a default MDT for this mVRF create a logical network in which each PE router appears as a PIM neighbor (one hop away) to every other PE router in the multicast domain, regardless of the actual physical distance between them.

The MTI is automatically created when an mVRF is configured. The BGP peering address is assigned as the MTI interface source address, and the PIM protocol is automatically enabled on each MTI.

When the router receives a multicast packet from the customer side of the network, it uses the incoming interface's VRF to determine which mVRFs should receive it. The router then encapsulates the packet using GRE encapsulation. When the router encapsulates the packet, it sets the source address to that of the BGP peering interface and sets the destination address to the multicast address of the default MDT, or to the source address of the data MDT if configured. The router then replicates the packet as needed for forwarding on the appropriate number of MTI interfaces.

When the router receives a packet on the MTI interface, it uses the destination address to identify the appropriate default MDT or data MDT, which in turn identifies the appropriate mVRF. It then decapsulates the packet and forwards it out the appropriate interfaces, replicating it as many times as are necessary.

Note

- Unlike other tunnel interfaces that are commonly used on Cisco routers, the mVPN MTI is classified as a LAN interface, not a point-to-point interface. The MTI interface is not configurable, but you can use the **show interface tunnel** command to display its status.
- The MTI interface is used exclusively for multicast traffic over the VPN tunnel.
- The tunnel does not carry unicast routed traffic.

PE Router Routing Table Support for mVPN

Each PE router that supports the mVPN feature uses the following routing tables to ensure that the VPN and mVPN traffic is routed correctly:

- Default routing table—Standard routing table used in all Cisco routers. This table contains the routes that are needed for backbone traffic and for non-VPN unicast and multicast traffic (including Generic Routing Encapsulation (GRE) multicast traffic).
- VPN routing/forwarding (VRF) table—Routing table created for each VRF instance. Responsible for routing the unicast traffic between VPNs in the provider network.
- Multicast VRF (mVRF) table—Multicast routing table and multicast routing protocol instance created for each VRF instance. Responsible for routing the multicast traffic in the multicast domain of the network. This table also includes the multicast tunnel interfaces that are used to access the multicast domain.

Multicast Distributed Switching Support

mVPN supports multicast distributed switching (MDS) for multicast support on a per-interface and a per-VRF basis. When configuring MDS, you must make sure that no interface (including loopback interfaces) has the **no ip mroute-cache** command configured.

Hardware-Assisted IPv4 Multicast

Cisco IOS Release 15.4SY supports hardware acceleration for IPv4 multicast over VPN traffic, which forwards multicast traffic to the appropriate VPNs at wire speed without increased RP CPU utilization.

In a customer VRF, PFC hardware acceleration supports multicast traffic in PIM dense, PIM sparse, PIM bidirectional, and PIM Source Specific Multicast (SSM) modes.

In the service provider core, PFC hardware acceleration supports multicast traffic in PIM sparse, PIM bidirectional, and PIM SSM modes. In the service provider core, PFC hardware acceleration does not support multicast traffic in PIM dense mode.

Information About mVPN with L3VPN over mGRE

Note See the "Configuring mVPN with L3VPN over mGRE" section on page 12-22 for more information.

Overview

Release 15.0(1)SY1 and later releases support multicast virtual private network with Layer 3 virtual private network over multipoint generic routing encapsulation (mVPN with L3VPN over mGRE). mVPN with L3VPN over mGRE provides VPN connectivity between networks that are connected by standards-based IP-only networks. mGRE tunnels overlay the IP network and connect PE devices to transport VPNs that support deployment of L3 PE-based VPN services over the IP core.

Note

- Because mGRE is a point-to-multipoint model, fully meshed GRE tunnels are not required to interconnect PE devices.
- Multicast and unicast traffic use separate tunnels, MDT for multicast and mGRE for unicast.

Route Maps

By default, VPN unicast traffic is sent using an LSP. The mVPN with L3VPN over mGRE feature uses user-defined route maps to determine which VPN prefixes are reachable over an mGRE tunnel and which VPN prefixes are reachable using an LSP. The route map is applied to advertisements for VPNv4 and VPNv6 address families. The route map uses a next hop tunnel table to determine the encapsulation method for the VPN traffic.

Traffic routed over an mGRE tunnel uses an alternative address space; all next hops are reached by encapsulating the traffic in an mGRE tunnel. To configure a specific route to use an mGRE tunnel, requires configuration of an entry for that route to the route map. The new entry remaps the Network Layer Reachability Information (NLRI) of the route to the alternative address space. If there is no remap entry in the route map for a route, then traffic on that route is forwarded over an LSP.

The mVPN with L3VPN over mGRE feature automatically provisions the alternative address space, normally held in the tunnel-encapsulated virtual routing and forwarding (VRF) instance. To ensure that all traffic reachable through the address space is encapsulated in an mGRE tunnel, the feature installs a single default route out of a tunnel. The feature also creates a default tunnel on the route map. The default route map can be attached to the appropriate BGP updates.

Tunnel Endpoint Discovery and Forwarding

The mVPN with L3VPN over mGRE feature must be able to discover the remote PEs in the network and construct tunnel forwarding information for the remote PEs. The feature must be able to detect when a remote PE is no longer valid and remove the tunnel forwarding information for that PE.

If an ingress PE receives a VPN advertisement over BGP, it uses the route target attributes (which it inserts into the VRF) and the MPLS VPN label from the advertisement, to associate the prefixes with the appropriate customer. The next hop of the inserted route is set to the NLRI of the advertisement.

The advertised prefixes contain information about remote PEs in the system (in the form of NLRIs), and the PE uses this information to notify the system when an NLRI becomes active or inactive. The system uses this notification to update the PE forwarding information.

When the feature receives notification of a new remote PE, it adds the information to the tunnel endpoint database and creates an adjacency associated with the tunnel interface. The adjacency description includes information on the encapsulation and other processing required to send encapsulated packets to the new remote PE.

The feature places the adjacency information into the tunnel encapsulated VRF. When a VPN NLRI is remaped to a route in the VRF (using the route map), the feature links the NLRI to the adjacency, which links the VPN to the tunnel.

Tunnel Decapsulation

When the egress PE receives a packet from a tunnel interface that uses themVPN with L3VPN over mGRE feature, the PE decapsulates the packet to create a VPN label tagged packet, and forwards the packet.

Tunnel Source

The mVPN with L3VPN over mGRE feature uses a single tunnel configured as an mGRE tunnel to configure a system with a large number of endpoints (remote PEs). To identify the origin of tunnel-encapsulated packets, the system uses the tunnel source information.

At the transmitting (ingress) PE, when a VPN packet is sent to a tunnel, the tunnel destination is the NLRI. At a receiving (egress) PE, the tunnel source is the address that the packets encapsulated in the mGRE tunnel are received on. Therefore, at the egress PE the packet destination must match the NLRI from the local PE.

Default Settings for mVPNs

None.

How to Configure mVPNs

Note These configuration tasks assume that BGP is already configured and operational on all routers that are sending or receiving the multicast traffic. In addition, BGP extended communities must be enabled (using the **neighbor send-community both** or **neighbor send-community extended** command) to support the use of MDTs in the network.

Configuring a Multicast VPN Routing and Forwarding Instance

Configuring a VRF Entry

To configure a VRF entry, perform this task:

	Command	Purpose
Step 1	`Router# configure terminal`	Enters global configuration mode.
Step 2	`Router(config)# ip vrf` *vrf_name*	Configures a VRF routing table entry and a Cisco Express Forwarding (CEF) table entry and enters VRF configuration mode.
Step 3	`Router(config-vrf)# do show ip vrf` *vrf_name*	Verifies the configuration.

This example show how to configure a VRF named blue and verify the configuration:

```
Router# configure terminal
Router(config)# ip vrf blue
Router(config-vrf)# do show ip vrf blue
Name                              Default RD          Interfaces
  blue                              <not set>
```

Configuring the Route Distinguisher

To configure the route distinguisher, perform this task:

	Command	Purpose
Step 1	`Router(config-vrf)# rd route_distinguisher`	Specifies the route distinguisher for a VPN IPv4 prefix.
Step 2	`Router(config-vrf)# do show ip vrf vrf_name`	Verifies the configuration.

When configuring the route distinguisher, enter the route distinguisher in one of the following formats:

- 16-bit AS number:your 32-bit number (101:3)
- 32-bit IPv4 address:your 16-bit number (192.168.122.15:1)

This example show how to configure 55:1111 as the route distinguisher and verify the configuration:

```
Router(config-vrf)# rd 55:1111
Router(config-vrf)# do show ip vrf blue
Name                              Default RD          Interfaces
blue                              55:1111
```

Configuring the Route-Target Extended Community

To configure the route-target extended community, perform this task:

	Command	Purpose
Step 1	`Router(config-vrf)# route-target [import | export | both] route_target_ext_community`	Configures a route-target extended community for the VRF.
Step 2	`Router(config-vrf)# do show ip vrf detail`	Verifies the configuration.

When configuring the route-target extended community, note the following information:

- **import**—Imports routing information from the target VPN extended community.
- **export**—Exports routing information to the target VPN extended community.
- **both**—Imports and exports.
- *route_target_ext_community*—Adds the 48-bit route-target extended community to the VRF. Enter the number in one of the following formats:
  - 16-bit AS number:your 32-bit number (101:3)
  - 32-bit IPv4 address:your 16-bit number (192.168.122.15:1)

This example shows how to configure 55:1111 as the import and export route-target extended community and verify the configuration:

```
Router(config-vrf)# route-target both 55:1111
Router(config-vrf)# do show ip vrf detail
VRF blue; default RD 55:1111; default VPNID <not set>
VRF Table ID = 1
  No interfaces
  Connected addresses are not in global routing table
  Export VPN route-target communities
    RT:55:1111
  Import VPN route-target communities
    RT:55:1111
  No import route-map
  No export route-map
CSC is not configured.
```

Configuring the Default MDT

To configure the default MDT, perform this task:

Command	Purpose
`Router(config-vrf)# mdt default group_address`	Configures the default MDT.

When configuring the default MDT, note the following information:

- The *group_address* is the multicast IPv4 address of the default MDT group. This address serves as an identifier for the mVRF community, because all provider-edge (PE) routers configured with this same group address become members of the group, which allows them to receive the PIM control messages and multicast traffic that are sent by other members of the group.
- This same default MDT must be configured on each PE router to enable the PE routers to receive multicast traffic for this particular mVRF.

This example shows how to configure 239.1.1.1 as the default MDT:

```
Router(config-vrf)# mdt default 239.1.1.1
```

Configuring Data MDTs (Optional)

To configure optional data MDTs, perform this task:

Command	Purpose
`Router(config-vrf)# mdt data group_address wildcard_bits [threshold threshold_value] [list access_list]`	(Optional) Configures a data MDTs for the specified range of multicast addresses.

When configuring optional data MDTs, note the following information:

- *group_address1*—Multicast group address. The address can range from 224.0.0.1 to 239.255.255.255, but cannot overlap the address that has been assigned to the default MDT.
- *wildcard_bits*—Wildcard bit mask to be applied to the multicast group address to create a range of possible addresses. This allows you to limit the maximum number of data MDTs that each mVRF can support.

- **threshold** *threshold_value*—(Optional) Defines the threshold value in kilobits, at which multicast traffic should be switched from the default MDT to the data MDT. The *threshold_value* parameter can range from 1 through 4294967 kilobits.
- **list** *access_list*—(Optional) Specifies an access list name or number to be applied to this traffic.

This example shows how to configure a data MDT:

```
Router(config-vrf)# mdt data 239.1.2.0 0.0.0.3 threshold 10
```

Enabling Data MDT Logging

To enable data MDT logging, perform this task:

Command	Purpose
`Router(config-vrf)# mdt log-reuse`	(Optional) Enables the recording of data MDT reuse information, by generating a SYSLOG message whenever a data MDT is reused. Frequent reuse of a data MDT might indicate a need to increase the number of allowable data MDTs by increasing the size of the wildcard bitmask that is used in the **mdt data** command.

This example shows how to enable data MDT logging:

```
Router(config-vrf)# mdt log-reuse
```

Sample Configuration

The following excerpt from a configuration file shows typical VRF configurations for a range of VRFs. To simplify the display, only the starting and ending VRFs are shown.

```
!
ip vrf mvpn-cus1
 rd 200:1
 route-target export 200:1
 route-target import 200:1
 mdt default 239.1.1.1
!
ip vrf mvpn-cus2
 rd 200:2
 route-target export 200:2
 route-target import 200:2
 mdt default 239.1.1.2
!
ip vrf mvpn-cus3
 rd 200:3
 route-target export 200:3
 route-target import 200:3
 mdt default 239.1.1.3
!
...

ip vrf mvpn-cus249
 rd 200:249
 route-target export 200:249
 route-target import 200:249
 mdt default 239.1.1.249
 mdt data 239.1.1.128 0.0.0.7
```

Displaying VRF Information

To display all of the VRFs that are configured on the switch, use the **show ip vrf** command:

```
Router# show ip vrf

  Name                              Default RD          Interfaces
  green                             1:52                GigabitEthernet6/1
  red                               200:1               GigabitEthernet1/1
                                                        GigabitEthernet3/16
                                                        Loopback2

Router#
```

To display information about the MDTs that are currently configured for all mVRFs, use the **show ip pim mdt** command. The following example shows typical output for this command:

```
Router# show ip pim mdt

  MDT Group       Interface   Source                VRF
* 227.1.0.1       Tunnel1     Loopback0             BIDIR01
* 227.2.0.1       Tunnel2     Loopback0             BIDIR02
* 228.1.0.1       Tunnel3     Loopback0             SPARSE01
* 228.2.0.1       Tunnel4     Loopback0             SPARSE02
```

Note To display information about a specific tunnel interface, use the **show interface tunnel** command. The IPv4 address for the tunnel interface is the multicast group address for the default MDT of the mVRF.

To display routing information for a particular VRF, use the **show ip route vrf** command:

```
Router# show ip route vrf red

Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS inter area
       * - candidate default, U - per-user static route, o - ODR
       P - periodic downloaded static route

Gateway of last resort is not set

     2.0.0.0/32 is subnetted, 1 subnets
C       2.2.2.2 is directly connected, Loopback2
     3.0.0.0/32 is subnetted, 1 subnets
B       3.3.3.3 [200/0] via 3.1.1.3, 00:20:09
C    21.0.0.0/8 is directly connected, GigabitEthernet3/16
B    22.0.0.0/8 [200/0] via 3.1.1.3, 00:20:09

Router#
```

To display information about the multicast routing table and tunnel interface for a particular mVRF, use the **show ip mroute vrf** command. The following example shows typical output for a mVRF named BIDIR01:

```
Router# show ip mroute vrf BIDIR01

IP Multicast Routing Table
Flags: D - Dense, S - Sparse, B - Bidir Group, s - SSM Group, C - Connected,
       L - Local, P - Pruned, R - RP-bit set, F - Register flag,
       T - SPT-bit set, J - Join SPT, M - MSDP created entry,
```

```
       X - Proxy Join Timer Running, A - Candidate for MSDP Advertisement,
       U - URD, I - Received Source Specific Host Report, Z - Multicast Tunnel
       Y - Joined MDT-data group, y - Sending to MDT-data group
Outgoing interface flags: H - Hardware switched
 Timers: Uptime/Expires
 Interface state: Interface, Next-Hop or VCD, State/Mode
(*, 228.1.0.1), 00:16:25/stopped, RP 10.10.10.12, flags: SJCF
  Incoming interface: Tunnel1, RPF nbr 10.10.10.12, Partial-SC
  Outgoing interface list:
    GigabitEthernet3/1.3001, Forward/Sparse-Dense, 00:16:25/00:02:49, H
(6.9.0.100, 228.1.0.1), 00:14:13/00:03:29, flags: FT
  Incoming interface: GigabitEthernet3/1.3001, RPF nbr 0.0.0.0, RPF-MFD
  Outgoing interface list:
    Tunnel1, Forward/Sparse-Dense, 00:14:13/00:02:46, H

Router#
```

Note In this example, the **show ip mroute vrf** command shows that Tunnel1 is the MDT tunnel interface (MTI) being used by this VRF.

Configuring Multicast VRF Routing

Note BGP should be already configured and operational on all routers that are sending or receiving multicast traffic. In addition, BGP extended communities must be enabled (using the **neighbor send-community both** or **neighbor send-community extended** command) to support the use of MDTs in the network.

Enabling IPv4 Multicast Routing Globally

To enable IPv4 multicast routing globally, perform this task:

	Command	Purpose
Step 1	`Router# configure terminal`	Enters global configuration mode.
Step 2	`Router(config)# ip multicast-routing`	Enables IPv4 multicast routing globally.

This example show how to enable IPv4 multicast routing globally:

```
Router# configure terminal
Router(config)# ip multicast-routing
```

Enabling IPv4 Multicast VRF Routing

To enable IPv4 multicast VRF routing, perform this task:

Command	Purpose
`Router(config)# ip multicast-routing vrf vrf_name [distributed]`	Enables IPv4 multicast VRF routing.

When enabling IPv4 multicast VRF routing, note the following information:

- *vrf_name*—Specifies a particular VRF for multicast routing. The *vrf_name* should see a VRF that has been previously created, as specified in the "Configuring a Multicast VPN Routing and Forwarding Instance" section on page 12-11.
- **distributed**—(Optional) Enables Multicast Distributed Switching (MDS).

This example show how to enable IPv4 multicast VRF routing:

```
Router# configure terminal
Router(config)# ip multicast-routing vrf blue
```

Specifying the PIM VRF RP Address

To specify the PIM VRF rendezvous point (RP) address, perform this task:

Command	Purpose
`Router(config)# ip pim vrf vrf_name rp-address rp_address [access_list] [override] [bidir]`	Specifies the PIM RP IPv4 address for a (required for sparse PIM networks):

When specifying the PIM VRF RP address, note the following information:

- **vrf** *vrf_name*—(Optional) Specifies a particular VRF instance to be used.
- *rp_address*—Unicast IP address for the PIM RP router.
- *access_list*—(Optional) Number or name of an access list that defines the multicast groups for the RP.
- **override**—(Optional) In the event of conflicting RP addresses, this particular RP overrides any RP that is learned through Auto-RP.
- **bidir**—(Optional) Specifies that the multicast groups specified by the *access_list* argument are to operate in bidirectional mode. If this option is not specified, the groups operate in PIM sparse mode.
- Use bidirectional mode whenever possible, because it offers better scalability.

This example show how to specify the PIM VRF RP address:

```
Router(config)# ip pim vrf blue rp-address 198.196.100.33
```

Configuring a PIM VRF Register Message Source Address (Optional)

To configure a PIM VRF register message source address, perform this task:

Command	Purpose
`Router(config)# ip pim vrf vrf_name register-source interface_type interface_number`	(Optional) Configures a PIM VRF register message source address. You can configure a loopback interface as the source of the register messages.

This example show how to configure a PIM VRF register message source address:

```
Router(config)# ip pim vrf blue register-source loopback 3
```

Configuring an MSDP Peer (Optional)

To configure a multicast source discovery protocol (MSDP) peer, perform this task:

Command	Purpose
`Router(config)# ip msdp vrf vrf_name peer {peer_name | peer_address} [connect-source interface_type interface_number] [remote-as ASN]`	(Optional) Configures an MSDP peer.

When configuring an MSDP peer, note the following information:

- **vrf** *vrf_name*—Specifies a particular VRF instance to be used.
- {*peer_name* | *peer_address*}—Domain Name System (DNS) name or IP address of the MSDP peer router.
- **connect-source** *interface_type interface_number*—Interface name and number for the interface whose primary address is used as the source IP address for the TCP connection.
- **remote-as** *ASN*—(Optional) Autonomous system number of the MSDP peer. This is for display-only purposes.

This example show how to configure an MSDP peer:

```
Router(config)# ip msdp peer router.cisco.com connect-source gigabitethernet 1/1 remote-as
109
```

Configuring the Maximum Number of Multicast Routes (Optional)

To configure the maximum number of multicast routes, perform this task:

Command	Purpose
`Router(config)# ip multicast vrf vrf_name route-limit limit [threshold]`	(Optional) Configures the maximum number of multicast routes that can be added for multicast traffic.

When configuring the maximum number of routes, note the following information:

- **vrf** *vrf_name*— Enables route limiting for the specified VRF.
- *limit*—The number of multicast routes that can be added. The range is from 1 to 2147483647, with a default of 2147483647.
- *threshold*—(Optional) Number of multicast routes that can be added before a warning message occurs. The valid range is from 1 to the value of the *limit* parameter.

This example show how to configure the maximum number of multicast routes:

```
Router(config)# ip multicast vrf blue route-limit 200000 20000
```

Configuring IPv4 Multicast Route Filtering (Optional)

To configure IPV4 multicast route filtering, perform this task:

Command	Purpose
`Router(config)# ip multicast mrinfo-filter` *access_list*	(Optional) Configures IPV4 multicast route filtering with an access list. The *access_list* parameter can be the name or number of a access list.

This example show how to configure IPV4 multicast route filtering:

```
Router(config)# ip multicast mrinfo-filter 101
```

Sample Configuration

The following excerpt from a configuration file shows the minimum configuration that is needed to support multicast routing for a range of VRFs. To simplify the display, only the starting and ending VRFs are shown.

```
!
ip multicast-routing
ip multicast-routing vrf lite
ip multicast-routing vrf vpn201
ip multicast-routing vrf vpn202

...

ip multicast-routing vrf vpn249
ip multicast-routing vrf vpn250


...

ip pim rp-address 192.0.1.1
ip pim vrf lite rp-address 104.1.1.2
ip pim vrf vpn201 rp-address 192.200.1.1
ip pim vrf vpn202 rp-address 192.200.2.1

...

ip pim vrf vpn249 rp-address 192.200.49.6
ip pim vrf vpn250 rp-address 192.200.50.6
...
```

Displaying IPv4 Multicast VRF Routing Information

To display the known PIM neighbors for a particular mVRF, use the **show ip pim vrf neighbor** command:

```
Router# show ip pim vrf 98 neighbor

PIM Neighbor Table
Neighbor          Interface                Uptime/Expires    Ver   DR
Address                                                            Prio/Mode
40.60.0.11        Tunnel96                 00:00:31/00:01:13 v2    1 / S
40.50.0.11        Tunnel96                 00:00:54/00:00:50 v2    1 / S

Router#
```

Configuring Interfaces for Multicast Routing to Support mVPN

Multicast Routing Configuration Overview

Protocol Independent Multicast (PIM) must be configured on all interfaces that are being used for IPv4 multicast traffic. In a VPN multicast environment, you should enable PIM on at least all of the following interfaces:

- Physical interface on a provider edge (PE) router that is connected to the backbone.
- Loopback interface that is used for BGP peering.
- Loopback interface that is used as the source for the sparse PIM rendezvous point (RP) router address.

In addition, you must also associate mVRFs with those interfaces over which they are going to forward multicast traffic.

BGP should be already configured and operational on all routers that are sending or receiving multicast traffic. In addition, BGP extended communities must be enabled (using the **neighbor send-community both** or **neighbor send-community extended** command) to support the use of MDTs in the network.

Configuring PIM on an Interface

To configure PIM on an interface, perform this task:

	Command	Purpose
Step 1	`Router# configure terminal`	Enters global configuration mode.
Step 2	`Router(config)# interface` *type* {*slot/port* | *number*}	Enters interface configuration mode for the specified interface.
Step 3	`Router(config-if)# ip pim` {**dense-mode** | **sparse-mode** | **sparse-dense-mode**}	Enables PIM on the interface.

When configuring PIM on an interface, note the following information:

- You can use one of these interface types:
  - A physical interface on a provider edge (PE) router that is connected to the backbone.
  - A loopback interface that is used for BGP peering.
  - A loopback interface that is used as the source for the sparse PIM network rendezvous point (RP) address.
- These are the PIM modes:
  - **dense-mode**—Enables dense mode of operation.
  - **sparse-mode**—Enables sparse mode of operation.
  - **sparse-dense-mode**—Enables sparse mode if the multicast group has an RP router defined, or enables dense mode if an RP router is not defined.
- Use **sparse-mode** for the physical interfaces of all PE routers that are connected to the backbone, and on all loopback interfaces that are used for BGP peering or as the source for RP addressing.

This example shows how to configure PIM sparse mode on a physical interface:

```
Router# configure terminal
Router(config)# interface gigabitethernet 10/1
Router(config-if)# ip pim sparse-mode
```

This example shows how to configure PIM sparse mode on a loopback interface:

```
Router# configure terminal
Router(config)# interface loopback 2
Router(config-if)# ip pim sparse-mode
```

Configuring an Interface for IPv4 VRF Forwarding

To configure an interface for IPv4 VRF forwarding, perform this task:

Command	Purpose
`Router(config-if)# ip vrf forwarding` *vrf_name*	(Optional) Associates the specified VRF routing and forwarding tables with the interface. If this is not specified, the interface defaults to using the global routing table. **Note** Entering this command on an interface removes the IP address, so reconfigure the IP address.

This example shows how to configure the interface for VRF blue forwarding:

```
Router(config-if)# ip vrf forwarding blue
```

Sample Configuration

The following excerpt from a configuration file shows the interface configuration, along with the associated mVRF configuration, to enable multicast traffic over a single mVRF:

```
ip multicast-routing vrf blue
ip multicast-routing

ip vrf blue
 rd 100:27
 route-target export 100:27
 route-target import 100:27
 mdt default 239.192.10.2

interface GigabitEthernet1/1
 description blue connection
 ip vrf forwarding blue
 ip address 192.168.2.26 255.255.255.0
 ip pim sparse-mode

interface GigabitEthernet1/15
 description Backbone connection
 ip address 10.8.4.2 255.255.255.0
 ip pim sparse-mode

ip pim vrf blue rp-address 192.7.25.1
ip pim rp-address 10.1.1.1
```

Configuring mVPN with L3VPN over mGRE

- Configuring an L3VPN Encapsulation Profile, page 12-22 (required)
- Configuring BGP and Route Maps, page 12-23 (required)

Note See the "Information About mVPN with L3VPN over mGRE" section on page 12-9 for more information.

Configuring an L3VPN Encapsulation Profile

Note Transport protocols such as IPv6, MPLS, IP, and Layer 2 Tunneling Protocol version 3 (L2TPv3) can also be used in this configuration.

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**l3vpn encapsulation ip** *profile-name* **Example:** `Router(config)# l3vpn encapsulation ip tunnel encap`	Enters L3 VPN encapsulation configuration mode to create the tunnel.
Step 4	**transport ipv4 [source** *interface-type interface-number***]** **Example:** `Router(config-l3vpn-encap-ip)# transport ipv4 source loopback 0`	(Optional) Specifies IPv4 transport source mode and defines the transport source interface. • If you use the **transport ipv4 source** *interface-type interface-number* command, make sure that the specified source address is used as the next hop in BGP updates advertised by the PE. • If you do not use this command, the **bgp update source** or **bgp next-hop** command is automatically used as the tunnel source.
Step 5	**protocol gre [key** *gre-key***]** **Example:** `Router(config-l3vpn-encap-ip)# protocol gre key 1234`	Specifies GRE as the tunnel mode and sets the GRE key.
Step 6	**end** **Example:** `Router(config-l3vpn-encap-ip)# end`	Exits L3 VPN encapsulation configuration mode and returns to privileged EXEC mode.
Step 7	**show l3vpn encapsulation ip** *profile-name* **Example:** `Router# show l3vpn encapsulation ip tunnel encap`	(Optional) Displays the profile health and the underlying tunnel interface.

Configuring BGP and Route Maps

Perform this task to configure BGP and route maps. The following steps also enable you to link the route map to the application template and set up the BGP VPNv4 and VPNv6 exchange so that the updates are filtered through the route map.

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**router bgp** *as-number* **Example:** `Router(config)# router bgp 100`	Specifies the number of an autonomous system that identifies the router to other BGP routers and tags the routing information passed along, and enters router configuration mode.
Step 4	**bgp log-neighbor-changes** **Example:** `Router(config-router)# bgp log-neighbor-changes`	Enables logging of BGP neighbor resets.
Step 5	**neighbor** *ip-address* **remote-as** *as-number* **Example:** `Router(config-router)# neighbor 209.165.200.225 remote-as 100`	Adds an entry to the BGP or multiprotocol BGP neighbor table.
Step 6	**neighbor** *ip-address* **update-source** *interface name* **Example:** `Router(config-router)# neighbor 209.165.200.225 update-source loopback 0`	Allows BGP sessions to use any operational interface for TCP connections.
Step 7	**address-family ipv4** **Example:** `Router(config-router)# address-family ipv4`	Enters address family configuration mode to configure routing sessions that use IPv4 address prefixes.
Step 8	**no synchronization** **Example:** `Router(config-router-af)# no synchronization`	Enables the Cisco IOS software to advertise a network route without waiting for an IGP.
Step 9	**redistribute connected** **Example:** `Router(config-router-af)# redistribute connected`	Redistributes routes from one routing domain into another routing domain and allows the target protocol to redistribute routes learned by the source protocol and connected prefixes on those interfaces over which the source protocol is running.
Step 10	**neighbor** *ip-address* **activate** **Example:** `Router(config-router-af)# neighbor 209.165.200.225 activate`	Enables the exchange of information with a BGP neighbor.

	Command or Action	Purpose
Step 11	**no auto-summary** **Example:** `Router(config-router-af)# no auto-summary`	Disables automatic summarization and sends subprefix routing information across classful network boundaries.
Step 12	**exit** **Example:** `Router(config-router-af)# exit`	Exits address family configuration mode.
Step 13	**address-family vpnv4** **Example:** `Router(config-router)# address-family vpnv4`	Enters address family configuration mode to configure routing sessions, such as BGP, that use standard VPNv4 address prefixes.
Step 14	**neighbor** *ip-address* **activate** **Example:** `Router(config-router-af)# neighbor 209.165.200.225 activate`	Enables the exchange of information with a BGP neighbor.
Step 15	**neighbor** *ip-address* **send-community both** **Example:** `Router(config-router-af)# neighbor 209.165.200.225 send-community both`	Specifies that a communities attribute, for both standard and extended communities, should be sent to a BGP neighbor.
Step 16	**neighbor** *ip-address* **route-map** *map-name* **in** **Example:** `Router(config-router-af)# neighbor 209.165.200.225 route-map SELECT_UPDATE_FOR_L3VPN in`	Applies the named route map to the incoming route.
Step 17	**exit** **Example:** `Router(config-router-af)# exit`	Exits address family configuration mode.
Step 18	**address-family vpnv6** **Example:** `Router(config-router)# address-family vpnv6`	Enters address family configuration mode to configure routing sessions, such as BGP, that use VPNv6 address prefixes.
Step 19	**neighbor** *ip-address* **activate** **Example:** `Router(config-router-af)# neighbor 209.165.200.252 activate`	Enables the exchange of information with a BGP neighbor.
Step 20	**neighbor** *ip-address* **send-community both** **Example:** `Router(config-router-af)# neighbor 209.165.200.252 send-community both`	Specifies that a communities attribute, for both standard and extended communities, should be sent to a BGP neighbor.

	Command or Action	Purpose
Step 21	**neighbor** *ip-address* **route-map** *map-name* **in** **Example:** `Router(config-router-af)# neighbor 209.165.200.252 route-map SELECT_UPDATE_FOR_L3VPN in`	Applies the named route map to the incoming route.
Step 22	**exit** **Example:** `Router(config-router-af)# exit`	Exits address family configuration mode.
Step 23	**route-map** *map-tag* **permit** *position* **Example:** `Router(config-router)# route-map SELECT_UPDATE_FOR_L3VPN permit 10`	Enters route-map configuration mode and defines the conditions for redistributing routes from one routing protocol into another. • The **redistribute** router configuration command uses the specified map tag to reference this route map. Multiple route maps may share the same map tag name. • If the match criteria are met for this route map, the route is redistributed as controlled by the set actions. • If the match criteria are not met, the next route map with the same map tag is tested. If a route passes none of the match criteria for the set of route maps sharing the same name, it is not redistributed by that set. • The *position* argument indicates the position a new route map will have in the list of route maps already configured with the same name.
Step 24	**set ip next-hop encapsulate l3vpn** *profile-name* **Example:** `Router(config-route-map)# set ip next-hop encapsulate l3vpn my profile`	Indicates that output IPv4 packets that pass a match clause of the route map are sent to the VRF for tunnel encapsulation.
Step 25	**set ipv6 next-hop encapsulate l3vpn** *profile-name* **Example:** `Router(config-route-map)# set ip next-hop encapsulate l3vpn tunnel encap`	Indicates that output IPv6 packets that pass a match clause of the route map are sent to the VRF for tunnel encapsulation.
Step 26	**exit** **Example:** `Router(config-route-map)# exit`	Exits route-map configuration mode and enters global configuration mode.
Step 27	**exit** **Example:** `Router(config)# exit`	Exits global configuration mode.

Configuration Examples for mVPNs

mVPN Configuration with Default MDTs Only

The following excerpt from a configuration file shows the lines that are related to the mVPN configuration for three mVRFs. (The required BGP configuration is not shown.)

```
!
service timestamps debug datetime msec
service timestamps log datetime msec
service password-encryption
service compress-config
!
hostname MVPN Router
!
boot system flash slot0:
logging snmp-authfail
!
ip subnet-zero
!
no ip domain-lookup
ip host tftp 223.255.254.238
!
ip vrf mvpn-cus1
 rd 200:1
 route-target export 200:1
 route-target import 200:1
 mdt default 239.1.1.1
!
ip vrf mvpn-cus2
 rd 200:2
 route-target export 200:2
 route-target import 200:2
 mdt default 239.1.1.2
!
ip vrf mvpn-cus3
 rd 200:3
 route-target export 200:3
 route-target import 200:3
 mdt default 239.1.1.3
!
ip multicast-routing
ip multicast-routing vrf mvpn-cus1
ip multicast-routing vrf mvpn-cus2
ip multicast-routing vrf mvpn-cus3
ip multicast multipath
frame-relay switching
mpls label range 4112 262143
mpls label protocol ldp
mpls ldp logging neighbor-changes
mpls ldp explicit-null
mpls traffic-eng tunnels
mpls tdp discovery directed-hello accept from 1
```

```
mpls tdp router-id Loopback0 force
platform flow ip destination
no platform flow ipv6
platform rate-limit unicast cef glean 10 10
platform qos
platform cef error action freeze

...

vlan internal allocation policy ascending
vlan access-log ratelimit 2000
!
vlan 2001-2101,3501-3700,4001,4051-4080,4093
!
!
!
interface Loopback0
 ip address 201.252.1.14 255.255.255.255
 ip pim sparse-dense-mode
!
interface Loopback1
 ip address 209.255.255.14 255.255.255.255
!
interface Loopback10
 ip vrf forwarding mvpn-cus1
 ip address 210.101.255.14 255.255.255.255
!
interface Loopback11
 ip vrf forwarding mvpn-cus1
 ip address 210.111.255.14 255.255.255.255
 ip pim sparse-dense-mode
!
interface Loopback12
 ip vrf forwarding mvpn-cus1
 ip address 210.112.255.14 255.255.255.255

...

!
interface GigabitEthernet3/3
 mtu 9216
 ip vrf forwarding mvpn-cus3
 ip address 172.10.14.1 255.255.255.0
 ip pim sparse-dense-mode
!

...

!
interface GigabitEthernet3/19
 ip vrf forwarding mvpn-cus2
 ip address 192.16.4.1 255.255.255.0
 ip pim sparse-dense-mode
 ip igmp static-group 229.1.1.1
 ip igmp static-group 229.1.1.2
 ip igmp static-group 229.1.1.4
!
interface GigabitEthernet3/20
 ip vrf forwarding mvpn-cus1
 ip address 192.16.1.1 255.255.255.0
 ip pim sparse-dense-mode
!
...
```

mVPN Configuration with Default and Data MDTs

The following sample configuration includes three mVRFs that have been configured for both default and data MDTs. Only the configuration that is relevant to the mVPN configuration is shown.

```
...
!
ip vrf v1
 rd 1:1
 route-target export 1:1
 route-target import 1:1
 mdt default 226.1.1.1
 mdt data 226.1.1.128 0.0.0.7 threshold 1
!
ip vrf v2
 rd 2:2
 route-target export 2:2
 route-target import 2:2
 mdt default 226.2.2.1
 mdt data 226.2.2.128 0.0.0.7
!
ip vrf v3
 rd 3:3
 route-target export 3:3
 route-target import 3:3
 mdt default 226.3.3.1
 mdt data 226.3.3.128 0.0.0.7
!
ip vrf v4
 rd 155.255.255.1:4
 route-target export 155.255.255.1:4
 route-target import 155.255.255.1:4
 mdt default 226.4.4.1
 mdt data 226.4.4.128 0.0.0.7
!
ip multicast-routing
ip multicast-routing vrf v1
ip multicast-routing vrf v2
ip multicast-routing vrf v3
ip multicast-routing vrf v4
mpls label protocol ldp
mpls ldp logging neighbor-changes
mpls tdp router-id Loopback1
platform ip multicast replication-mode ingress
platform ip multicast bidir gm-scan-interval 10
no platform flow ip
no platform flow ipv6
platform cef error action freeze
!

...

vlan internal allocation policy ascending
vlan access-log ratelimit 2000
!
!
interface Loopback1
 ip address 155.255.255.1 255.255.255.255
 ip pim sparse-mode
!
interface Loopback11
 ip vrf forwarding v1
 ip address 155.255.255.11 255.255.255.255
```

```
 ip pim sparse-dense-mode
!
interface Loopback22
 ip vrf forwarding v2
 ip address 155.255.255.22 255.255.255.255
 ip pim sparse-mode
!
interface Loopback33
 ip vrf forwarding v3
 ip address 155.255.255.33 255.255.255.255
 ip pim sparse-mode
!
interface Loopback44
 ip vrf forwarding v4
 ip address 155.255.4.4 255.255.255.255
 ip pim sparse-mode
!
interface Loopback111
 ip vrf forwarding v1
 ip address 10.1.1.1 255.255.255.252
 ip pim sparse-dense-mode
 ip ospf network point-to-point
!
interface GigabitEthernet1/1
 description Gi1/1 - 155.50.1.155 255.255.255.0 - peer dut50 - mpls
 mtu 9216
 ip address 155.50.1.155 255.255.255.0
 ip pim sparse-mode
 mpls ip
!
interface GigabitEthernet1/2
 ip vrf forwarding v1
 ip address 155.1.2.254 255.255.255.0
 ip pim sparse-mode
!
interface GigabitEthernet1/3
 description Gi1/3 - 185.155.1.155/24 - vrf v1 stub peer 185.Gi1/3
 ip vrf forwarding v1
 ip address 185.155.1.155 255.255.255.0
 ip pim sparse-mode
!

...

!
interface GigabitEthernet1/48
 ip vrf forwarding v1
 ip address 157.155.1.155 255.255.255.0
 ip pim bsr-border
 ip pim sparse-dense-mode
!
interface GigabitEthernet6/1
 no ip address
 shutdown
!
interface GigabitEthernet6/2
 ip address 9.1.10.155 255.255.255.0
 media-type rj45
!
interface Vlan1
 no ip address
 shutdown
!
router ospf 11 vrf v1
```

```
 router-id 155.255.255.11
 log-adjacency-changes
 redistribute connected subnets tag 155
 redistribute bgp 1 subnets tag 155
 network 10.1.1.0 0.0.0.3 area 155
 network 155.255.255.11 0.0.0.0 area 155
 network 155.0.0.0 0.255.255.255 area 155
 network 157.155.1.0 0.0.0.255 area 0
!
router ospf 22 vrf v2
 router-id 155.255.255.22
 log-adjacency-changes
 network 155.255.255.22 0.0.0.0 area 155
 network 155.0.0.0 0.255.255.255 area 155
 network 157.155.1.0 0.0.0.255 area 0
!
router ospf 33 vrf v3
 router-id 155.255.255.33
 log-adjacency-changes
 network 155.255.255.33 0.0.0.0 area 155
!
router ospf 1
 log-adjacency-changes
 network 155.50.1.0 0.0.0.255 area 0
 network 155.255.255.1 0.0.0.0 area 155
!
router bgp 1
 bgp router-id 155.255.255.1
 no bgp default ipv4-unicast
 bgp log-neighbor-changes
 neighbor 175.255.255.1 remote-as 1
 neighbor 175.255.255.1 update-source Loopback1
 neighbor 185.255.255.1 remote-as 1
 neighbor 185.255.255.1 update-source Loopback1
 !
 address-family vpnv4
 neighbor 175.255.255.1 activate
 neighbor 175.255.255.1 send-community extended
 neighbor 185.255.255.1 activate
 neighbor 185.255.255.1 send-community extended
 exit-address-family
 !
 address-family ipv4 vrf v4
 no auto-summary
 no synchronization
 exit-address-family
 !
 address-family ipv4 vrf v3
 redistribute ospf 33
 no auto-summary
 no synchronization
 exit-address-family
 !
 address-family ipv4 vrf v2
 redistribute ospf 22
 no auto-summary
 no synchronization
 exit-address-family
 !
 address-family ipv4 vrf v1
 redistribute ospf 11
 no auto-summary
 no synchronization
 exit-address-family
```

```
!
ip classless
ip route 9.255.254.1 255.255.255.255 9.1.10.254
no ip http server
ip pim bidir-enable
ip pim rp-address 50.255.2.2 MCAST.MVPN.MDT.v2 override bidir
ip pim rp-address 50.255.3.3 MCAST.MVPN.MDT.v3 override bidir
ip pim rp-address 50.255.1.1 MCAST.MVPN.MDT.v1 override bidir
ip pim vrf v1 spt-threshold infinity
ip pim vrf v1 send-rp-announce Loopback11 scope 16 group-list MCAST.GROUP.BIDIR bidir
ip pim vrf v1 send-rp-discovery Loopback11 scope 16
ip pim vrf v1 bsr-candidate Loopback111 0
ip msdp vrf v1 peer 185.255.255.11 connect-source Loopback11
ip msdp vrf v1 cache-sa-state
!
!
ip access-list standard MCAST.ANYCAST.CE
 permit 2.2.2.2
ip access-list standard MCAST.ANYCAST.PE
 permit 10.1.1.1
ip access-list standard MCAST.BOUNDARY.VRF.v1
 deny   226.192.1.1
 permit any
ip access-list standard MCAST.GROUP.BIDIR
 permit 226.192.0.0 0.0.255.255
ip access-list standard MCAST.GROUP.SPARSE
 permit 226.193.0.0 0.0.255.255
ip access-list standard MCAST.MVPN.BOUNDARY.DATA.MDT
 deny   226.1.1.128
 permit any
ip access-list standard MCAST.MVPN.MDT.v1
 permit 226.1.0.0 0.0.255.255
ip access-list standard MCAST.MVPN.MDT.v2
 permit 226.2.0.0 0.0.255.255
ip access-list standard MCAST.MVPN.MDT.v3
 permit 226.3.0.0 0.0.255.255
ip access-list standard MCAST.MVPN.RP.v4
 permit 227.0.0.0 0.255.255.255
!
access-list 1 permit 226.1.1.1
access-list 2 deny   226.1.1.1
access-list 2 permit any
...
```

Verifying the mVPN with L3VPN over mGRE Configuration

Use the following examples to verify that the configuration is working properly:

Endpoint Creation

You can verify the tunnel endpoint that has been created:

```
Router# show tunnel endpoints tunnel 0

 Tunnel0 running in multi-GRE/IP mode

 Endpoint transport 209.165.200.251 Refcount 3 Base 0x2AE93F0 Create Time 00:00:42
   overlay 209.165.200.254 Refcount 2 Parent 0x2AE93F0 Create Time 00:00:42
```

Adjacency

You can verify that the corresponding adjacency has been created:

```
Router# show adjacency tunnel 0

   Protocol Interface                 Address
  IP       Tunnel0                   209.165.200.251(4)
  TAG      Tunnel0                   209.165.200.251(3)
```

Profile Health

You can use **show l3vpn encapsulation** *profile-name* command to get information on the basic state of the application. The output of this command provides you details on the references to the underlying tunnel.

```
Router# show l3vpn encapsulation ip tunnel encap

    Profile: tunnel encap
    transport ipv4 source Auto: Loopback0
    protocol gre
    Tunnel Tunnel0 Created [OK]
    Tunnel Linestate [OK]
    Tunnel Transport Source (Auto) Loopback0 [OK]
```

Configuration Sequence for mVPN with L3VPN over mGRE

This example shows the configuration sequence for mVPN with L3VPN over mGRE:

```
vrf definition Customer A
 rd 100:110
 route-target export 100:1000
 route-target import 100:1000
 !
 address-family ipv4
 exit-address-family
 !
 address-family ipv6
 exit-address-family
!
!
ipv6 unicast-routing
!
l3vpn encapsulation ip sample_profile_name
 transport ipv4 source loopback 0
!
!
 interface Loopback0
  ip address 209.165.200.252 255.255.255.224
  ip router isis
!
interface gigabitethernet 1/1
 vrf forwarding Customer A
 ip address 209.165.200.253 255.255.255.224
 ipv6 address 3FFE:1001::/64 eui-64
!
router bgp 100
 bgp log-neighbor-changes
 neighbor 209.165.200.254 remote-as 100
 neighbor 209.165.200.254 update-source Loopback0
 !
 address-family ipv4
  no synchronization
  redistribute connected
  neighbor 209.165.200.254 activate
```

```
 no auto-summary
exit-address-family
!
address-family vpnv4
 neighbor 209.165.200.254 activate
 neighbor 209.165.200.254 send-community both
 neighbor 209.165.200.254 route-map SELECT_UPDATE_FOR_L3VPN in
exit-address-family
!
address-family vpnv6
 neighbor 209.165.200.254 activate
 neighbor 209.165.200.254 send-community both
 neighbor 209.165.200.254 route-map SELECT_UPDATE_FOR_L3VPN in
exit-address-family
!
address-family ipv4 vrf Customer A
 no synchronization
 redistribute connected
exit-address-family
!
address-family ipv6 vrf Customer A
 redistribute connected
 no synchronization
exit-address-family
!
!
route-map SELECT_UPDATE_FOR_L3VPN permit 10
set ip next-hop encapsulate sample_profile_name
set ipv6 next-hop encapsulate sample_profile_name
```

Tip For additional information about Cisco Catalyst 6500 Series Switches (including configuration examples and troubleshooting information), see the documents listed on this page:

http://www.cisco.com/en/US/products/hw/switches/ps708/tsd_products_support_series_home.html

Participate in the Technical Documentation Ideas forum